Institute of Software Technology

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

A Systematic Approach to extend the common Software Testing Types with modules specific to the field of Machine Learning

Sweekar Revanna

Course of Study: Information Technology

Examiner: Prof. Dr. Stefan Wagner

Supervisor: M.Sc. Jonas Fritzsch

Commenced: March 27, 2019

Completed: September 27, 2019

Abstract

In this data-driven age, many Machine learning (ML) or predictive analytics related software applications are developed, utilizing the data to extract knowledge and provide insights to the customers. Software testing plays an important role in assuring the quality of a software application. Hence, there is a need to combine these two distinctive domains and develop a systematic approach to detect the errors in the ML by practicing the principles of software testing. Recent publications emphasize the necessity of testing the ML model, the aspects to test in the ML domain and provide suggestions for possible tests. However, the extent and rigor of software testing principles as specified in the ISO/IEC/IEEE 29119 series was not sufficiently considered by these publications. Therefore, this thesis focuses on the applicability of common software testing types, techniques, and methods to the field of ML. To do so, we determine defects and errors that affect the model quality through expert interviews, literature review and occurrence frequency of the defects in a discussion group of the Kaggle website. Unit testing, data quality checks, and model evaluation metrics were common Quality Assurance (QA) practices followed by the interviewed data scientists. As the main contribution, this thesis presents a set of automated tests based on the ISO/IEC/IEEE 29119 series of software testing standards that check data and model for the ML related defects. The automated tests were evaluated against publicly available data and the codes of ML model for the competition, "Titanic: Machine Learning from Disaster" on Kaggle website. The evaluation revealed defects hidden in the data and the model, demonstrating the benefit of our proposed extension of software testing principles to the ML domain. Furthermore, from seven data scientists indicated that these additional tests are easily understandable and usable.

Contents

List of Figures

List of Tables

List of Listings

List of Abbreviations

AUC Area Under the Curve. 37

ML Machine learning. 3

NA Not Available. 49

NaN Not a Number. 49

NLP Natural Language Processing. 17

QA Quality Assurance. 3

RQ1 Research Question 1. 17

RQ2 Research Question 2. 17

RQ3 Research Question 3. 17

SDLC Software Development Life Cycle. 30

SVM Support Vector Machines. 27

UI User Interface. 23

1 Introduction

The world of software engineering has seen an evolution from the basic calculating engine to recent trends of Cloud Computing, Big Data, and Machine learning (ML). With each of these transitions taking place, software engineering objectives are made to adapt to these new trends thereby motivating the exploration of new and reliable methods and approaches [ABB+19]. Currently, a lot of advancements are made in the field of ML and data science, which is continuously growing day by day. Research in the field of data science is carried by a wide range of industries like finance, health, entertainment, and transportation, etc. Companies such as Google [1], Amazon [2], Uber [3], etc. are using data science to bring out knowledge and insights about their users. This helps in understanding the user needs and developing ML-based applications to provide a customer-tailored personal experience to their users. However, the need to ensure the quality of ML-based applications is significant [Gér17]. Furthermore, apart from the testing of the application itself, testing of ML-based programs require testing the performance of the models (algorithms), as well as the quality of the data, fed to the models [BK18].

1.1 Motivation

Software testing plays an important role in determining the quality of a software application or product. In the field of software engineering, the software testing or QA phase holds equal importance as that of other phases like requirement analysis, software development, etc. [JAAA16]. Traditional software applications produce deterministic results that aid in evaluating software quality against a set of expected outcomes [Buc19]. At present, there are many practices for performing software testing both on functional and non-functional aspects of the software [CD13]. In this age of knowledge discovery, specific areas of data science, named as ML and predictive analytics, aim in utilizing data to extract knowledge useful for prognosis. In ML, models based on algorithms identifying statistical patterns with past data provide valuable probabilistic predictions about future events [MKA06].

From the user's perspective, the results from predictive analytics models (predicting outcomes based on newly collected input data) are sometimes taken for granted. However, Géron.A [Gér17] mentions the existence of problems like overfitting, underfitting, etc. which affects the performance of the ML model in the production phase []. He also mentioned that these problems cannot be easily detected when we only evaluate the model performance after training the model with a

[1] Google Data Science: https://developers.google.com/analytics/solutions/data-scientists
[2] Amazon Data Science : https://www.bernardmarr.com/default.asp?contentID=712
[3] Uber Data science : https://georgianpartners.com/data-science-disruptors-uber-uses-applied-analytics-competitive-advantage/

cross-validation strategy. For example, the ML model might display very high accuracy only after its first training, since there are chances that the model had learned the outcomes, based on limited training data, but then start to predict incorrectly when requested to score freshly on new data.

"Classical" software testing phases like unit testing and system testing [CD13], validate the software based on expected outcomes. On the other side, model performance indicators are considered to be the metrics of evaluation in ML. Metrics like accuracy, precision, recall, etc., obtained from the confusion matrix[4] are evaluated to measure the model's prediction quality [Gér17]. On seeing the high accuracy of the ML model, the stakeholders like the business team or sometimes the customers certify the model as applicable to the daily business [MKA06] [Buc19]. Nevertheless, some hidden bugs and errors in the ML model can be hard to spot while the model is being developed [MKA06]. Considering the use case of customer analytics, there may be situations where the model is developed based on the historical data of a customer's product preference. It would be important to analyze the quality of predictions the model makes before the ML model is deployed onto a software product.

Hence, there is a need to have an approach involving the principles of software testing coupled with methods to spot the errors or problems present in ML models, which lead to underperformance while in production usage. This research provides an approach to apply the software testing types and phases, to ML specific modules. This mitigates the problems arising during the production usage. Similarly, data scientists can leverage from this work by checking for the errors in the ML model they develop and evaluating its quality. By doing this, they can guarantee that the model they built is free from certain errors like overfitting, underfitting, leakage, etc. Thereby, the research helps the data science practitioners to be cognizant of the model quality along with the results it produces.

1.2 Goal

The goal of the thesis is to evaluate whether the aspects of software testing like test phases, test techniques, test types, etc. [CD13] can be simulated for the ML platform. To achieve this goal, at first, a class of defects in ML models will be identified. Next, we determine the reasons for the process of testing being insufficient, when the same is performed in the domain of ML similar to the way it is conducted in software applications. Suitable additional testing types would be suggested for a specific type of defect concerning the field of ML. The additional tests suggested would be evaluated. Based on the results of the evaluation, conclusions would be made whether a particular defect can be identified by conducting the additional tests and the extent of similarity with the software testing practices.

[4]Confusion Matrix: https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/

1.3 Research Questions

The emphasis of the thesis is on QA, particularly for ML models. The primary focus is to identify the problems and hidden errors which affect the quality of the ML models. We analyze the testing process carried out by the data scientists (considered for our research) in their work activities and assess the adequacy of the testing process to develop better models. Based on the analysis, we suggest additional tests and evaluate the ability of the suggested additional tests to reveal such kind of errors. The following research questions are outlined to accomplish the primary objective:

Research Question 1 (RQ1): What are the problems and hidden errors present in ML approaches that are not covered by Software testing?

It is important to be aware of the different problems and errors that lower the quality of the ML model. The intention is to gather the common defects or errors that data scientists identify while developing the model. This facilitates us in realizing a class of defects present in the field of ML that are different from the defects detected while testing a traditional software application.

Research Question 2 (RQ2): What are the specifics of such problems and reasons for the missing coverage by the existing software testing types and methods?

We understand the practices followed by the data scientists in the direction of QA and compare these practices with the definitions outlined in the ISO/IEC/IEEE 29119 series of software testing standards [Pro13] and test design techniques [Tec15]. Based on the above analysis, we grasp the extent of QA performed to address the defects gathered as a part of the RQ1.

Research Question 3 (RQ3): What additional test processes, types, and techniques can extend established software testing phases to detect hidden errors in ML models?

This question is important to design the additional test processes, types, and techniques required to detect the defects that were drawn out from RQ1. These additional tests are evaluated by conducting the QA process in a similar to the testing of a traditional software application. Thereby, a test catalog is provided to the data scientists. When the data scientists conduct such tests, it makes them aware of the possible errors in the models they have developed.

1.4 Methodological Approach

This section explains the steps that will be pursued to answer the research questions addressed in the previous section.

The methodology to address RQ1 is based on the explorative strategy [SR13] involving three phases. The first is the expert interviews phase, where seven data scientists are interviewed in a semi-structured form [Sea99]. The interviews consist of various open and close-ended questions on defects or errors the data scientists experience when they develop the ML models. The second phase is the literature review. The last phase is analyzing the Kaggle's discussion[5] threads. This is to confirm the availability of the keywords representing the defects obtained from the previous two phases, in the text corpus using Natural Language Processing (NLP) concepts. The text corpus

[5]Kaggle discussion: https://www.kaggle.com/discussion

represents the discussions that have happened on the Kaggle website concerning various topics of data science [Goo19]. Finally, the results are interpreted to address RQ1. The strategy used to answer RQ1 is illustrated in Figure 1.1.

Phase	**Task**	**Reason of Selection**
Expert Interviews	Conduct semi-structured interviews and interpret qualitative answers received from data scientists.	To gain industry insights.
Literature Review	Review the scientific papers, journals and books.	To gain insights into the defects or errors related to ML or data science domain from scientific community.
Programmatic Approach	Extract data of all the discussions(as of 03-05-2019) happened in the website Kaggle [Goo19]. Apply the concepts of NLP on the text corpus collected to check the frequency of occurrence of the keywords. These keywords are related to defects or errors elucidated from the above two phases.	To gain confirmation whether the defects or errors collected from above two phases are also discussed globally.

Table 1.1: Distribution of the phases.

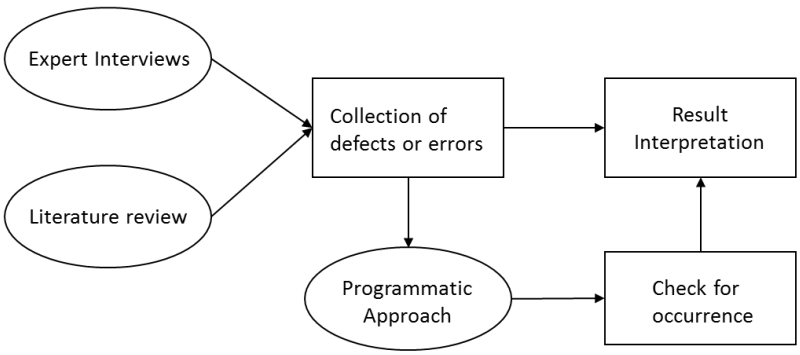


Figure 1.1: Strategy to address RQ1

RQ2 deals with gathering the information on the testing practices followed by the data scientists (considered for the interview) during or after the model development and then analyzing the reasons for not addressing the defects or errors related to the field of ML. Again, the expert interviews are conducted in a semi-structured format with open and closed type questions. Twelve data scientists are considered for analysis and the responses are documented in an interview form.

Based on the results of RQ1 and RQ2, we design suitable additional test processes, types, and techniques required to detect the defects that were extracted while answering RQ1. Later, these are evaluated against a sample code implementation from Kaggle to reveal errors related to ML. This sample code considered for evaluation is chosen from the Kaggle website's famous

binary classification problem known as "Titanic: Machine Learning from Disaster"[6]. Through this evaluation, we would conclude the extent to which the additional test processes, types, and techniques simulate software testing for the ML models as a part of RQ3. Finally, feedback on these additional tests is gathered and analyzed. Seven data scientists provide their feedback on advantages, disadvantages, understandability, and usability aspects of the additional tests designed as a part of the RQ3. The feedback interviews are conducted in a semi-structured format with open and closed type questions. The responses are collected in an interview form.

1.5 Thesis Outline

This thesis is organized from further chapters as follows:

Key Concepts (Chapter 2) helps to understand the basics of the topics of data science, ML, Kaggle, and software testing based on which the thesis is concentrated.

Related Work (Chapter 3) explains the research carried out in the domain of QA for ML or data science.

Scope (Chapter 4) describes the data science process practiced for a customer analytics use case and the assumptions considered as a part of this thesis. Through this chapter, the scope of the thesis is defined.

Results (Chapter 5) provides the results for each research question and explains the detailed approaches taken in achieving the results. This chapter forms the crux of the thesis.

Conclusion (Chapter 6) concludes this thesis by summarizing the results obtained in the previous chapter. This section also mentions the threats faced while validating the obtained results. Lastly, a section about future work mentioning the open topics that can be continued in the line of this research field.

[6]Kaggle Titanic: https://www.kaggle.com/c/titanic

2 Key Concepts

The key terms discussed in the introduction chapter of the thesis were Data Science, ML, Software testing, and Kaggle. Some of these topics are interlinked and hence there is a possibility of bewilderment between these topics. In the following sections, an explanation about these concepts are provided to aid in grasping the specifics about the master thesis.

2.1 Data Science

The domain under consideration for the thesis is data science. Hence giving details about the definition and the data science process pipeline helps to understand the domain better and get familiarized with the terminologies of this domain.

In 2006, a UK Mathematician and architect of Tesco's Clubcard, Clive Humby quoted:

"Data is the new oil!" [Mic19].

Data in its raw form would not generate any value for business. Whereas, if the same is cleaned and transformed. We can generate useful insights from the transformed data [Mic19].

Every second, data is generated in huge amounts in domains like entertainment, health analytics, e-commerce, astronomy, tech companies and, etc. So the stakeholders of these domains feel the necessity of using the data systematically to improve the business decision process [PF13]. Different authors and data scientists define data science in their way. Often data science is thought to be analogous to data mining or data analytics because the techniques involved to carry out data science, data mining or data analytics are fundamentally the same [LRU14]. Some of the definitions on Data Science, mentioned in few publications are provided below.

Provost. T and Fawcett, T [PF13] have defined the term data science as the principles, processes, and techniques related to deriving valuable information by analyzing the historical dataset for understanding a phenomenon or a business problem. Furthermore, the authors think that in the present day, scientific and business communities are focused on data-driven decision-making and their success is determined by how well principles, processes, and techniques are applied to analyze data for solving a particular problem or a phenomenon. So the authors believe that along with the data scientists, other members who closely work with data scientists in an organization are also expected to be cognizant of the essential principles of data science.

Another definition was provided by Schutt. R et al. [OS13] in which the authors concluded data science to be a set of activities performed by a data scientist. According to the authors, the skill levels required by a data scientist are Computer Science, Mathematics, Statistics, Machine Learning, Data Visualization, Domain Expertise, Effective Communication, and Presentation Skills. A data scientist effectively utilizes all the mentioned skills to perform a variety of tasks starting from the

setting up of the infrastructure necessary to collect data from various sources for which the skills of computer science is required and cleaning data based on the statistical skills. Upon cleaning the data, a data scientist visualizes the data and conducts exploratory data analysis. This is followed by the generalization of patterns hidden in the data and employ ML algorithms to build a relevant model. Testing the model with another set of data to verify and validate the quality so that the model can be encapsulated as a software product. Finally communicating the results or hypotheses with other stakeholders, like employees of the business team for which the data models are used, and receiving a feedback, if applicable, completes the process. The results are interpreted by making use of graphs and charts, to convey them in a clearer and effective manner [OS13].

Given the above, data science can be defined as a science consisting of a set of best practices where the data is utilized to decipher a broad space of problems. [OS13]

2.1.1 General Data Science Process

In Section 2.1 details regarding data science definition, activities of data scientist, and the process of data science was summarized. In this section, the data science process is explained in more detail. This is necessary to get acquainted with the different phases of the data science domain and thereby envision the necessity of QA for different phases. Schutt. R et al. [OS13] have the depicted general framework followed in any data science project in Figure 2.1 and they have explained in the following way.

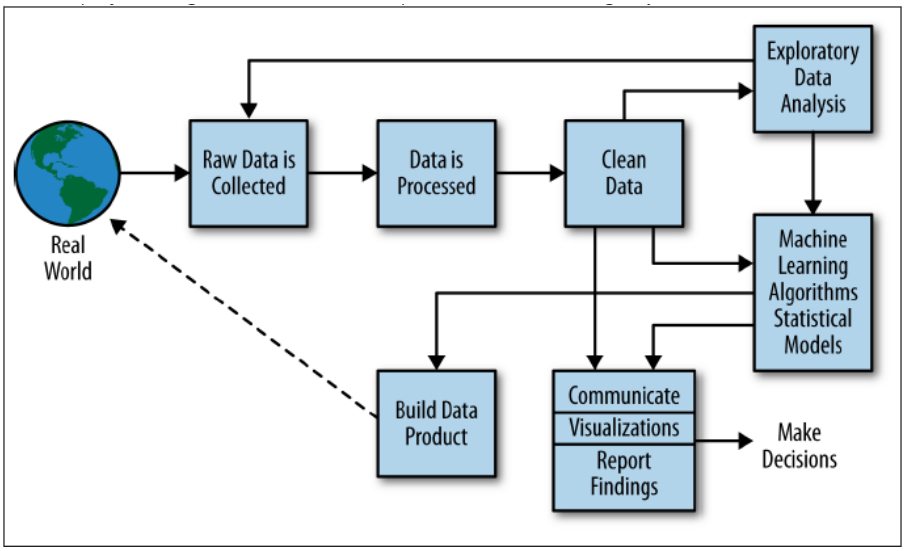


Figure 2.1: Data science process [OS13]

In Figure 2.1, practical situations in life or the entities who are a part of that situation defining a problem are depicted as the Real World. Patients getting checked up at a hospital for the detection of the flu, people who are using Amazon to buy different products, weather forecasting, etc. are all part of Real World. Raw data can be the records of inpatients and outpatients in a hospital, the user clicks on a particular website, etc. Data collection methods can include web scraping, making API calls, or data retrieving from a database. As a next step, the data is processed and cleaned to transform data required for analysis. Programming languages like Python, R or SQL helps in

carrying out these tasks. As a result, the data is obtained in a usable and required format. Generally, data is represented in a tabular format. Sometimes data is also presented in JSON, CSV and xlsx formats.

Per se, the cleaned data is not free from impurities. There exist duplicate rows or columns, missing values, incorrect data, and outliers. Exploratory data analysis ensures such irregularities are detected and the stages of data collection, processing and cleaning must be performed again to obtain sufficiently clean data which helps in generating good results.

Typically a data scientist is challenged with a classification problem, regression problem, segmentation or a description problem. Based on the problem statement data scientists use a corresponding ML algorithm or statistical methods like Naive Bayes, decision trees, linear regression, etc. to build a model. Next, the model is exposed to a set of data known as the Training dataset so that the model can detect a pattern thereby assisting itself in solving the defined problem.

Once the model is "built", its performance is evaluated with another set of data to which the model was not exposed previously known as the Testing dataset. The data scientist elucidates the model behavior with the problem statement, conceptualizes the graphs and communicates the result to the business team if he/she is convinced that the developed model exhibits good performance upon validation. If not, then he/she understands the reason for the depreciated behavior of the model, retrain the model by correcting the relevant bugs and perform the previously mentioned step. The communication skills of a data scientist play a significant part in persuading the stakeholders about the results.

As a final stage, it is also important to incorporate the model into software by developing the ML software product and present it in front of the "real world" again for interaction with the product. The dotted line indicates the partial completion of a feedback loop. Consider the instance of the YouTube[1] application and the weather forecast application. The data scientists analyze the users' behavior, for example, the choice of songs the users are listening to in case of a music streaming application and the data scientists develop a model that recommends users a list of new songs similar to what the users previously were listening to [CAS16]. We notice that such a system is influencing the behavior of the users or the "real world".

Another instance where there is no feedback loop visible in the data science process is weather forecast application. Based on the data obtained from the sensors stationed at various locations, the data scientists can predict the weather at a particular place and the same would be displayed on a User Interface (UI) [JHR18]. In this case, the users are just obtaining information about the weather. The behavior of the users is not influenced by the ML software product.

Based on the above explanation Figure 2.1 can be enhanced into Figure 2.2 by combining the data processing and data cleaning as the data processing stage. The output of the modeling stage is a model (ML or statistical) that would be embedded into a software product and the model's result would be communicated to the audience.

[1] Youtube recommendations : https://towardsdatascience.com/using-deep-neural-networks-to-make-youtube-recommendations-dfc0a1a13d1e

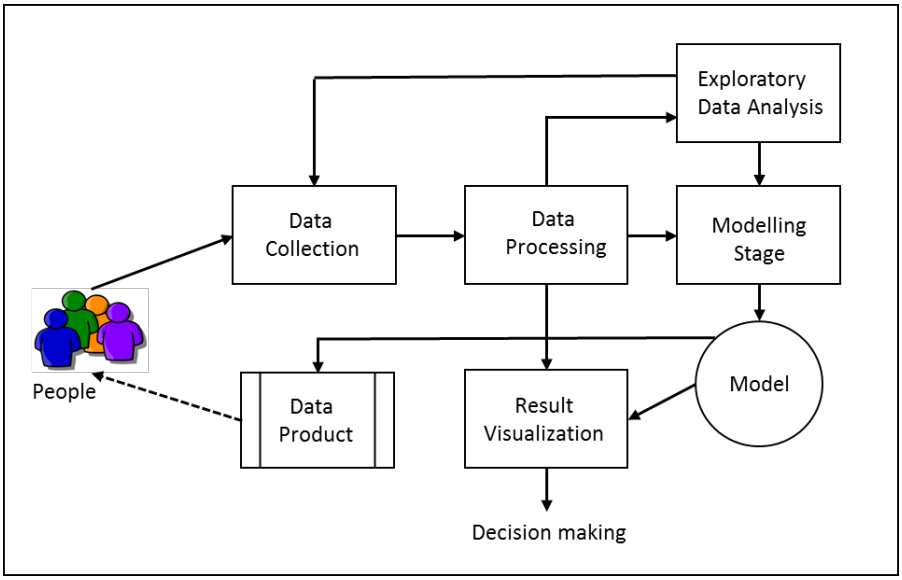


Figure 2.2: Enhanced Data science process [OS13]

The definition of data science and the explanation about the data science process pipeline introduces us to the domain thereby aiding us to understand the distinctness in performing a data science and a product resulting out of such a process to that of a general software application. This sub section also motivates us to know about the ML model.

2.2 Machine Learning

We came to know from the previous section that the output of the modeling stage in the data science process is a "model". This model can be developed based on statistics or ML [OS13]. So this subsection explains the definition of ML, its types and the names of some algorithms.

Based on the daily traveling experience, certain routes can be classified as busy and we would like to avoid such routes to reach our destination quickly. Similarly, during the time of winter, we can predict that the price of winter jackets are generally expensive and vice versa at the time of summer. Normally people based on their prior experience, tend to predict outcomes of events, classify things or situations to different categories or even to describe an event based on the relationship between other events. The process when a machine or system learns the patterns in the dataset provided as input and predicts the likelihood of possible outcomes can be termed as ML [Gér17].

Arthur Lee Samuel in 1959, defined ML as the class of algorithms by which computers or any machines would be able to detect and generalize a pattern based on the data provided as an input and reveal the possible outcomes of an event without any manual intervention [Sam00]. These algorithms should also have the capability to learn and improve their results when they are exposed to a fresh set of data. Figure 2.3 shows the most common types of ML, classified in terms of the training method [Gér17].

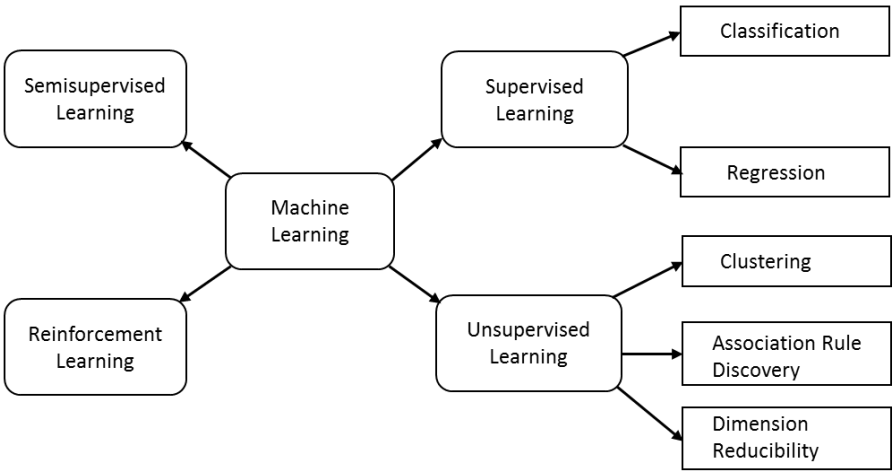


Figure 2.3: ML Types [Gér17]

2.2.1 Types of Machine Learning

The categorization of ML is based on various factors [Gér17]. Each use case in data science corresponds to one of the below-mentioned types and each ML type can be solved in a specific way. It is important to be aware of the different types in ML because whenever a data scientist is given a task, the task can be related to a particular ML type and corresponding algorithms of that ML type can be used to solve the task. Based on the way humans invigilate the process of training an algorithm, the types of ML can be distinguished as follows.

1. Supervised learning

   When the target or the required result for the given problem is cognized prior by the algorithm and it is included as a variable referred to "labels" along with other factors referred to as "predictors" (these predictors help in determining the patterns) in the dataset. This type is further classified into 2 subtypes, namely,

   a) Classification - The algorithm needs to distinguish between a given set of classes as the label. A typical example being the test for the flu: where many examples of patients' test reports are trained with the classes (ill or normal) to classify new reports [Gér17].

   b) Regression – The purpose of the algorithm is to predict the numeric value, percentages, or probabilities as the label. Predicting the exchange rates of a currency or the prices of diesel can be quoted as examples.

2. Unsupervised learning

   When the target or the label for the given problem is unavailable in the dataset and the algorithm needs to generalize and identify the groups. This type is further divided into the following types:

   a) Clustering – This type is employed when the problem is to identify the time during which the gym is occupied with maximum visitors. The algorithm is not revealed with this information and is expected to categorize on its own. The algorithm outputs the

result in the form like during weekdays 30% of the visitors use the gym in the morning, 40% of the visitors use the gym in the evening, 20% of the visitors use the gym in the night, and 10% of the visitors use the gym in the afternoon. Whereas during weekends 35% of the visitors use the gym in the morning, 30% of the visitors use the gym in the evening, 5% of the visitors use the gym in the night, and 30% of the visitors use the gym in the afternoon.

b) Visualization – This type keeps the original structure of the data intact and assigns separate clusters to distinguish between the overlapping data thereby assisting the identification of undetected patterns. Generally, the output is realized in the form of 2D or 3D graphs [Gér17].

c) Association Rule discovery – This type generally describes rules that results in an event based on the association of different events. For example person A is expected to pass in the subject 'XX' if he passes the subjects 'XY' and 'YY'.

3. Semisupervised learning

   This is a special type where the label has values filled for a few instances and there are no values for the remaining large number of instances. From the name, it indicates it is a coalition of both supervised and unsupervised learning. Geron, A [Gér17] considered a practical example of an application upon uploading a number of photos captured during the vacation are able to identify me since the application is already aware of me and it also identifies a friend of mine, who was with me on the vacation in photos 2, 4, 6, and 9, while a random person who has also appeared in some photos like 1 and 2. The algorithm in the application performs unsupervised learning. Upon classifying the person with names, then the application performs supervised learning by identifying us with names in all the photos.

4. Reinforcement learning

   This is a new form of ML where an agent which is usually the algorithm inspects the surroundings, decides and executes an activity and receives positive points termed as a *reward* in case of a success or negative points termed as *penalties* upon a failure [Gér17]. The algorithm repeatedly acts over time and receives rewards or penalties depending on success or failure until it familiarizes itself with the best strategy (termed as "*policy*") and obtains maximum points [Gér17]. When the algorithm is presented with a similar scenario, the policy drives the agent to select the necessary action. Figure 2.4 provided by Geron. A [Gér17] provides an example to help us clearly understand the meaning of reinforcement learning.

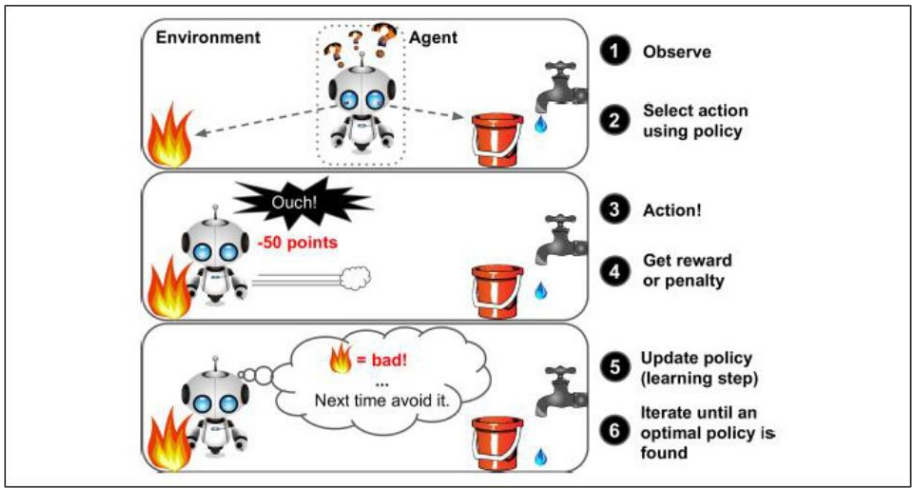


Figure 2.4: Reinforcement learning [Gér17]

2.2.2 Machine Learning Algorithms

The algorithms are utilized to develop the ML model. Being familiarized with these names helps us to avoid ambiguity when the names of the algorithms appear in further chapters. Some of the common ML algorithms discussed in the book [Gér17] are:

- Linear Regression
- Logistic Regression
- Decision Trees
- Support Vector Machines (SVM)
- Naive Bayes Forests
- k- Nearest Neighbors
- k- Means
- Principal Component Analysis
- Apriori

2.3 Kaggle

Since the data and the codes from one of the popular competition on Kaggle are used in the thesis for evaluating the concept developed, it is necessary to provide a slight overview of this website and introduce Kaggle. Kaggle is an online forum that brings aspiring and experienced data scientists and ML engineers closer to amplify one's knowledge by learning various courses offered by Kaggle,

participating in competitions, discussing questions or problems in code snippets by posting it on Kaggle. Currently, Kaggle is owned by Google LLC [2]. People from various countries with diverse skill-sets are members of this community [3].

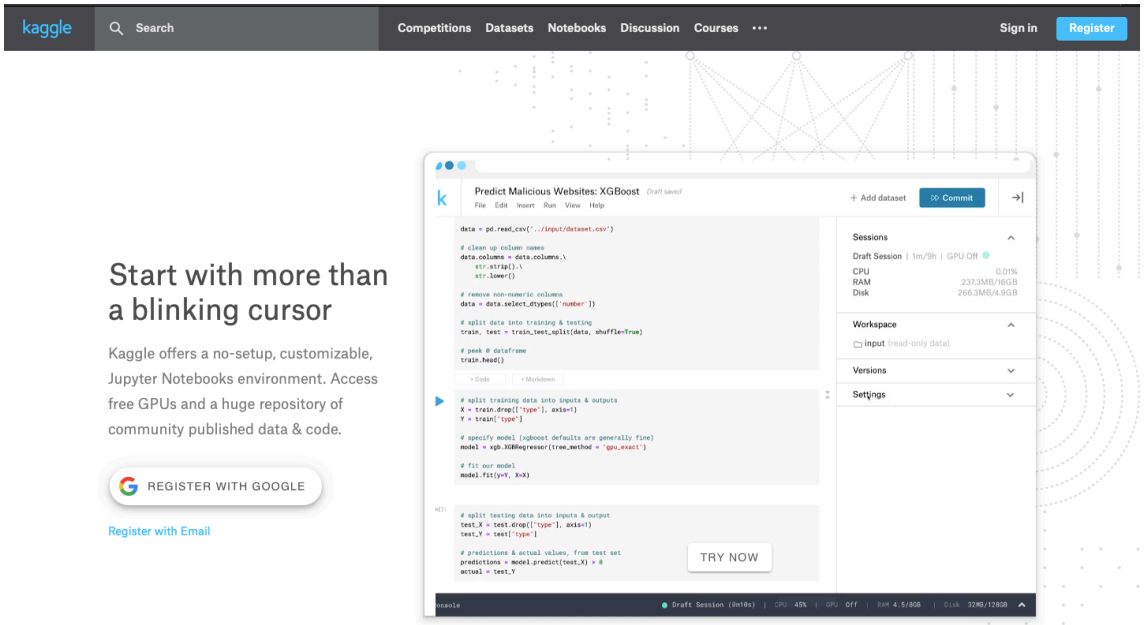


Figure 2.5: Kaggle homepage [Goo19]

Figure 2.5 [Goo19], shows the home screen of the Kaggle website with the features it has to offer for its users at the top. The main features being Competitions, Datasets, Kernels, Discussions, and Courses. Various competition is posted in Kaggle by several companies over different topics as shown in Figure 2.6 [Goo19]. Some of the competitions are hosted to enable the users to get hands-on experience with ML and such competitions will be labeled as Knowledge. [Goo19]

[2]Google buys Kaggle: https://www.cnet.com/news/google-buys-kaggle-and-its-gaggle-of-ai-geeks/
[3]Kaggle Official Blog: http://blog.kaggle.com/2017/06/06/weve-passed-1-million-members/

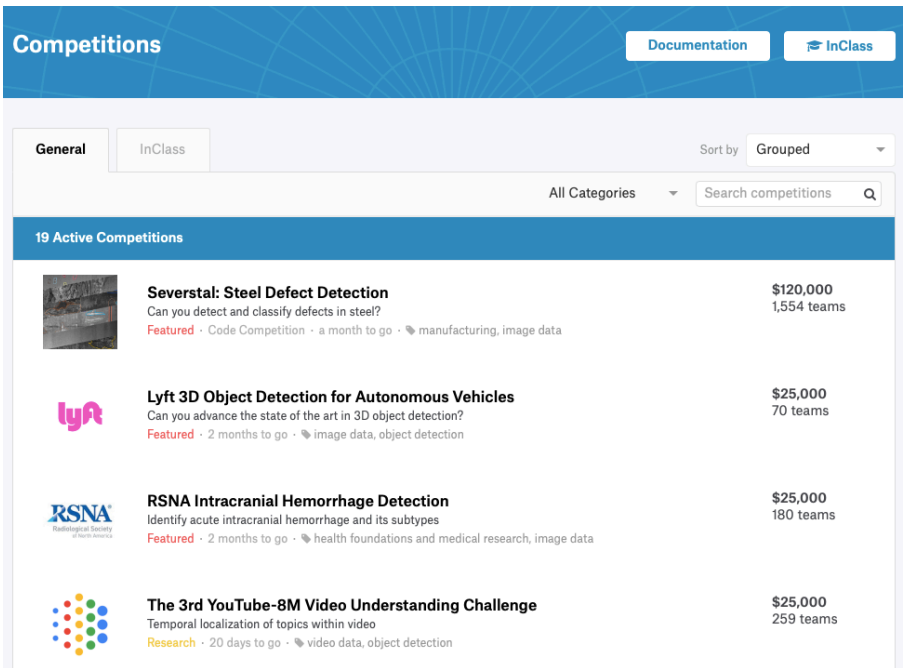


Figure 2.6: Kaggle competitions [Goo19]

Under each competition, an explanation about the overview, training dataset, testing dataset, and the expected result is given. Also, the required datasets are attached. These datasets are real data that can be used by the public based on their interest to practice solving real-world problems. At times business firms also use these datasets to build proof of concepts for their respective projects. Figure 2.7 [Goo19], displays the different datasets.

Kernels are the discussion threads where the ML domain experts upload their code written in python or R languages coupled with their insights, output of the code and visualizations in Kaggle. Sometimes novice users also upload their code and ask questions to get some concepts clarified. Some ML experts answer these questions in the form of comments and replies. This can serve as a knowledge base for users to know about certain aspects of ML in-depth and as a solution to the questions posted earlier in a step-wise manner. The same can be seen in Figure 2.8 [Goo19].

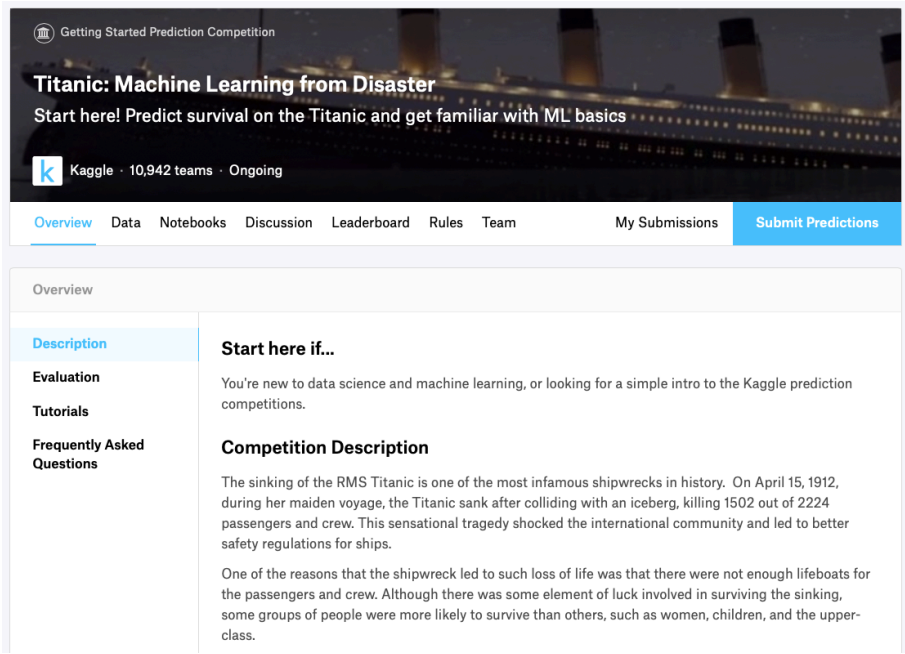


Figure 2.7: Data for a competition [Goo19]

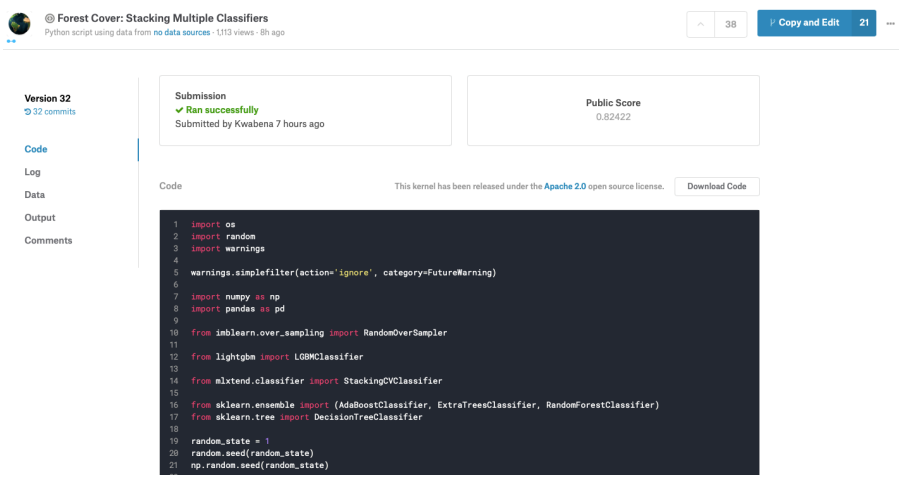


Figure 2.8: Kernels in Kaggle [Goo19]

2.4 Software Testing

Software testing is an integral part of the Software Development Life Cycle (SDLC) [CD13]. The master thesis is based on extending the software testing aspects to the domain of ML, so it is relevant to define and explain software testing. Software testing has been defined in many ways, one such way to define it can be the application of processes, techniques, and methods to assert that

a software application developed is performing correct functionalities as intended [CD13][AO16]. Testing confirms the qualification of a software product to be used. Different stakeholders have a different perspective for testing [Lim09] [CD13]:

1. Test manager – software testing should ensure that the software application developed fulfills the business requirements and the application should not produce errors while the customer or the user uses it.
2. Software Tester – testing is performed to identify all possible errors in the code, mistakes, and faults in the previous stages of software development.
3. Customer/User – The customer or user expects the product to be defect-free as a result of testing.

The importance of testing is explained in the [CD13] as follows.

1. Software testing ensures quality and provides a saleable aspect to an application as needed by the product owners.
2. Testing reveals the defects, bugs or errors present in the software. Since the expenditure incurred in correcting a bug or defect in software during the production phase is very high.
3. Testing affirms that the software product is working correctly.
4. It also ensures that the software is performing its functions in the right manner.
5. Good quality products motivate the customers to use it more and hence results in good business.

2.4.1 Testing Processes

The entire test management process (commonly) consists of three stages: Test Planning process, Test Execution process, and finally Test Completion process [CD13] [AO16]. In the test planning process, requirements translated into a design document would be further formulated into test cases, test scenarios, and test scripts. In the test execution process, the steps outlined in the artifacts mentioned in the previous sentence are carried out to evaluate the software application's actual behavior as against that of the expected behavior. If the actual and expected behaviors match, then the test completion process is carried out, else a defect is reported indicating this deviation. In the test completion process, the execution results of all the test cases, scenarios, and scripts will be verified and the status of all the defects logged will be checked. Based on these, a test closure report will be prepared to signal the hand-over of the testing activity. Figure 2.9 shows the dynamic test process [Pro13] [CD13]. The dynamic test process thoroughly describes the stages of the test management process.

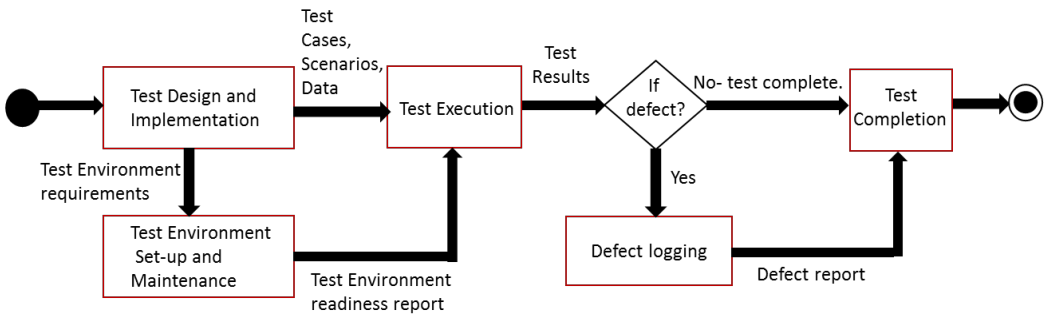


Figure 2.9: Dynamic Testing Process [Pro13] [CD13]

2.4.2 Test Design Techniques

ISO/IEC/IEEE 29119-4 describes the techniques by which test design is performed. The same is shown in Figure 2.10 [Tec15]. The test design techniques facilitate the software testers to understand the business requirements, realize the areas where defects can appear in the software application and finally draft the test cases [Doc13] and test scripts that can be referred during the test execution phase [Tec15]. All these tasks contribute to detecting the defects and thereby improving the quality of the software application.

The requirements and design specifications form the foundation for designing the test cases in the specification-based technique. The source code and its internal structure is the basis for structure-based technique. A tester would have the knowledge of the application or would have seen some errors arisen during the previous releases, based on which test cases would be created, forms the experience-based techniques [Tec15].

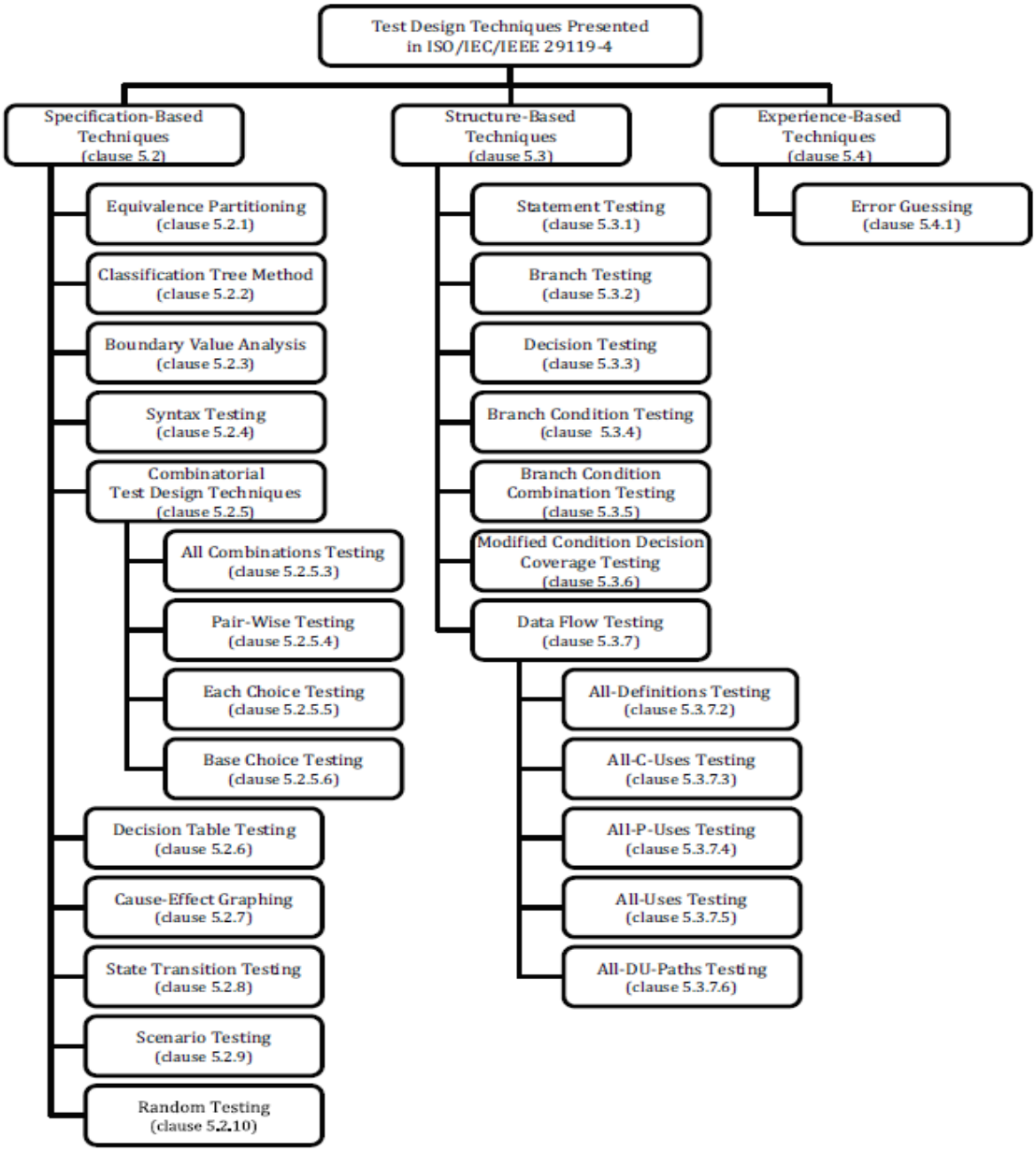


Figure 2.10: Test Design Techniques [Tec15]

2.4.3 Testing Phases and Test Types

Another important aspect of test design is strategizing the test process. The quality parameters like safety, reliability, functional suitability, usability, security, and so on [ISO10] are validated and verified by several test types, namely, performance testing, functional testing, usability testing, etc [CD13]. The test sub-process is conducting these testing types at different test levels/phases of the software development life cycle namely, integration testing, system testing, acceptance testing, and

so on [CD13]. Hence the entire testing process is a combination of such sub-processes [CD13]. The association of the testing types, testing levels, quality parameters to the test process is illustrated in Figure 2.11 [CD13].

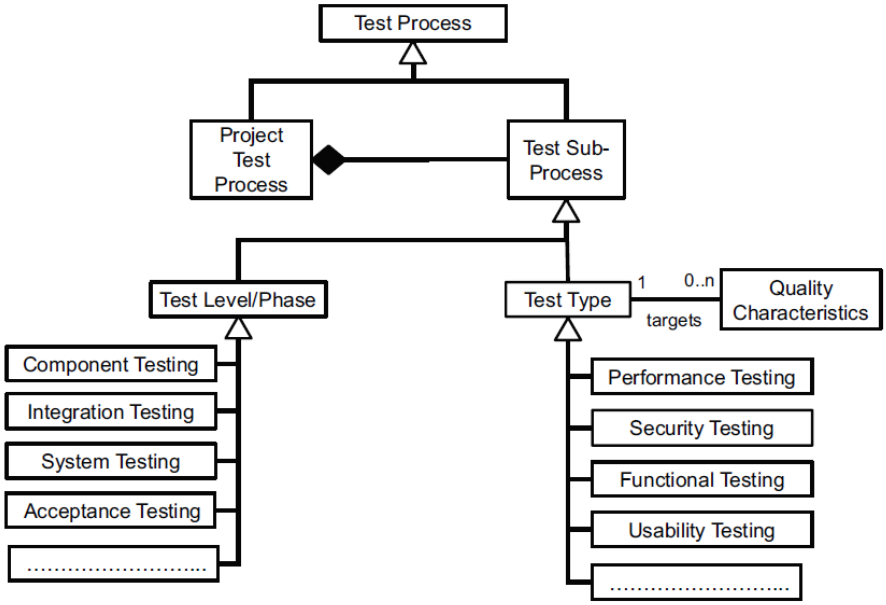


Figure 2.11: Test Phases and Types [CD13]

2.4.4 Testing Methods

There are three ways in which testing is performed. First is black-box testing which the tester supplies the input and validates against the requirement specifications without knowing the intrinsic details [Lim09]. The method by which the internal code structures and design specifications are tested white-box testing method [Lim09]. This method assures that software is developed in a right manner [Lim09]. The last type, gray-box testing is the combination of both the methods. The test cases are designed like the white box method by considering the design and requirements. Whereas the testing is performed in a black-box manner by providing the inputs and confirming that the application is good relation with the defined requirements [Lim09]. Figure 2.12 provides the diagrammatic representation of these three methods [Lim09].

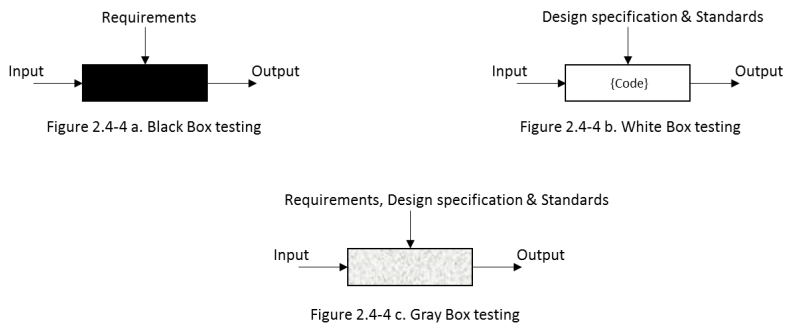


Figure 2.12: Test Design Techniques [Lim09]

2.5 Summary

Insights of the data science process, the types of ML, the processes and the phases of software testing along with an introduction to Kaggle were gained through this chapter. The thesis brings together the research domains namely, software testing, data science, and ML. Explanation about the various aspects used in these domains is provided in this chapter. This would help the audience to understand some of the terminologies associated with any of these domains. The major focus is to simulate software testing to test the ML model produced as an output of the modeling phase of the data science process. Therefore, assuring the quality of the ML related software. A section about the online forum of Kaggle was also included in this chapter, to provide an overview of the data and codes relevant for testing the concept developed as a part of this research.

3 Related Work

Attempting to combine the two diverse domains, namely, software testing, ML is a complex task. There is no correct approach to achieve such a task. Hence, several approaches are followed to integrate one domain with the other. In this chapter, we will discuss the various approaches followed by researchers in applying software testing to the domains of ML or data science.

Murphy, C et.al have discussed a software testing approach for ML application by considering two ML algorithms: SVM and MartiRank [MKA06]. The same authors in another paper focused on software QA of ML applications by examining the dependability attribute of the ML model which was developed based on ranking algorithms [MKA07]. They described a framework for testing and debugging ML applications that implemented ranking algorithms. Their framework consists of a test data set generator, tools that present the rankings by comparing several ML models, the inclusion of various trace options to the ML implementations, and services helping to analyze the traces. As a part of the evaluation of their framework, the authors utilized three different implementations. One implementation was written in Perl and the other two were written in C language. All the three implementations were on the ML algorithm developed to improve the ranking problem they had considered in their work. The use case considered by them was to prioritize the recovery mechanism for electrical distribution feeders that were susceptible to failures.

In the testing approach [MKA07], the authors compared whether all the three implementations yielded consistent results. They performed regression testing to check the behavior of a given implementation of the algorithm on any enhancements relative to the previous implementation . A utility was created by the authors to compare the models and report on the differences in each round. Higher importance given by them was only to the difference between the model rankings on the first round of comparison because any enhancements done to one of the implementations were found to alter the algorithm for the other implementations as well [MKA07]. The authors compared the metrics like quality (Area Under the Curve (AUC)[1]) for each ranking, the number of differences between the rankings (elements ranked differently), the Manhattan distance (sum of the absolute values of the differences in the rankings), and the Euclidean distance (in N-dimensional space). To check the similarity of the rankings, another metric used by the authors was normalized Spearman Footrule Distance (1 meant that the rankings are exactly the same, 0 meant that the rankings are completely in the opposite order). The authors found that the testing framework proposed by them helped to create, execute and analyze the test cases necessary to assure quality in their use case. Another interesting conclusion inferred was that their framework could be extended to other ML algorithms on carefully selecting the testing approach in case of a single implementation [MKA07]. The authors have emphasized the need for QA to ML applications.

[1]AUC: https://www.sciencedirect.com/science/article/abs/pii/S0031320396001422?via%3Dihub

On similar lines, Wickramage, N emphasized the need for QA for data science by suggesting the use of simulations [Wic16]. The above papers helped to focus on combining the domains of software testing and ML.

Breck, E et.al presented rules to award a score based on a set of tests that help in determining the quality of the ML system [BCN+16]. They considered testing the data, the features (realized from the data), the model development, the model quality, and the entire pipeline of the ML system. The focus of their research work was on defining the development of a quality ML system. Minor importance was given by them to the generic best practices of software engineering like the inclusion of good test coverage and a defined release process. The authors included some tests as a part of the rubric. These were testing the data quality, the correlation between each feature and the target, the pairwise relationship between each of the predictors, checking the model evaluation metrics, determining the model decay with respect to time, unit testing of the model, and comparing the model predictions with the true value of the target variable. These tests are considered in our thesis and are implemented at various phases of the model testing.

The same authors extended their earlier paper by presenting a set of 28 specific tests and monitoring needs based on the experience with a plethora of production ML systems in another scientific paper [BCN+17]. Their objective was to assist the audience in measuring the issues concerning the development of a reliable and production level ML system by presenting a scoring method. The authors suggested ways to prepare the ML systems for production-level functioning and decrease the costs incurred due to incorrect results provided by the ML system. Their work was on the supervised learning type of ML.

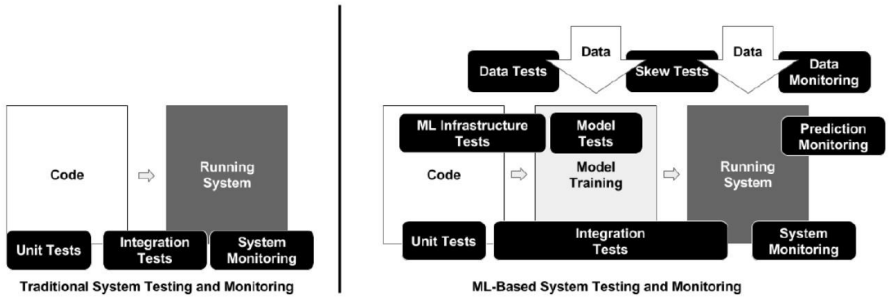


Figure 3.1: Testing and monitoring required for traditional and ML systems [BCN+17]

Figure 3.1 shows the authors' consideration of how system testing is carried out for a normal software application and the ML software application. Based on which they insisted that the ML software applications require a broad range of software testing. Similar to the suggestions made in their previous work, the broad categorization of tests indicated by them was testing the data and the features (realized from the data), the model development, the model quality, and the infrastructure of the ML system. Along with these, they proposed to monitor the ML system as an additional category of tests [BCN+17]. The additional test category devised by them monitors whether the ML system's performance is consistent over time and capable to maintain the quality [BCN+17]. They suggested to continuously train the model on the new data in the production environment and check the model performance. The authors also suggested having a dashboard to continuously view the results of monitoring in the form of relevant graphs and metrics [BCN+17]. They stated the necessity to alert the relevant team whenever the performance is seen to deteriorate significantly from the expected value.

Breck, E et.al also insisted on the difficulty to devise specific tests, since it is hard to realize the actual prediction behavior of any given model beforehand [BCN+17]. Less importance was given towards the best practices of software engineering like in their previous work [BCN+16]. We can relate the test categories defined for the ML system by the authors in [BCN+17] to the software testing phases defined by the ISO/IEC/IEEE 29119 series [CD13] as follows:

- Test for model development – similar to the unit testing
- Test for ML infrastructure – similar to the integration and to some extent of system test
- Monitoring Tests for ML – similar to user acceptance test and maintenance phase

For this thesis, we focus on considering the categories of test for features and data, test for model development and test for ML infrastructure as the different software testing levels for a customer analytics use case.

Nishi, Y et.al also showed a QA framework to evaluate ML products [NMOU18]. They had done it in two parts. In the first part, the authors proposed a QA policy for the ML products by introducing the principles of evaluation like Allowability, Achievability, Robustness, Avoidability and Improvability and a strategy to evaluate the ML product. As a second part, a test architecture consisting of test levels and fundamental test types for testing the ML product was proposed. Based on these two parts, the authors established the activity levels for QA. They structured the ML product into seven types, namely, World, System, Software, ML frame (ML algorithm), ML model, training data and data groups discriminated [NMOU18]. Next, they defined the principles of ML product evaluation which are Allowability, Achievability, Robustness, Avoidability, and Improvability and constructed a strategy to consider the permissible levels of the defined principles while evaluating the ML product. As a part of test architecture, the seven structures of the ML product served as the basis to categorize the software test levels and test types defined by the ISO/IEC/IEEE 29119 series relevant to the domain of ML. Later, they also used different ways to conduct testing of the ML component like snapshot testing, learning testing, and confrontation testing. Finally, the authors outlined the different levels of QA to be performed for the ML product as QA activity levels. Through their research paper, it was revealed that there are software testing levels, techniques, and types that can be employed to evaluate the quality of the ML model [NMOU18].

Nakajima [Nak18] explained the necessity to test the ML program and presented three views. Two views were the service and the product quality of the ML program whereas the third view was on testing the entire ML platform (pipeline) which was termed as an alternate view by him. The author further added that when the emphasis is given on the service aspect then the systems with better accuracy are considered even if the systems contain faults or errors. Whereas, the author also stated that product quality could be achieved by focusing on methods to remove faults from program implementations. The metamorphic testing type was suggested by him to achieve product quality. In the alternate view of the testing entire platform, validating data for outliers and other inconsistencies was suggested to be taken care of while implementing the ML program. Suggestions for considering neural networks, careful selection of hyperparameters, trying out the agile software process, the inclusion of a DevOps stage, and re-learning with a revised dataset were proposed by the author. Thereby, he justified the necessity of software testing for the ML program (or the model). An interesting point revealed by this paper was to consider both the service and the product quality aspects while designing the testing process [Nak18]. This provides an important point in our thesis as to not simply consider the ML models with high accuracy as the better one.

Braiek et.al [BK18] first analyzed the challenges faced while testing ML programs and described the existing solutions to test ML programs that were explained in the literature they referred to. The authors have also recommended possible research directions related to the testing of ML programs. The authors stated that the data and the model were the sources of errors in ML programs. This aspect serves as a basis to have two test automation scripts of dataset_evaluation_test and model_quality_test in our thesis.

To summarize, Murphy, C et.al [MKA07] carried out testing by comparing the consistency of the results yielded by three ML implementations. They conducted regression testing to compare how any enhancements to the current implementation of a ranking algorithm (considered by them) behaved with respect to its previous implementation. Wickramage, N [Wic16] suggested QA for data science to be conducted through simulations. Breck, E et.al presented a set of 28 tests and rules to award a score based on a set of tests that help in determining whether the ML system can be deployed to the production environment [BCN+17] [BCN+16]. The tests indicated by them was testing the data and the features realized from the data, the development and the quality of the model, the testing the infrastructure of the ML system, and continuously monitor the ML system. Braiek et.al [BK18] described more solutions to test ML programs. Nakajima [Nak18] considered testing the service quality, product quality, and the entire ML pipeline. Whereas, Nishi, Y et.al [NMOU18] proposed a QA policy for principles like Allowability, Achievability, Robustness, Avoidability, and Improvability for the ML product. Next, they defined software test levels and test types based on the previously mentioned principles.

We find that all the publications considered in this chapter emphasized the need for QA for ML. But, each of them has suggested different ways of performing testing. The authors did not provide information about the software test design techniques, test phases, test types and test methods required to perform the QA process for the ML model like it is executed for a software application. Hence, we would consider testing the data and the model for errors as suggested by Breck et. al in [BCN+17] and Braiek et.al in [BK18]. Furthermore, we would go beyond these works by extending the software test design techniques, phases, types, and methods to spot ML defects. Thereby, assuring the quality of the ML model.

4 Scope

This chapter starts with an illustration of a data science process applied to the below explained customer analytics use case. Next, the assumptions considered in the thesis are outlined to explain the necessary aspects included in the research. Hence, the scope of the thesis is unveiled for the further chapters to come.

4.1 Customer Analytics Use case

Let us consider an example of a person having the contract of a car for 2 years. Now, a customer analytics use case would be that person's preference to renew the car's contract at the end of the contract term [1]. Another example would be that there is a new product in the market and predicting the likelihood of customer buying that product [2]. These are some of the customer analytics use cases.

Figure 4.1 shows the data science process carried out in a customer analytics use case. The process consists of two database components, three phases, and a logical entity. This is similar to the data science process discussed in Section 2.1.1 and can be explained based on [OS13]. The data owner (business units/subsidiaries/market) represents the real-world entity as shown in Figure 2.1. The data owner stores the customer data to a data source. Later, any sensitive customer-related information is replaced with anonymous values to bind with data protection laws. Furthermore, the anonymized data is loaded into a local data source which can be utilized by the data scientists to carry out the data science process and gain insights over this data.

Pre-processing, feature engineering, and model selection are the phases present in the customer analytics data science process. Firstly during the pre-processing phase, the data scientists detect the missing values in the data, clean the irregularities and null values in the data, remove duplicate rows and columns, and analyze the distribution of each column to envision the possible features. It aims at transforming the dataset to be fit for further phases. Secondly, in the feature engineering phase, the attributes or columns in the dataset that help in prediction are selected and their correlation with the target variable is verified. These attributes are referred to as features [OS13]. Generally, the attributes with low correlation value are considered as features [OS13]. Also, new features can be extracted by relating two or more attributes mathematically and the correlation value of such newly derived features is examined to build a model. Lastly, in the modeling phase, three functions are performed. The first function is to select the best optimal algorithm based on the application use case and the features extracted in the previous phase. The algorithms can be from the field of

[1] Customer Churn: https://www.kdnuggets.com/2019/05/churn-prediction-machine-learning.html

[2] Analytics example: https://www.forbes.com/sites/blakemorgan/2018/12/20/10-examples-of-predictive-customer-experience-outcomes-powered-by-ai/3d4949375d0b

statistics, ML or deep learning. The second function is to train the chosen algorithm with training data so that the algorithm learns the pattern required for prediction. In the end, the trained algorithm is validated with a testing dataset to check whether the algorithm has learned well or not. Finally, a logical entity termed as a "Model" is obtained as a result of the modeling stage.

Results are communicated through these web services. A model API service is designed to publish the results of the model. While the market team maintains a market API service to fetch the prediction results from the model API. Request to the market API contains the customer information like the anonymized ID of the customer and in response, a score is returned. Similarly, a request is made to the model API from the market API and the model API returns the score based on the results predicted by the model. The score obtained in the current cycle will be stored again in the data source to be used in the further cycles for the re-training of the model. The score published by the API services is the probability of prediction of the customer's preference for a particular product.

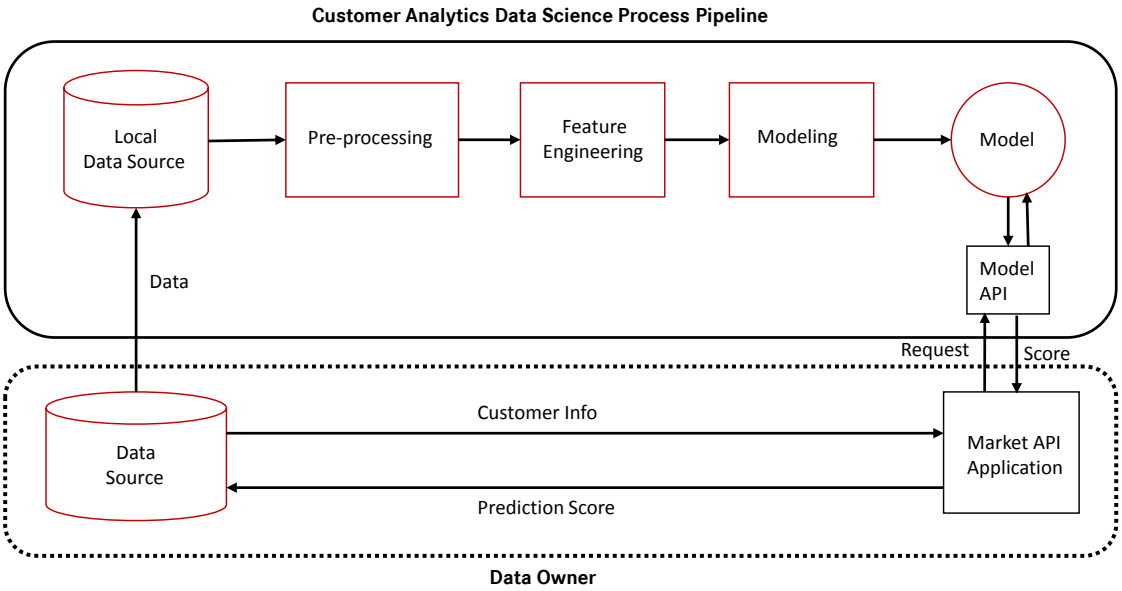


Figure 4.1: Customer Analytics Data Science Process Pipeline

4.2 Assumptions

The central focus of this master thesis is to assure the quality of the ML model by investigating the output of the model on applying the data. This is similar to the black-box testing method [Lim09] in software engineering. Hence we limit the scope to the black-box method of testing.

Below are some more remarks that relate to the specifics of the scope.

- Usage of the terms testing and validation

For this master thesis, the to term "testing" is used to mean a set of activities performed to confirm the correct behavior of the software product, as defined in Section 2.4 on page 30. Thus, the term testing is not limited to only a process by which model behavior is assured. The latter meaning often finds usage in the data science community.

For this master thesis, the term “validation”, is used to denote a set of actions conducted aiming to check whether the software product produces expected results [CD13]. Whereas in ML, the process of cross-validation (in short, validation), means a validation test and/or adjustment of model hyperparameters [Gér17]. Also, the terms QA and testing are used interchangeably in this master thesis.

- Focus on binary classification is the chosen ML type

  The master thesis focuses on binary classification. This is since the binary classification is vastly used for customer analytics use case and it is a simple type of ML where the output is either yes or no [Gér17]. Although multiclass predictions and time series analysis also find usage in customer analytics use cases, they are outside of the scope of this work.

- Selection of ML related defects

  The subset of ML related defects considered in the master thesis was selected to accommodate the major needs of the data science models that were identified in different procedures (e.g. literature survey, interviews with data scientist discussions, etc.).

- Metamorphic testing and A/B testing

  The main focus of the master thesis is to test the model as black box. In Metamorphic testing specific behaviors of the functionality to be tested is found out and a metamorphic relation is formed based on the specific behaviors [CCY98]. Besides, the functionality under test and metamorphic relation are applied with a small amount of data to check whether both functionality and relation produce the same result [BK18]. This is similar to white-box testing which is out of focus for the thesis.

  A/B testing in Software engineering is all about having two different features implemented for the same software. In the ML domain, A/B testing is analogous to verify the performance of two or more models for the same problem [Zhe15]. Generally, techniques like cross-validation [OS13] [Gér17] is used to select the right model for a given ML use case. Implicitly A/B testing is carried out during the model selection stage.

- Usage of the terms model and ML model

  In this master thesis, we refer the term model to the ML model. If we have to refer to any other model, then we would explicitly mention it in our document. For example, we explicitly mention ’UML model’ in case of providing information on it, in the thesis.

4.3 Summary

Section 4.1 and Section 4.2 explained the objectives and the considerations of this research. Based on the scope defined in this chapter, we carry out the research and design the tests to identify the defects specific to the domain of ML or data science.

5 Results

In this chapter, details of the approaches followed for answering each of the research questions and the results obtained on following these approaches are explained.

5.1 Research Question 1 (RQ1)

In software engineering, "Error" can be a deviation between an observed or actual outcome and a true or expected outcome induced due to the software developer's activity [GVE08] [ISO10]. Similarly, a "Defect" can be an imperfection in a system preventing it from accomplishing its necessary function [Ter90] [GVE08].

RQ1 deals with collecting information about major defects or errors faced by the data scientists in their daily activities. A combination of two research fields, namely, ML and software testing provides a broad scope to be considered in concluding this research question. We followed methods like expert interviews and literature review to gather the defects affecting the quality of the data and the model. Additionally, we verified whether the errors elicited from the previous two methods were discussed in the Kaggle forum as well, via a program.

5.1.1 Expert Interviews

Expert interviews were conducted in a semi-structured format and contained a combination of an open and close type of questions. The interviewees were interviewed personally, while their responses were documented in an interview form. The interview revolved around a set of three questions on defects or errors specific to the field of ML. It also involved one follow-up question. Table 5.1 on the next page provides the intent of each question along with the questions asked.

Intent of Questions	Questions
Explores experience of the experts with all the problems or errors or defects, they encountered during their daily activities.	What are the problems/defects/errors associated with ML or data science process experienced by you?
Extract information about defects which can be detected from the data.	Are there problems/defects/errors that can be revealed from the data? If yes, please mention those defects?
Gather information on the defects that decrease the model's ability to provide the likelihood of possible outcomes.	Are there problems that specifically lower the performance of the model (model decaying) and cannot be revealed by the dataset? If yes, please mention those defects.
Follow-up question to obtain additional information helpful for the research.	Additionally, please let me know if you have any other comments regarding any other defects or problems (e.g. suggestions on any scientific or review paper that you found extremely useful)

Table 5.1: Interview Questions for RQ1.

Seven data scientists - one team lead, three senior data scientists and three data scientists - were interviewed to gather insights about the defects or errors concerned with the field of ML or data science. The interviewees were asked some general questions regarding their work. It was followed by brainstorming them on the goals of the thesis and the relevance of the interview in the research. Afterward, they were asked about their experiences and opinions on various defects or errors which were relevant to the field of ML, they have come across in their daily activities. Answers about the source of those defects, were also gathered. Follow-up questions requesting for additional information extended the interviews. Finally, the interview concluded with a discussion on the impact of the quality of model predictions on business. The demographic data of the interviewees is depicted in the Table 5.2.

Attribute	Attribute Value	Number of experts
Type of organization	Services	7
Qualification of the interviewees	Doctorates (PhD)	4
	Master's degree	2
	Bachelor's degree	1
Age of Data Scientist (in years)	20-30	2
	30-40	2
	40-50	3

Table 5.2: Demographics data of the interviewees for RQ1.

The results of the interviews along with the graphs are explained as follows.

Question 1: What are the problems/ defects/ errors associated with ML or data science process experienced by you?

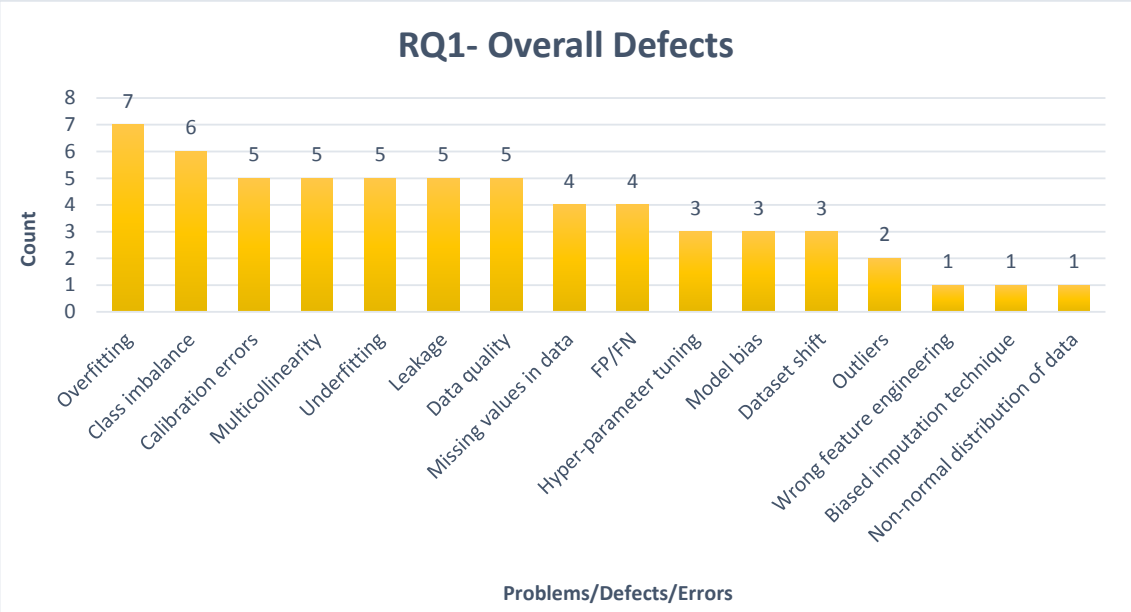


Figure 5.1: Response from experts on overall defects

Responses from Experts

The response to this question is illustrated by Figure 5.1. Most of the experts stated that overfitting and class imbalance are the two common defects encountered by them. These are followed by defects like calibration errors, multicollinearity, underfitting, leakage, and data quality that takes the next place according to the responses of the data scientist. Problems like missing values in data, false positives/false negatives, dataset shift, hyperparameter tuning, and model bias received moderate responses. Two data scientists discussed about the outliers problem in the interview. Wrong feature engineering, biased imputation technique, etc. were the responses with lower count as seen from the graph. This is indicative of two possibilities that either such defects are not commonly encountered by them or most of the interviewed experts already handle such defects effectively that they haven't felt the need to list those.

Question 2: Are there problems/defects/errors that can be revealed from the data? If yes, please mention those defects?

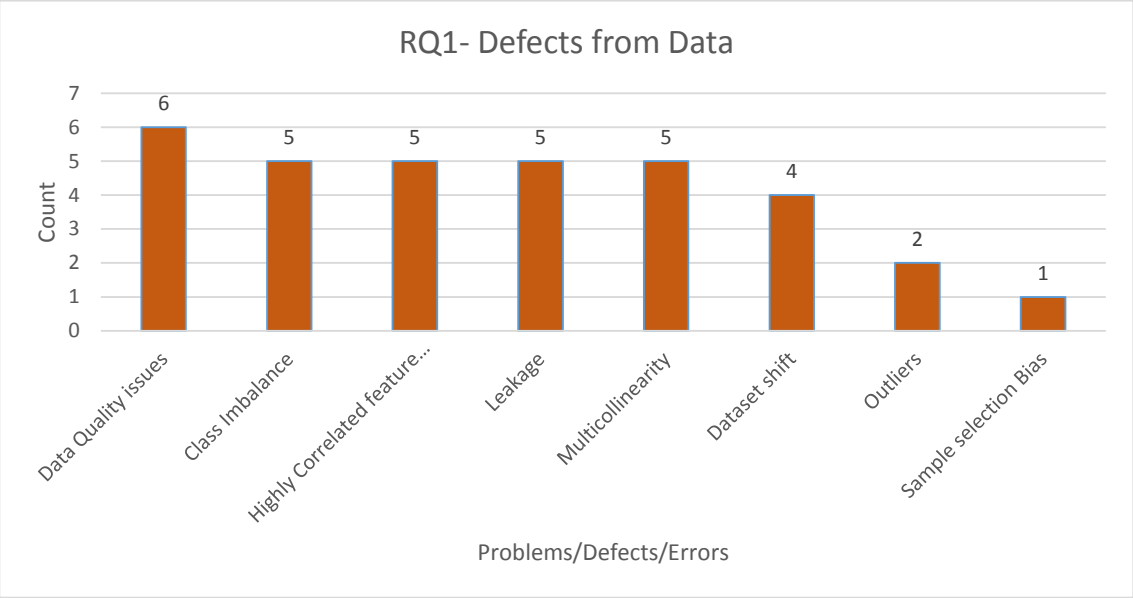


Figure 5.2: Response from experts on defects detected from data.

Responses from Experts

85% of the experts gave their opinion that data quality issues need to be considered. They considered checking the quality of the data is important to gain useful insights upfront from the data and essential for generating relevant features required for generalizing the patterns. Most of the experts believed that problems like class imbalance, leakage, highly correlated feature variables, and multicollinearity also arose from the specifics of data. Surprisingly, the problem of high feature correlation was being included as an answer to this question but the same response was not provided to the earlier question. When this was asked, the data scientists regarded that highly correlated features influence the model to provide erroneous results and depend on the skills of the data scientist in visualizing the data. The next commonly given response was the problem of dataset shift. Few responses considered problems like outliers and sample selection bias also arise out of the data. The reason for outliers getting fewer counts is that it is already treated in one developed package used by the interviewed data scientists. As for the sample selection, it is also left to be handled by the data scientist.

Question 3: Are there problems that specifically lower the performance of the model (Model decaying) and cannot be revealed by the dataset? If yes, please mention those defects.

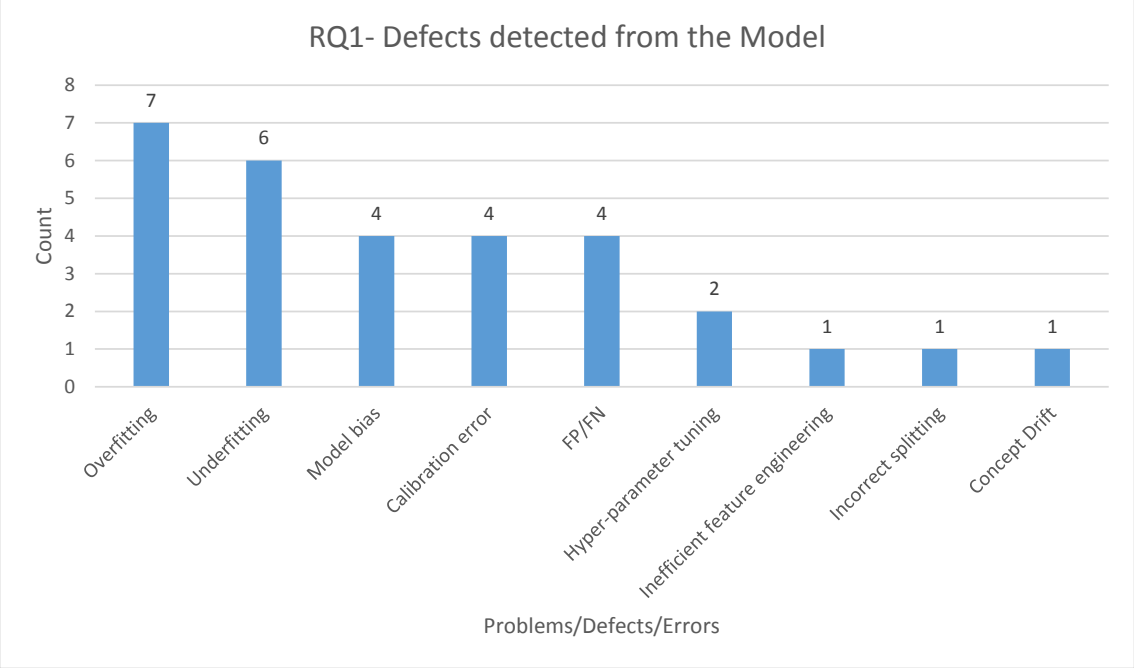


Figure 5.3: Response from experts on defects detected by testing the model.

Responses from Experts

Similar to the first interview question, most of the experts expressed that overfitting and underfitting are the common defects that are checked for. Followed by the problems of calibration errors, model bias, and false positives and false negatives. Few data scientists outlined the problems of inefficient feature engineering, hyperparameter tuning, and incorrect splitting strategy for the model. But, they suggested that these problems are generally due to the experience of data scientists. One data scientist stated that concept drift is one of the most critical defect.

For the follow-up question, few data scientists answered that even though the possibility of defects like leakage, dataset shift, and class imbalance can be detected from the data. The presence of such defects can also be verified by testing the model. Apart from that, one data scientist suggested checking the computation time required to train the model.

5.1.2 Literature Review

In this sub-section, books, scientific papers, and journals were examined to determine the possible errors or defects or problems that degrade the quality of the ML model.

Braiek et al. [BK18] outlined that errors would be present in the data that would be used for training and testing a ML model and also in the model. This explains the need to conduct the QA process that ensures developing trustworthy ML models. The common problems found in data, highlighted by them, were the presence of invalid or undefined values like Not Available (NA) or Not a Number (NaN), duplicate rows, outliers, variable represented in different formats, as well as noisy data, and changes in the data distribution. They also highlighted overfitting as a defect of a ML model.

More information on the defects related to the domain of ML was explained in detail by Géron. A [Gér17]. The problems discussed in the book could be classified as those which are present in the data and those that can be detected after the model is trained. Problems in connection with the dataset size, noisy data, sampling bias, and problems related to feature engineering would be present in the data. Whereas the problems of overfitting and underfitting the model can be detected after the model is trained and during the testing of the model.

The paper on the issues and challenges of data mining by Sharma, B.R et al. [SKM13] outlined a list of problems in the field of data mining. Major issues listed were outliers, overfitting, bad quality of data, size of the dataset, and the shift in the distribution of the data. The same issues apply to the data science domain as well. Since data mining and data science involves the same techniques to be performed [LRU14].

Most of the other literature referred also covers the previously mentioned defects in general. Some of the other distinctive defects mentioned in the literature were the usage of RAM mentioned by Breck, E et al. [BCN+17], imbalanced class mentioned by Zheng, A in [Zhe15], leakage problems and the necessity of calibrating the results are mentioned by Schutt, R et.al in [OS13].

5.1.3 Programmatic Approach

From the results of the expert interviews and literature survey, we found that overfitting, underfitting, class imbalance, missing values in the data, calibration, leakage, and outliers were commonly discussed defects. The expert interviews were conducted with seven data scientists. And, the publications we referred to, mentioned the same defects. As a next step, we wanted to confirm whether such defects are discussed by a global community of practitioners in the domain of ML or data science. The programmatic approach explained in this subsection helped us to achieve this.

At first, we extract the data of the discussions happened on an online forum, Kaggle, in the form of text. Then, unwanted data is removed from the text. Finally, we check the presence of the keywords such as overfit, underfit, imbalance, missing values, calibration, leakage, and outlier, in the text data of the discussions happened on an online forum. These words represent the lemmatized[1] form of the words: overfitting, underfitting, class imbalance, missing values in the data, calibration, leakage, and outliers. This programmatic approach is based on NLP[2].

The implementation of this approach is shown in Figure 5.4 on the next page. The implementation consists of three parts, namely, data extraction, data filtering, and frequency of keyword occurrence.

5.1.3.1 Data Extraction

The data required to create a text corpus to check the frequency of occurrence of common defects or errors in data science or ML is extracted from Kaggle discussions threads. The Kaggle Public API [3] is used to get all discussions happening on Kaggle's forums, datasets, and competitions using a discussion id. A script was written in the Python programming language to send requests to Kaggle

[1]Lemmatization: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html
[2]NLP: http://www.cs.bham.ac.uk/ pjh/sem1a5/pt1/pt1_history.html
[3]kaggle json: https://www.kaggle.com/topics/9565.json

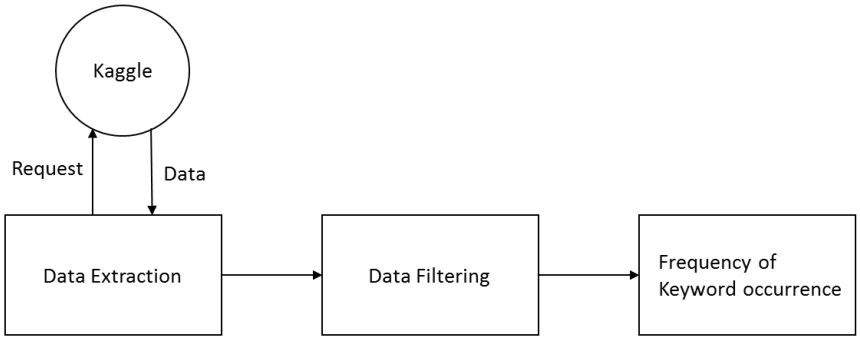


Figure 5.4: Implementation of the programmatic approach

and acquire data via the public API. The script is provided in the Appendix A.1. Listing 5.1 1 shows how to use the API to get a discussion thread using the discussion id. In this example, 9565 represents the id of a discussion thread. The RQ1 sample dataset consists of id's in the range from 1 to 75000 as of 03 May 2019.

Listing 5.1 GET request to retrieve information about a discussion using discussion id

```
GET /topics/9565.json HTTP/1.1
Host: www.kaggle.com
Accept: application/json;charset=UTF-8

HTTP/1.1 200 OK
{
        "id": 9565,
        "parentName": "Titanic: Machine Learning from Disaster",
        "comment": {
                "author": {
                        "displayName":"ifguy12",
                        ....
                },
                "content": "<p>Suppose we have ...",
                "postDate": "2014-06-25T16:46:15",
                "votes": 0
        },
        "commentList": {
                "comments": [
                        {
                                "author": {
                                        "displayName":"John Uckele",
                                        ...
                                },
                                "content": "It's worth nothing...",
                        },....
                ]
        }
}
```

5.1.3.2 Data Filtering

The data extracted from the previous phase is filtered out for specific keys and its corresponding values in the JSON, to create a new JSON structure for the text corpus by using another python script named data_filtering.py provided in the Appendix A.2. The new JSON structure is shown in Listing 5.2.

Listing 5.2 JSON structure for a discussion thread

```
{
        "id":integer,
        "competition": string,
        "author": string,
        "date_time": string,
        "overview": string,
        "comments": string,
        "votes": integer
}
```

The JSON structure has the following keys:

- **id**: discussion thread id
- **competition**: name of the competition where the discussion has raised
- **author**: name of the person who started discussion thread based on Kaggle's forms, competitions or datasets
- **date_time**: the date and time when the discussion thread was started
- **overview**: the content of the discussion thread
- **comments**: all the comments and replies received for that discussion thread
- **votes**: the upvotes received for the discussion thread

The values from the Kaggle public API when filtered using the data structure from Listing 5.2 can be illustrated below using the Listing 5.3.

Listing 5.3 Example JSON structure for discussion thread with id: 9565

```
{
        "id":9565,
        "competition": "Titanic: Machine Learning from Disaster",
        "author": "ifguy12",
        "date_time": "2014-06-25T16:46:15",
        "overview": "<p>Suppose we have ...",
        "comments": "It's worth nothing...",
        "votes": 0
}
```

5.1.3.3 Frequency of Keyword Occurrence

The final part of the programmatic approach is to search for the occurrence of the keywords in the text corpus. These keywords refer to the defects or errors in the ML domain and were collected from the expert interviews and literature review. We used PySpark [4] and Microsoft Azure [5] platform.

The data filtered in Section 5.1.3.2 is loaded onto the cloud storage of Azure. Appendix A.3.1 provides the code for this step. The data in the JSON structure is presented in the relational schema by Azure. Next, all the keys of the filtered data are mapped as the columns in the tabular format as shown in Figure 5.5.

author	comments	competition	date_time	id	overview	votes
ifguy12	It's worth noting that the Parch and SibSp features might lump together useful features. If you wanted to extract separate Sib and Sp features from the data, you could look at the names and find married couples. Likewise, by checking last names and ages, Parch could be teased apart. Correct, not all of the passengers are listed in either the train or test data for this competition. You can find the full data here: http://www.columbia.edu/~cjd11/charles_dimaggio/DIRE/resources/R/titanic.csv I do wonder where you're finding a married Davies pair on this boat though, the only Davies I see with (Sibsp, Parch) = (1, 1) is Master John Morgan Jr., and he looks like he's 8 years old. It was just an random example and a coincidence that the names were the same.	Titanic: Machine Learning from Disaster	2014-06-25T16:46:15.43Z	9565	Suppose we have the following two passengers: SibSp Parch Mr. Davies 1 1 Mrs. Davies 1 1 Mr. Davies has 1 spouse. But he has no parents and children. Yet Parch=1 for him. The same also is true for Mrs. Davies. So maybe the parents or children of these passengers are not listed?	0

Figure 5.5: Mapping of JSON structure to relational structure

The text data contained in the columns of overview and comments are merged to form a single column composing the entire text data of a single discussion thread. Appendix A.3.2 provides the code for this step. The text data contains punctuation marks, HTML tags, and words that are not in a uniform case. The necessity of refinement defines us to remove all the unwanted elements mentioned previously from the text data. All the words are changed to lower case to ensure uniformity. Hence, the data is refined to perform natural language processing. The code to achieve this is provided in Appendix A.3.3.

Breaking down of the sentences in the refined text data into a list of words, or tokens, is essential to analyze the number of discussions which consist of keywords specific to defects or problems related to the domain of data science. While executing the tokenization[6] task itself, a regular expression with the keywords of defects or errors is applied, so that all the words are strained out of the text corpus. This results in a list consisting of only the defect keywords.

[4]PySpark : https://spark.apache.org/docs/latest/api/python/index.html
[5]Microsoft Azure : https://azure.microsoft.com/en-us/
[6]Tokenization:https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html

A new column is appended based on the size of the list for each row. A value of '0' refers to the discussions without the keywords about the errors. A value other than '0' refers to the discussions with the keywords about the errors. Next, the number of rows with the value other than '0' are summed up. They represent number of discussions about the errors related to the domain of ML done on an online community. Likewise, the vice-versa is done to infer the number of discussions not about the defects or errors related to the domain of ML. This is achieved by the code seen in Appendix A.3.5. In addition to this inference, the percentage distribution of each error keyword can be interpreted.

The format of the query input containing the list of keywords as seen from Appendix A.3.4 was:

"overfit | underfit | missing values | imbalance | covariate shift | outlier | leakage | calibration | dataset shift | drift"

The symbol '|' denotes bitwise OR operator[7]. And the words represent the defects of overfitting, underfitting, missing values in the data, class imbalance, outliers, leakage, calibration errors and dataset shift.

From the graph Figure 5.6 it can be seen that out of 75000 discussions collected from Kaggle as of 03 May 2019, around 7500 discussions contained at least one of the keywords. And around 68000 discussions did not contain any of such keywords.

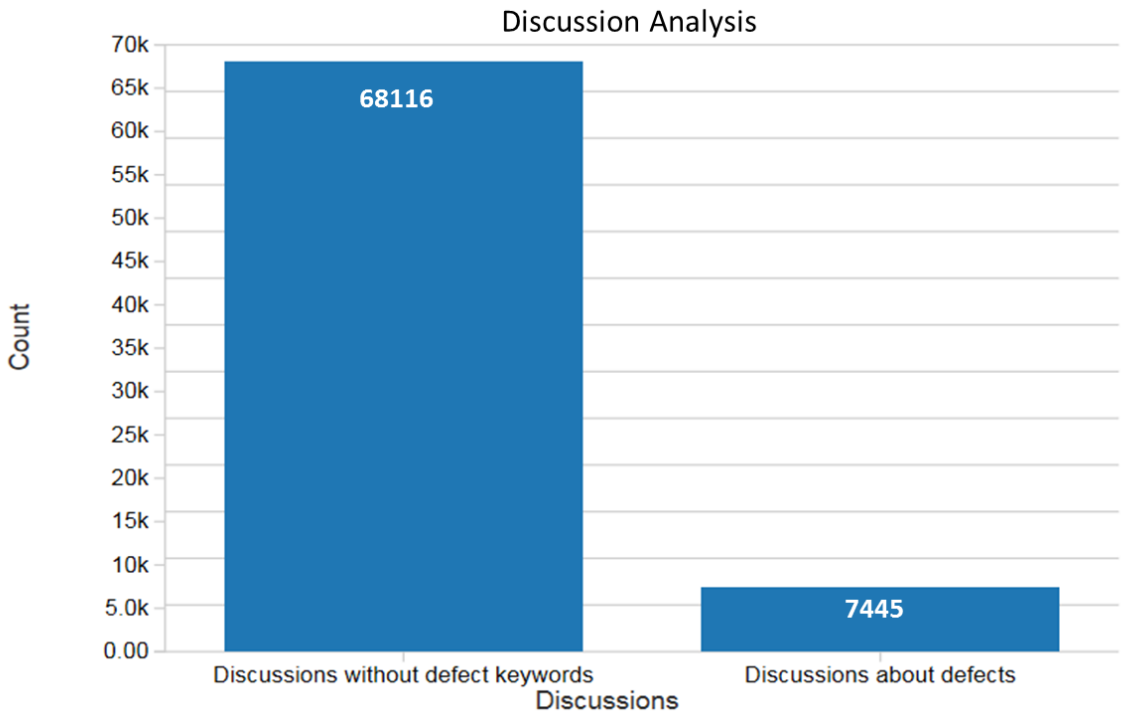


Figure 5.6: Distribution of discussions on Kaggle

As a part of further analysis, the distribution of each of the defect keyword is considered to find out the common defects among the selected list of keywords as seen from the Figure 5.7.

[7]Bitwise operator: https://docs.python.org/2.0/ref/bitwise.html

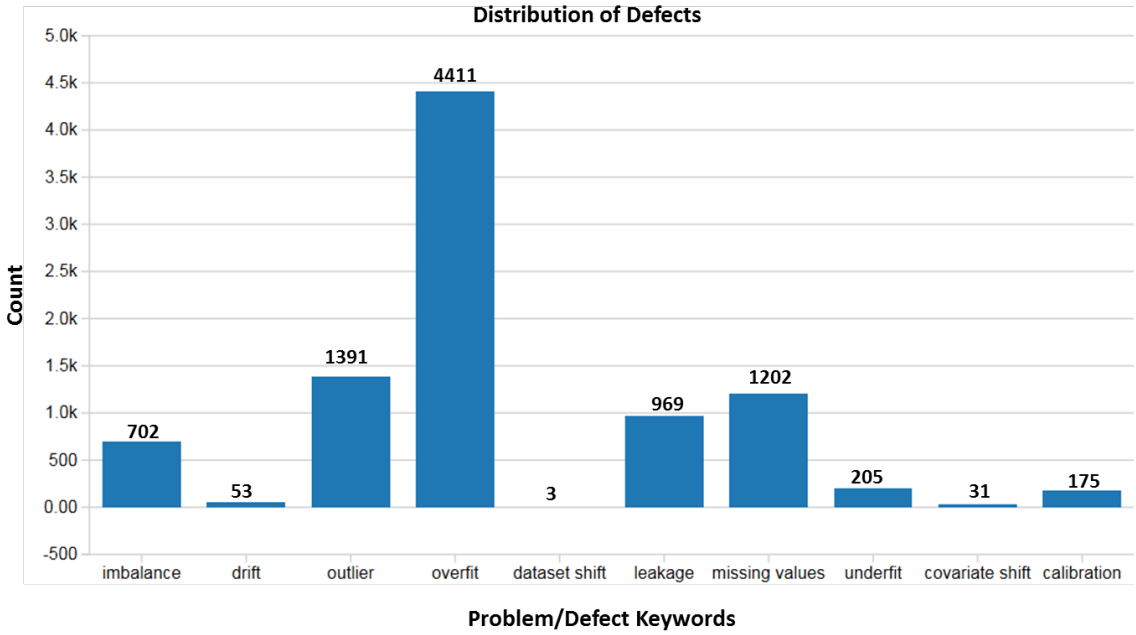


Figure 5.7: Distribution of ML related defects

The keywords in Figure 5.7 can be regarded as lemmatized (i.e. with removed inflectional endings). The words with inflectional endings were also queried separately, to check how well our NLP program found the given keywords. We expect that the number of each of the found words in the lemmatized case should not exceed the number of the corresponding word in the non-lemmatized case. So, we used the keywords such as overfitting, outliers, and underfitting for querying instead of the keywords overfit, outlier and underfit. And, we ran the query once again. Indeed, we observed from Figure 5.8 that the count of the discussions with the defect keywords decreased and the count of the discussions without the defect keywords increased as compared to that of Figure 5.6.

Similarly, from Figure 5.9 we observe that there is a reduction in the count for the defects keywords "overfitting", "underfitting", and "outliers" as compared to that of the counts seen in the graph Figure 5.9, because we have used the non-lemmatized form of the keywords for the query.

Even though the count of the defect keywords decreased when modified individually, the overall trend remains the same. From the defect keyword list considered, we observe that the keyword overfit is seen to be the most common, inferring that the defect overfitting is most widely discussed. Following the overfitting are the outliers, missing values, leakage, class imbalance, underfitting, and calibration errors. The keywords drift, covariate shift and dataset shift all mean the same in the context and are observed to be the least discussed topic among the list of defects considered for the query. We can also note that the analysis made from the programmatic approach is based on the keyword list considered for querying and it was not exhaustive.

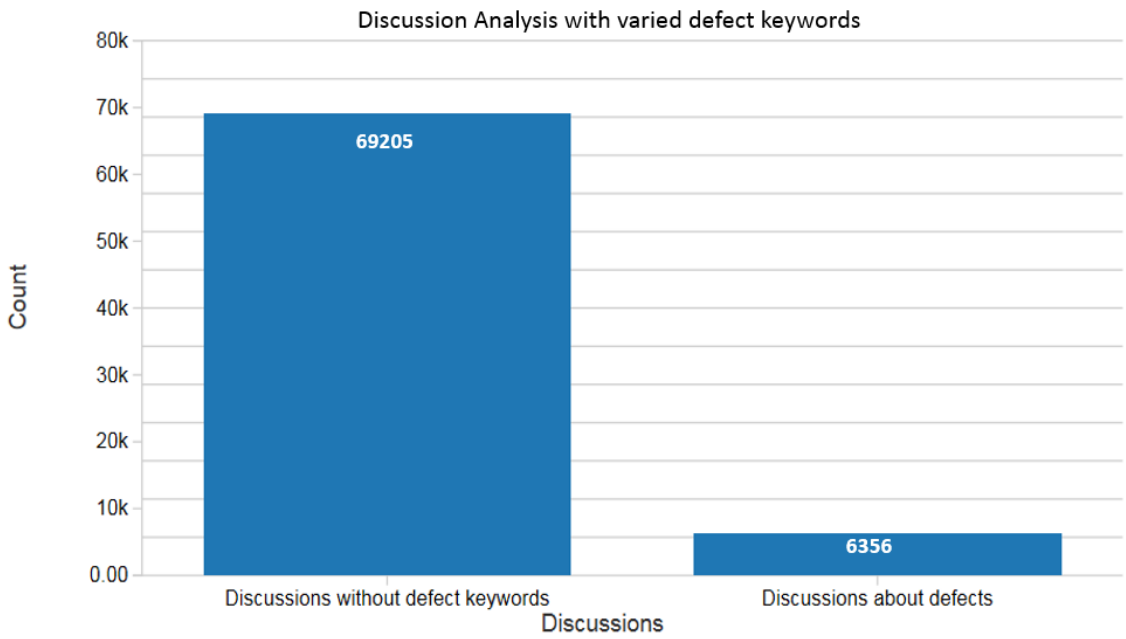


Figure 5.8: Distribution of the discussions based on the varied defect keywords

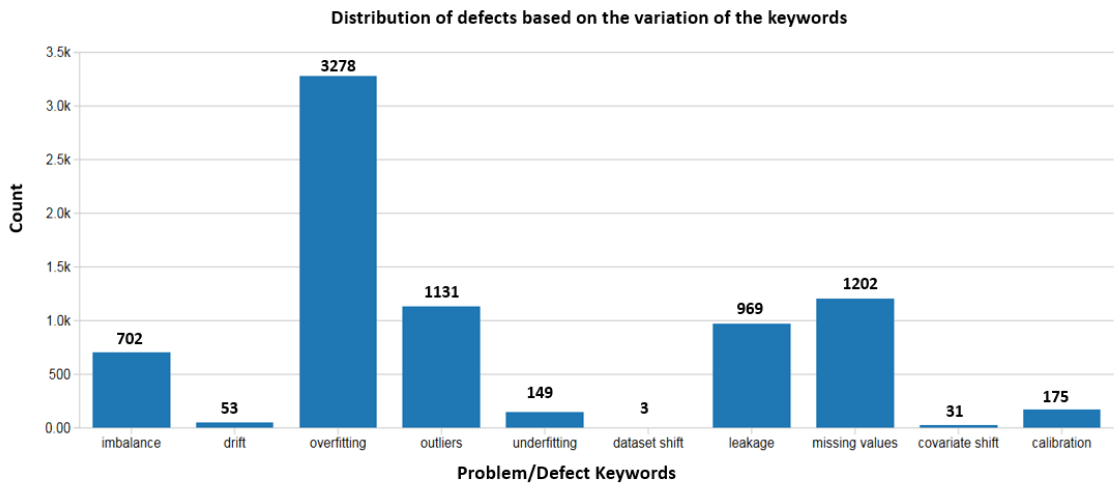


Figure 5.9: Distribution of the defects based on the varied keywords

5.1.4 RQ1: Result Summary

From the results of expert interviews, we found that the common defects experienced by the interviewed data scientists were data quality checks, class imbalance, leakage, multicollinearity, dataset shift, outliers, overfitting, underfitting, calibration error, and false positives/false negatives. The experts from our case study design answered that the first 6 defects i.e. data quality checks, class imbalance, leakage, multicollinearity, dataset shift, and outliers could be identified while examining the data quality. And, the defects like overfitting, underfitting, calibration error, false

positives/false negatives were found to be detected from the model after it is trained. Apart from that, the interviewed data scientists mentioned that the defects like leakage, dataset shift, and class imbalance could be detected from the data and also be verified by testing the model.

From the results of the literature review, we found that the common defects discussed were missing values in the data, the shift in the distribution of data, overfitting, outliers, size of the data, underfitting, class imbalance, leakage, and duplicate rows. Finally, from the results of the programmatic approach, defects such as overfitting, outliers, missing values, leakage, class imbalance, underfitting, calibration errors, and dataset shift were identified from the text corpus of the discussions data extracted from Kaggle.

We also observed the trend of these defects based on their frequency of occurrence in all the three conducted methods. For the sake of analysis, we categorize the defects as frequently, moderately and least frequently discussed. The trend we observed was that overfitting and missing values (data quality) were more frequently discussed defects in all the three methods we had conducted. And, defects such as leakage and class imbalance were moderately discussed in all the three conducted methods.

We see a difference in the trend with the other defects. The three methods conducted by us do not rank the defects unanimously. The results of expert interviews show that the defects such as calibration error and underfitting were moderately discussed. Whereas, the same two defects were discussed least frequently in the literature review and the programmatic approach as compared to the other defects. Conversely, we observed that defect outliers was frequently discussed in the literature review and the programmatic approach. But, very few data scientists mentioned about outliers during the interview. Another observation was that the dataset shift was least frequently discussed defect in the programmatic approach as compared to the other defects considered. Whereas, the same defect was discussed moderately in the expert interviews and literature survey. As the final step, we analyzed potential factors that lead to the differences between the results of the expert interviews and the results of the two latter methods. One such case was that of the outliers where the interviewed data scientists already handled the problem of outliers. Thus, it was least frequently discussed in the responses collected from expert interviews.

Based on the results obtained by conducting the aforementioned approaches, the following defects, problems, or errors that are specific to the field of ML, are provided in Table 5.3 on the following page. We have included all the defects provided in Table 5.3 on the next page in the scope of the thesis, except for outliers. This was influenced by the results of the expert interview. The terms mentioned in Table 5.3 on the following page, are induced either due to the misinterpretation of the data and algorithms or due to the mishandling of data and parameters while developing the model by a data scientist. This mishandling or misinterpretation would cause the model to provide imperfect results. Hence, these terms can be referred to as “Errors” or “Defects”. These defects are hidden in the ML model affecting the quality and are not specified in the business requirements against which these can be confirmed.

Defect	Definition
Overfitting	Problem where a model learns too well from the data and can't categorize correctly. [Gér17] [SKM13]
Underfitting	A model cannot generalize the data well. [Gér17]
Class Imbalance	Arises when number of observations belonging to one class of target variable is significantly lower compared to the other class. [GYD+08] [Zhe15]
Multicollinearity	Finding out the correlation between all the features and between features and target variable. [Ste15] [Pau06]
Leakage	Model created and trained by using information outside the training dataset or the information that was not available at the time of occurrence of the event and was collected in the future. [OS13]
Dataset Shift	Distributions of training and testing data are different. [QSSL09]
Type 1 and Type 2 Errors (False positives and False negatives)	Type 1 Error: model predicts more false positives. Type 2 Error: model predicts more false negatives. [TFL18] [Ber17]
Outliers	Values that diverges or deviates from other values on data. [SKM13]
Calibration errors	Model calibration is the degree to which a model's predicted probability estimates true correctness likelihood. [OS13]

Table 5.3: Defects related to ML and their definitions.

5.2 Research Question 2 (RQ2)

RQ2 helps to infer the reasons for not addressing the defects listed in Table 5.3, which are gathered from answering RQ1. It is essential to collect details about the QA practices followed in the ML domain. Hence, expert interviews with the data scientists facilitate in gathering information about the QA practices conducted by them. Their responses are compared with the definitions in the ISO/IEC/IEEE 29119 series of software testing standards [CD13] [Pro13] [Tec15]. This comparison helps us to summarize the reasons for the inadequacy of the degree of QA conducted by the interview participants, to identify the ML related defects elicited from RQ1.

The interviews were conducted in person and the responses were documented in an interview form. The interview revolved around a set of two questions based on the testing practices followed by the data scientists to address the defects or errors specific to the field of ML. One question was included to know the reason for not conducting QA. This question was applicable if the participants answered that they did not follow any QA practice. There were two follow-up questions. Table 5.5 on the next page provides the information about the intent, over which questions were asked and the questions.

Twelve data scientists were interviewed to gather insights about the QA practices followed by them. Four senior data scientists, seven data scientists and one intern pursuing data science were interviewed in a semi-structured format. The interviewees were asked some general questions regarding their work and the topic of QA. It was followed by brainstorming on the goals of the

Intent of the Questions	Questions
Explore the testing practices followed by the interviewed data scientists to check the quality of the model they develop.	1. Are there any software testing/validation/QA phases like Unit testing, integration testing, system testing etc. followed to evaluate the quality of the dataset or performance of the ML/statistical model for the errors?
Gain insights about the type of QA measures taken by the interviewed data scientists to ensure the quality of the results produced by the model.	2. If first or second option was selected for the above question, please name the tests conducted?
Reason for the interviewed data scientists for not including QA measures as a part of their work practice.	3. If no, please let me know any factors due to which QA is not conducted?
Follow-up questions to obtain additional information about the thesis.	4. Which testing types would you suggest that treat the ML model as black box? Please give a short rationale or explanation. 5. Additionally, please let me know if you have any other suggestions regarding with my master-thesis:

Table 5.4: Interview Questions for RQ2.

thesis and the relevance of the interview for the RQ2. Afterward, they were asked about their experiences and opinions on the QA process they perform and the reason if they are not performing any QA. Finally, the interview was ended with some follow-up questions requesting for additional information. Demographics summary of the experts is shown in the following table.

Attribute	Attribute Value	Number of experts
Type of organization	Services	12
Qualification of the interviewees	Doctorates (PhD) Master's degree Bachelor's degree	4 6 2
Age of Data Scientist (in years)	20-30 30-40 40-50	5 4 3

Table 5.5: Demographics data of the interviewees for RQ2.

Question 1: Are there any software testing/validation/QA types like unit testing, integration testing, system testing, etc. followed to evaluate the quality of the dataset or performance of the ML/statistical model for the errors?

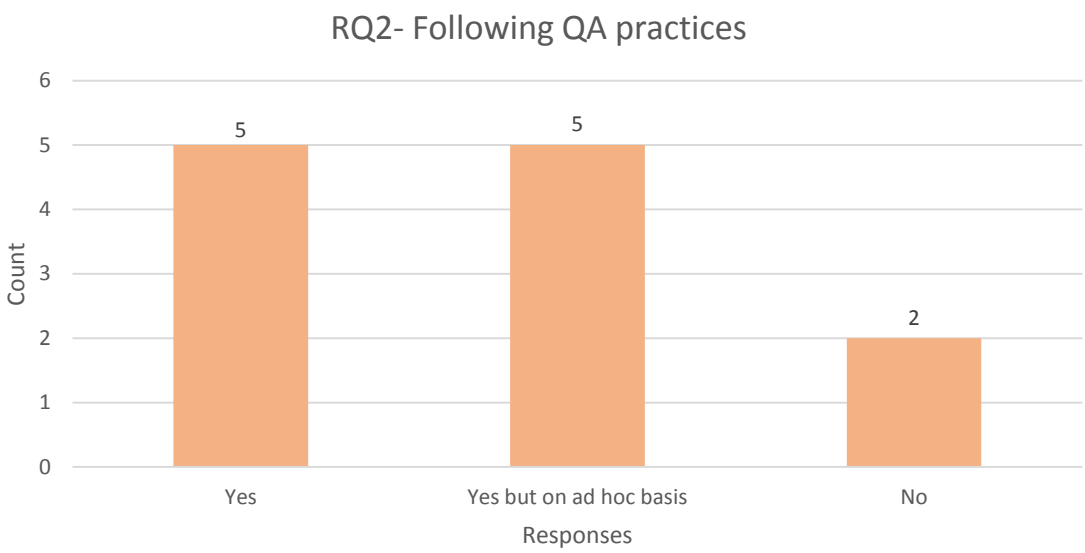


Figure 5.10: Response from experts on following QA practices.

Responses from Experts

The response to this question is illustrated by graph Figure 5.10. Five experts stated that they did testing on an ad hoc basis for the model they develop. The same number of data scientists answered that they conduct testing of data and the model they develop. While two data scientists expressed that they won't follow the mentioned types of QA. Through the responses of the data scientists, it was revealed that the majority of the data scientist follow QA as a part of their work.

Question 2: If the first or second option was selected for the above question, please name the tests conducted?

Responses from Experts

From the Figure 5.11 on the facing page, we observe the responses obtained. It was revealed that six data scientists performed quality checks on data, unit testing on the code and five performed model quality checks. Two data scientists responded that they performed integration testing and one response was obtained for model calibration check. The data scientists answered that the quality checks conducted on data were mostly checking for missing values and outliers in data. Likewise, the model quality checks were about evaluating the model quality metrics and comparing the performance of the model developed with a simple and a complex algorithm. In a nutshell, we can sense that QA is carried out but not formally.

Question 3: If no, please let me know any factors due to which QA is not conducted?

Responses from Experts

Surprisingly, two responses indicated that no formal QA procedures were conducted due to the lack of knowledge. Also, no official guidelines were specified to verify the quality of data and the model. Additionally, they stated that they assessed the model quality based on the metrics obtained when they performed cross-validation.

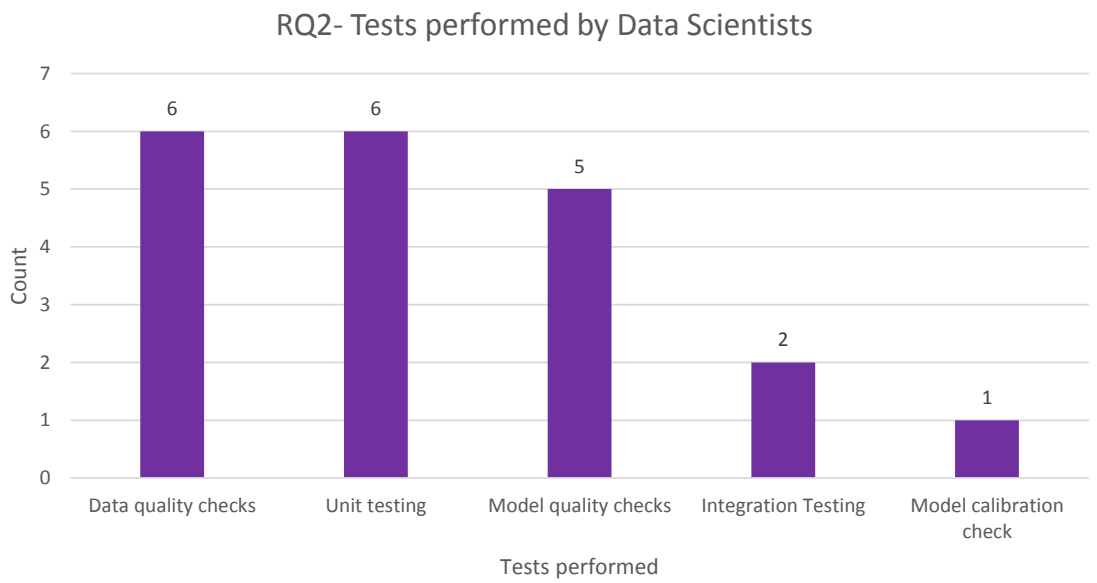


Figure 5.11: Response about the tests performed by the experts as QA practice.

Follow-up question: Which testing types would you suggest that treat the ML model as black box? Please give a short rationale or explanation.

Responses from Experts

For the follow-up question, most of the data scientists felt the need to test the ML model like black box. Furthermore, they emphasized to have a thorough check conducted to reveal possible flaws in data like missing values, duplicate rows and columns, check the correlation of each label with another label and also with the target variable, and check the distribution of classes in the target variable. Similarly, the emphasis was laid on testing the model by obtaining all possible metrics like accuracy, precision, recall, etc. for evaluation. Other suggestions included to develop a model with a simple algorithm and use it as a reference to compare other models developed with complex algorithms, ensuring the cross-validation done, checking the training time, including some statistical tests, and thinking model like gray box rather than black box. One additional suggestion was to consider testing based on the use case and not generalizing everything.

5.2.1 RQ2: Result Summary

From the responses acquired from twelve experts, we infer that the software testing phases commonly conducted by six data scientists are unit testing. This is in line with our expectation, as unit testing is the initial test phase in the software testing [Pro13] [GVE08]. Thus, serving as the fundamental for further testing phases such as integration, system and user acceptance testing [Pro13]. On the other hand, the results of the expert interview suggested that integration testing was least frequently mentioned, with two votes. Whereas, other software test phases like system testing and user acceptance testing were not mentioned by the interviewed data scientists.

In parallel, the responses from the majority of the interviewed data scientists were centered on testing the quality of the data. This test was also mentioned by six data scientists from our case study. The same importance given to unit testing and testing the data quality by the interviewed data scientists signifies that the data quality test is indisposable in ML. Also, five experts from our case study emphasized on obtaining the model evaluation metrics. Since they analyzed the quality of the ML model based on the model evaluation metrics. Another interesting response obtained by two interviewed data scientists was that there was no formal specification given to them for validating the results produced by the model.

The results of the expert interview showed that it is equally important to test the data, the written algorithm (code), and the model (which is our system under consideration). The discussions with data scientists also revealed the difficulties connected with putting thresholds to identify certain data or ML model quality defects.

Based on the definitions provided in the ISO/IEC/IEEE 29119 series for Software Testing in [CD13], [Pro13], and [Tec15], a defect is detected when the software application exhibits a deviation in behavior which is against the formal specifications or requirements. In the domain of ML, there are no formal specifications or requirements provided to address the quality of the results produced by the ML model which helps to detect the defects that are obtained as a part of RQ1 [MKA07]. Without formal specifications, it is difficult to draft the test cases or test procedure form which can be used to capture the test execution results [Doc13].

To summarize the above, the defects in the model cannot be detected by conducting just unit testing (and integration testing as well). Instead, we should perform data quality and model quality tests in addition to the unit testing and integration testing. We can include the analysis of the model evaluation metrics in the model quality test. And, we can perform the model quality test like the system testing phase, in a more suitable and precise manner for ML by preparing a test procedure form and capturing the test execution results.

5.3 Research Question 3 (RQ3)

Inferences from the results of RQ1 and RQ2 form the foundation for outlining the additional tests as a part of RQ3. This section begins with an explanation about the test automation scripts, comprising the additional tests. These are required to identify the defects found in RQ1. Next, these automation scripts (with additional tests) are evaluated by considering three scenarios. Finally, feedback from the data scientists on the advantages, disadvantages, understandability, and usability of the developed tests is collected and analyzed.

5.3.1 Test Automation Script Design

Data is important for any ML activity [PF13]. In this subsection, we provide the details of the data, considered for training and evaluating the model quality. Next, we explain the two test automation scripts, designed to detect the defects in the data and in the chosen model. These tests were designed, based on the responses of the data scientists who participated in the interview for RQ2 and on the suggestion by Braiek et.al in their paper [BK18].

Two test automation scripts, namely, dataset_evaluation_test and model_quality_test were developed in R language and the results of the tests were rendered in R markdown file. The scripts assist in having an additional level of verification and validation. This informs the data scientists about the quality of the data and the model they develop. Figure 5.12 shows the phases in a data science process pipeline, where these scripts were applied. The dataset_evaluation_test script is used to detect the possible defects in the data. The model_quality_test helps in identifying the defects after the model development and training.

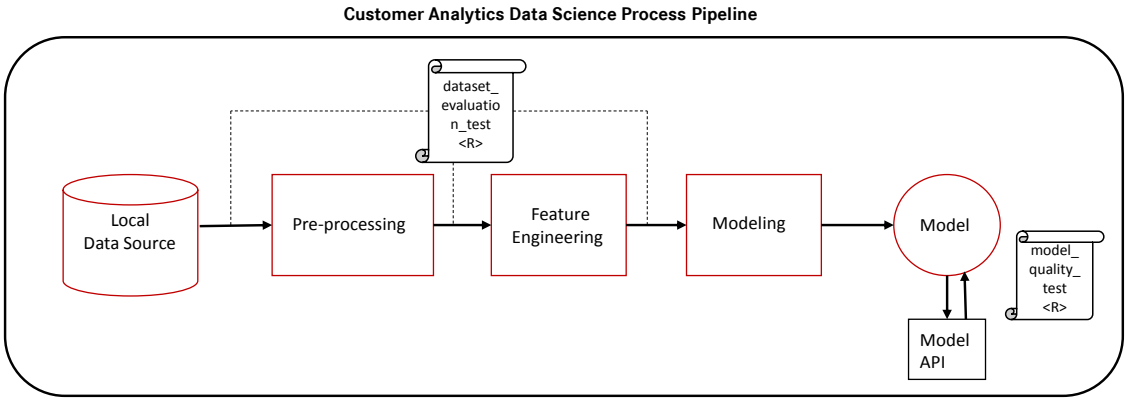


Figure 5.12: Data Science Process Pipeline with the proposed test automation scripts

These two test automation scripts were designed to detect errors like leakage, class imbalance, overfitting, underfitting, and so on, that are related to the domain of ML. These were extracted based on the experience of the data scientists as a part of interviews for RQ1. Hence, we can make use of the experience-based test design technique [Tec15] to frame the test cases. From Section 4.2, the scope we had considered was to test the ML model as a black box and so the test method considered was the black-box method [Lim09]. Based on the summary of the results from the interview for RQ2, system testing [Pro13] phase was identified as an important phase, following the data quality test and unit testing. According to [Pro13], the functional testing type assures the functional quality aspects of the tested software. Therefore, we used the functional testing type for validating the model.

5.3.1.1 Dataset

The data used is the publicly available "Titanic: Machine Learning from Disaster" dataset from Kaggle [Kag19a]. The objective of the challenge is to analyze the features that were attributed to the survivors in surviving the shipwreck of Titanic and predict whether a given person survived the disaster or not using ML skills and tools. This is based on the data, containing information regarding the passengers who were assumed to be on board RMS Titanic during the disaster. There are two files "train.csv" and "test.csv". The file "train.csv" is used to analyze the pattern for solving the problem and consists of a target column indicating "1" if the passenger survived and "0" signifying the death of the passenger in the tragedy. The data also contains other variables which are referred to as "features". The file "test.csv" contains a fresh set of unseen data on which predictions need to be made post analysis, model development, training, and testing. For evaluation, we only consider "train.csv" as the training data.

The training dataset contains 891 rows of data having details about the passengers and 12 columns indicating the different attributes required for analysis and generalization of patterns. Details about each column in the dataset are provided in Table 5.6 [Kag19a]. An overview of the dataset is illustrated in Figure 5.13 [Kag19a].

Column Name	**Description**	**Datatype**
PassengerID	Unique ID associated with each passenger.	Numeric
Survived	Target or outcome variable indicating whether a passenger survived the disaster or not.	Numeric
Pclass	Type of accommodation and facilities availed by a passenger. This column also suggests the socio-economic status of a passenger.	Numeric
Name	Name of the passenger	String
Sex	Passenger's sex	String
Age	Passenger's age	Numeric
SibSp	Number of siblings or spouses of the passenger aboard the voyage	String
Parch	Number of parents or children of the passenger aboard the voyage	Numeric
Ticket	Ticket number of the passenger	String
Fare	Price paid for the ticket	Numeric
Cabin	Cabin number of the passenger	String
Embarked	Place where the passengers boarded the ship	String

Table 5.6: Details of the training dataset (Kaggle's Titanic competition) [Kag19a]

# Passen...	# Survived	# Pclass	A Name	A Sex	# Age	A SibSp	# Parch	A Ticket	# Fare	A Cabin	A Embar...
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25		S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen , Miss. Laina	female	26	0	0	STON/O2. 3101282	7.925		S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1	C123	S
5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05		S

Figure 5.13: Dataset of Kaggle's Titanic competition [Kag19a]

5.3.1.2 Dataset Evaluation Test

This subsection explains the tests developed in the automation script "dataset_evaluation_test¨ aimed at informing the data scientists about the profile of the data. Data scientists take care of analyzing the data to ensure that the data is deprived of data quality issues. It also helps to derive sense, needed for further understanding of the patterns, identified by the ML model. The dataset evaluation test is performed before the feature engineering stage (see Figure 5.12 on page 63) as the script informs data scientists about various errors that the data might induce in the model, when not dealt properly. The same script can also be applied before the pre-processing stage to verify the profile of the data. And, the script can also be executed after the feature engineering stage, to analyze the newly formed feature impact on the target variable.

The developed script informs the data scientists on class imbalance, leakage, multicollinearity, and covariate shift. Other performed tests include the signature check and check for missing values. Class imbalance occurs when the percentage share of one class in the distribution of the target variable is lower than the percentage share of the other classes [Zhe15]. To test for the defect, the percentage share of each class in the distribution of the target variable is calculated. Hence, we inform the data scientist on this defect when there is a significant difference in the percentage share of one class. Similarly, we can test the class distribution of categorical predictors.

Leaky predictors are the independent variables that convey information, collected after the occurrence of an event. This information is not known at the moment of the event occurrence. These predictors are said to correlate highly with the target variable [OS13] [KRPS12]. Therefore, target leakage is identified by the high value of the chi-squared correlation coefficient with the target variable in the data. On the other hand, data leakage is traced by checking for the duplicate values of a predictor in the training and testing data.

Testing multicollinearity [Ste15] [Pau06] is conducted by checking the correlation between the predictors which are numeric values. We also check conditional independence between numerical and categorical predictors. Finally, the correlation of all the predictors with the target variable is tested by computing the chi-squared coefficient and information gain. Highly correlated predictors are reported for further analysis.

For the dataset shift defect, online research suggested checking whether the data in the training and testing dataset has the same distribution by developing separate models [Fra14] [Sta19]. In our work, the dataset shift is detected by checking whether the population stability index is higher than a statistically determined threshold [Yur18].

To conclude, the first test automation script includes the above-mentioned tests and it informs about the inconsistencies, present in the data.

5.3.1.3 Model Quality Test

In this subsection, an explanation is provided about the tests, executed in the automation script model_quality_test designed to assure the model quality. Further, we simulate the system test phase, realizing the model as a black box. The considered defects are overfitting, underfitting, leakage,

class imbalance, calibration errors, model decay due to dataset shift, type 1, and type 2 errors. The detection criteria developed in the script of such defects are simple and based on the definition of the defects rather than developing complex ML models or neural networks.

The entry points to commence with the testing phase are the model object, the dataset split into training and testing sets. The dataset split determines how the data is utilized in generalizing the patterns for the model to generate results. Thereby, a conclusion about sample selection bias is made. At first, predictions are made on the testing data and training data on the testing data without the target variable, based on the identified pattern in the training data. Next, the confusion matrix is generated with the model predictions and true values of the target variables. From the confusion matrix, various calculated model evaluation metrics like accuracies, balanced accuracies, precision, recall, F1 score, Log Loss, etc. are displayed in the test results [Zhe15].

We calculate the difference between the balanced accuracy of the model prediction made on the testing and training data. If the difference exceeds the use case specific threshold value (5% in our case), the test notifies that the model is overfitted or affected by leakage [OS13]. For testing underfitting, the balanced accuracy of model prediction on the training data is compared with a selected threshold (60% in our case). If it is less than the selected threshold value, the test signals underfitting defect. Similarly, if the predicted positive and negatives values are significantly lower than the established boundary value (80% for our use case), then type 1 and type 2 errors are detected respectively [PF13]. Model decay due to the dataset shift can be determined by comparing the balanced test accuracies of the model obtained for the current and the previous timeframe [OS13] [QSSL09] [Fra14].

In the developed scripts, the metric accuracy was considered to determine the class imbalance of the model. Further, a set of graphs like the Precision-Recall curve [SR15], AUC of the receiver operating characteristic (ROC) curve [PF13], and cumulative gain charts[8] extend the analysis. These graphs facilitate data scientists with information to make better decisions on the class imbalance defect and the overall model quality. Likewise, the calibration plot helps in detecting and determining the calibration errors [OS13].

To summarize, the second test automation script simulates the system testing phase for the ML domain. This script tests the defects and informs on the model quality.

5.3.2 Evaluation

To evaluate our implementation, we considered three scenarios. In the first scenario, we do not conduct the dataset_evaluation_test and start with the model test. This is done by executing only the second script i.e. model_quality_test, to identify overfitting, underfitting, and other defects affecting the chosen model. In the second scenario, we execute dataset_evaluation_test to identify the possible defects in the data. Based on this information, we select the right features and develop a ML model. Later, the quality of the developed model is evaluated by executing the model_quality_test script. In the third scenario, we execute the second automation script on the third party code, submitted to Kaggle. This code implementation is tested for any ML related defects.

[8] CG Chart: https://www.ibm.com/support/knowledgecenter/de/SSLVMB_24.0.0/spss/tutorials/mlp_bankloan_outputtype_02.html

5.3.2.1 Scenario 1: Initial Functional Test

The initial scenario of testing the ML model as black box by providing the inputs and analyzing the outputs is considered here. Initially, we check whether the ML model chosen for evaluation, produces results when applied on the testing data without any failure. Later, we validate whether the results are “0”s and “1”s as it is a binary classification use case. In these two cases, we simulate the system testing phase as conducted for a software application.

Figure 5.14 shows that when the (testing) data was applied to the chosen model, it predicted the results in the form of “0”s and “1”s without any failure. Thereby, the test cases for validating the model functionality and the format of the prediction results were satisfied.

Step 6. Testing Stage

(i) Model prediction with testing data and training data

Sc1TC1: Validate Model functionality.

Sc1TC2: Validate the format of the prediction results.

```
## Model prediction is given below:
```

```
##   [1] 0 0 1 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0
##  [36] 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0
##  [71] 1 1 0 0 1 1 0 1 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1
## [106] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0
## [141] 0 0 1 0 0 0 1 1 1 1 0 0 1 0 0 1 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1
## [176] 1 0 0
## Levels: 0 1
```

Figure 5.14: Validating the prediction results of the ML model

An extension to the above mentioned steps was to determine of the model quality. This was evaluated by testing for the defects, which were extracted as a part of RQ1. The objective is to prove that system testing, when conducted similar to a general software application, would not be adequate to ensure the goodness of the model’s results. This test reveals the possible defects the model would be suffering from.

We executed the model_quality_test on the chosen ML model and verified for the defects. From Figure 5.15a, we observed that there was no sign of overfitting or leakage. But, the model’s balanced test and train accuracies were very low. Figure 5.15b displayed that the model was not affected by underfitting but the balanced training accuracy was low. The model also predicted more false positives and false negatives.

(iv) Test for Overfitting and Leakage

Sc1TC3 and Sc1TC5: Verify for Overfitting and Leakage.

The testing accuracy is 62.83%.

The training accuracy is 62.52%.

The difference between the accuracy calculated on training data and testing data is 0.31%.

```
## The difference in accuracies is less than 5%.
```

```
## No Overfitting or Leakage.
```

(a) Test results for overfitting and leakage.

(v) Test for Underfitting

Sc1TC4: Verify for Underfitting.

Accuracy of predictions on training data is 62.52%.

```
## Model is NOT underfitted.
```

(vi) Test for Type 1 and Type 2 errors

Sc1TC9: Verify for Type 1 errors.

Percentage of positive predicted values is 73.19%.

Sc1TC10: Verify for Type 2 errors.

Percentage of negative predicted values is 60%.

```
##  Check for both Type 1 and Type 2 errors: More False positives and False negatives !!
```

(b) Test results for underfitting, type 1 and 2 errors.

Figure 5.15: Verifying the ML model results for defects.

Furthermore, Figure 5.16 showed that the model was moderately calibrated. The defects hidden in the model were revealed through the model_quality_test script. This helped us to realize the need for having additional tests that assure the quality of the model against the defects elicited from RQ1.

(ix) Check Calibrations

Sc1TC7: Verify for Calibration error.

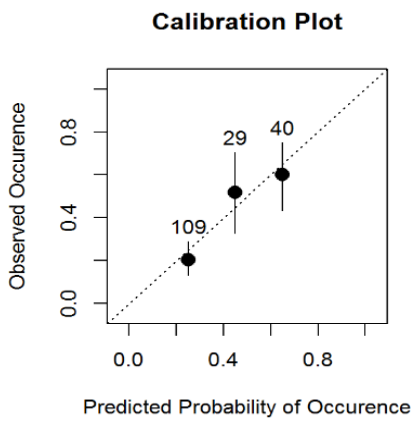


Figure 5.16: Calibration plot of the ML predictions.

Appendix A.4.1 provides the information regarding the test procedure form consisting of the activities performed in this scenario, the examination of the result, the actual results observed during testing, and the test result. The template of the test procedure form is based on [Doc13]. The above mentioned checks like validating the prediction results of the model and the tests for overfitting, underfitting, calibration, etc., forms the activities. And, the same are documented under the activities column . Experience-based technique [Tec15] was employed for designing the activities performed in this scenario. The examination of the result column contains the correct behavior of the model, whereas we documented the observed behavior of the model in the actual result column.

We marked "Pass" in the test result column if the actual result matched the expected. “Warning” was marked in the test result when there was a deviation with respect to actual and expected results. The status “Warning” was used to alert the data scientists about the defects in the developed model. This provides them with the freedom to research further on the defects rather than concluding the test is failed. From Figure A.1, we observed deviation between the expected and actual results for the tests of calibration, type 1, and type 2 errors. Also, the balanced train and test accuracies of the model were significantly low. Hence, we marked the test result with the status as "Warning" for those tests. And, the rest of the tests were marked with "Pass" as the status.

5.3.2.2 Scenario 2: Data and Model Quality Test

In this scenario, we first analyze the possible defects and inconsistencies in the data by running the dataset_evaluation_test script. The results of the dataset_evaluation_test revealed interesting points in the data.

- Result of the class imbalance check: Figure 5.17 on the next page indicates that there is no possibility of the class imbalance problem. Also, we observe the class distribution of the target variable along with the graph for better analysis.
- From Figure 5.18 on the following page we see that there are no duplicates in the data. The size of the dataset is sufficient enough to conduct training. But test for leakage showed that there is no data leakage but, target leakage is possible due to the columns Name and Ticket.
- Results of the signature check and completeness check: Figure 5.19 on page 71 indicates that there is no mismatch with the column types and names of the training and testing data. Also, the dimensions of the considered training and testing data are better. But, we see that there are NA values and empty values in the columns Age, Cabin and Embarked. Also, the number of such inconsistencies are displayed.
- Dataset shift test identified that apart from the PassengerId, all other variables in the training data are likely to cause drifts. Figure 5.20 on page 71 displays the results of the dataset shift test.
- Mutlicollinearity test revelead that there is moderate correlation between the columns Fare and the Pclass, as well as between the columns Parch and SibSp. This can be seen from Figure 5.21 on page 72.

Imbalanced Class check

```
## Class imbalance might not arise.
```

```
## target_variable
##        0        1
## 61.61616 38.38384
```

```
## Number of classes in the Target variable: 2 | Target variable type: integer
```

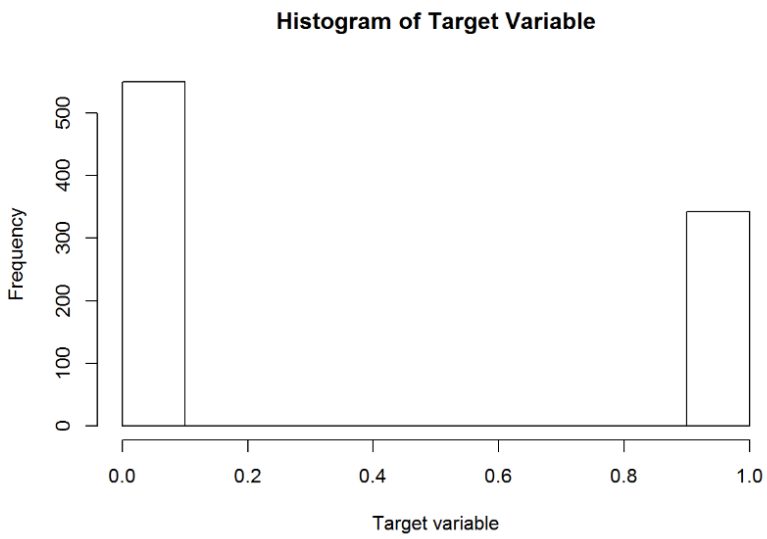


Figure 5.17: Test for class imbalance

Duplicates Check

```
## [1] "The percentage of duplication in the dataset is 0 %."
```

Size of dataset

```
## [1] "The train dataset is large enough to conduct training."
```

Leakage check

Target Leakage: Check highly correlated predictors

```
## Warning!! Target leakage possible.
```

```
## Following predictors are highly correlated with the target variable.
```

```
## [1] "Name"   "Ticket"
```

Data Leakage: Check for duplicates in Training and Testing data

```
## No Data Leakage
```

Figure 5.18: Test for duplicates and leakage

Signature Checks

Check Column types of Training and Testing data

```
## Column types are same in training and testing dataset.
```

Check Column names of Training and Testing data

```
## Column names are same in the training and testing dataset.
```

Check the dimensions of Training and Testing data

```
## The dimension of the training dataset is better than the testing dataset.
```

Number of rows in train dataset: 713.

Number of rows in test dataset: 178.

Completeness Check

```
## There are 139 NA values in the column: Age
```

```
## There are 553 empty values in the column: Cabin
```

Number of non NA values in each column in %

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##   100.00000   100.00000   100.00000   100.00000   100.00000    80.50491
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##   100.00000   100.00000   100.00000   100.00000   100.00000   100.00000
```

Number of non empty values in each column in %

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##   100.00000   100.00000   100.00000   100.00000   100.00000          NA
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##   100.00000   100.00000   100.00000   100.00000    22.44039    99.85975
```

Figure 5.19: Test for signature and completeness check

Check dataset shift

```
## Column: PassengerId | PSI value: 0.174054824453785 | There is 'NO' drift from column PassengerId
```

```
## Dataframe: Survived | PSI value: Inf | There might be DRIFT from column Survived
```

```
## Dataframe: Pclass | PSI value: 20.7241239140201 | There might be DRIFT from column Pclass
```

```
## Dataframe: Name | PSI value: 20.7241239140201 | There might be DRIFT from column Name
```

```
## Dataframe: Sex | PSI value: 20.7255881878603 | There might be DRIFT from column Sex
```

```
## Dataframe: Age | PSI value: Inf | There might be DRIFT from column Age
```

```
## Dataframe: SibSp | PSI value: Inf | There might be DRIFT from column SibSp
```

```
## Dataframe: Parch | PSI value: 20.7241239140201 | There might be DRIFT from column Parch
```

```
## Dataframe: Ticket | PSI value: 20.7241239140201 | There might be DRIFT from column Ticket
```

```
## Dataframe: Fare | PSI value: 20.7241239140201 | There might be DRIFT from column Fare
```

```
## Dataframe: Cabin | PSI value: 20.7241239140201 | There might be DRIFT from column Cabin
```


Figure 5.20: Test for dataset shift

Check for Multicollinearity

Check the correlation between the predictors that have continuous variable

```
## Columns with numeric type that are less correlated:
```

```
## [1] "PassengerId" "Pclass"      "Age"         "SibSp"       "Parch"
## [6] "Fare"
```

	PassengerId	Pclass	Age	SibSp	Parch	Fare
PassengerId	0	0.02	0.03	0.04	0.02	0.00
Pclass	0	0.00	0.34	0.08	0.00	0.56
Age	0	0.00	0.00	0.23	0.20	0.11
SibSp	0	0.00	0.00	0.00	0.44	0.17
Parch	0	0.00	0.00	0.00	0.00	0.24
Fare	0	0.00	0.00	0.00	0.00	0.00

Check the correlation between the predictors with continuous values and predictors with categorical values

	PassengerId	Pclass	Age	SibSp	Parch	Fare
Name	Dep|Cond.Dep	Dep|Cond.Dep	Dep|Cond.Dep	Dep|Cond.Dep	Dep|Cond.Dep	Dep|Cond.Dep
Sex	Ind	Dep|Cond.Dep	Dep|Cond.Dep	Dep|Cond.Dep	Dep|Cond.Dep	Dep|Cond.Dep
Ticket	Dep|Cond.Dep	Dep|Cond.Dep	Dep|Cond.Dep	Dep|Cond.Dep	Dep|Cond.Dep	Dep|Cond.Dep
Cabin	Dep|Cond.Dep	Dep|Cond.Dep	Dep|Cond.Dep	Dep|Cond.Dep	Dep|Cond.Dep	Dep|Cond.Dep
Embarked	Dep|Cond.Dep	Dep|Cond.Dep	Dep|Cond.Dep	Dep|Cond.Dep	Dep|Cond.Dep	Dep|Cond.Dep

Figure 5.21: Test for multicollinearity

Based on the results of the dataset_evaluation_test, we corrected the data, extracted the relevant features, developed a suitable ML model and trained the model with the training data. We executed the model_quality_test script on trained model. This was done to test the model for possible defects on the testing data, when considered as black box. This strategy serves as a resemblance to the system testing phase with additional tests, which are designed to reveal possible defects the model would be suffering from.

The test results seen from Figure 5.22 on the facing page, indicates that there was significant improvement in the accuracies and there were no possibilities of overfitting, underfitting, and leakage. There were no type 1 and type 2 errors as well.

The graphs seen from Figure 5.23 on the next page, were considered to display the precision-recall curve (PR curve), the receiver operating characteristic curve (ROC), the Gain chart and the calibration plots. PR curve displays the trade-off between precision and recall. ROC shows the trade-off between true positive rate and false positive rate.The Gain chart shows the how well the model predicts true positives and the calibration plot depicts the calibration of the model prediction results. The graphs depicted good model quality results and indicated no class imbalance defect. Also, the model results were well calibrated compared to the previous scenario explained in the Section 5.3.2.1.

(iv) Test for Overfitting and Leakage

Sc2TC4 and Sc2TC6: Verify for Overfitting and Leakage.

The testing accuracy is 85.39%.

The training accuracy is 90.27%.

The difference between the accuracy calculated on training data and testing data is 4.88%.

```
## The difference in accuracies is less than 5%.
```

```
## No Overfitting or Leakage.
```

(v) Test for Underfitting

Sc2TC5: Verify for Underfitting.

Accuracy of predictions on training data is 90.27%.

```
## Model is NOT underfitted.
```

(vi) Test for Type 1 and Type 2 errors

Sc2TC10: Verify for Type 1 errors.

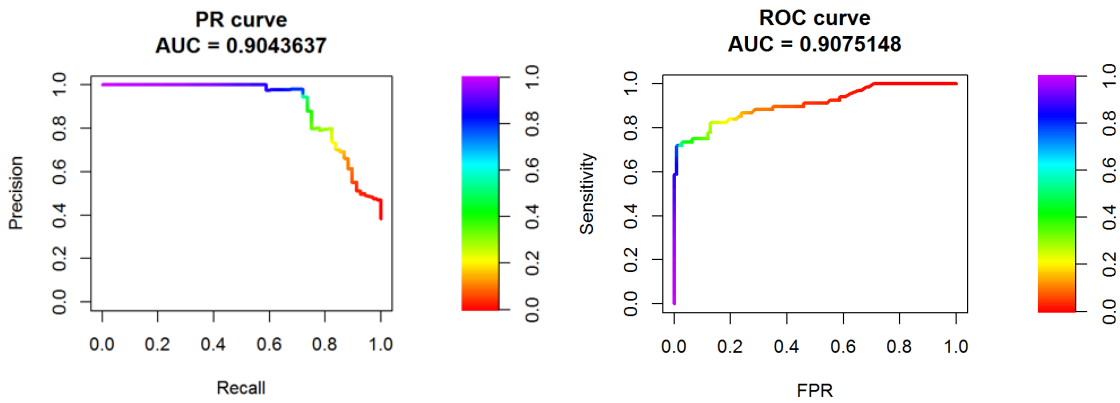
Percentage of positive predicted values is 85.48%.

Sc2TC11: Verify for Type 2 errors.

Percentage of negative predicted values is 94.34%.

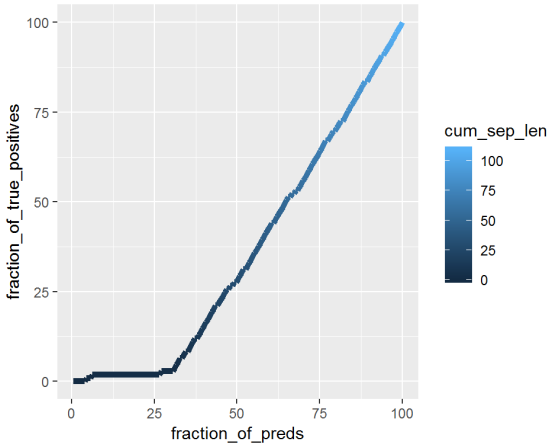
```
## No Type 1 and Type 2 errors.
```

Figure 5.22: Scenario 2: Results of the tests conducted on the ML model.

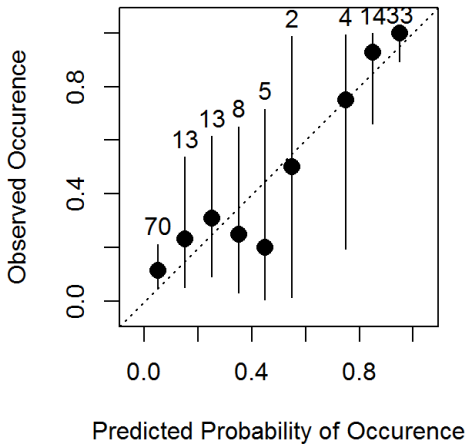


(a) Precision-Recall curve

(b) Receiver Operating Curve



(c) Cumulative Gain charts



(d) Calibration plot

Figure 5.23: Plots considered in the test approach to analyze the quality of the ML model

Figure A.2 in Appendix A.4.2 shows the test procedure form. This provides the overall status of the tests conducted. We observed deviations in the tests for completeness, leakage, dataset shift, and calibration results. Hence, we indicated the test result column for those tests with the status of "Warning". This test procedure form helps in tracking the defects and take suitable measures in fixing those defects, if applicable.

5.3.2.3 Scenario 3: Testing the Third Party Code

The organizers of the competition “Titanic: Machine learning from Disaster” in Kaggle had set “Accuracy” as the metric to determine whether the model performs its function of prediction correctly [Kag19b]. In this sub section, the codes submitted to the Kaggle forum were analyzed for the public score provided by the Kaggle. This public score[9] is calculated based on the accuracy of the model’s prediction on the test data provided in the competition. One such submitted code is considered and subjected to the tests that are designed as a part of the thesis. Through the set of additional tests, the model is checked for accuracy along with additional metrics like precision, recall, specificity, F1 score, etc. Besides these evaluation metrics, the model is investigated for individual defects. This evaluation can be used to illustrate the feasibility of the extension of software testing to modules specific to ML.

The code submission considered from Kaggle displayed a score (accuracy in this case) of “80.382” [elm19]. The model_quality_test script was executed for this code implementation and discovered that the accuracy of predictions on the testing data was found to be “81.08”. Whereas, the accuracy of predictions on the test data by balancing the distribution of the target class was “79.09” as seen from the model performance metrics in Figure 5.25. So we can infer that the accuracy hosted by the Kaggle and the accuracy calculated by the model_quality_test are almost equal. On further analysis for individual defects, the script identified that the ML model was overfitted as seen from Figure 5.24. Also, the percentage of false negatives were more and the results of the model were not properly calibrated. All these can also be observed from Figure 5.27 and Figure 5.26.

[9]Kaggle score: https://www.kaggle.com/c/titanic/leaderboard

(iv) Test for Overfitting and Leakage

Sc3TC3 and Sc3TC5: Verify for Overfitting and Leakage.

The testing accuracy is 79.09%.

The training accuracy is 88.11%.

The difference between the accuracy calculated on training data and testing data is 9.02%.

```
## The difference in accuracies is more than 5%.
```

```
## Model is either Overfitted or there is Leakage. !!
```

```
## Check for leaky predictors in case of 'Leakage'.
```

```
## Check for Bias-Variance trade-off for 'Overfitting'.
```

Figure 5.24: Test Result for Overfitting and Leakage

Model Performance Metrics

metric_names	metric_values
Test accuracy	81.08000
Train accuracy	89.69000
Balanced Test accuracy	79.09000
Balanced Train Accuracy	88.11000
Precision	82.76000
Recall	87.59000
Positive Predicted Value	82.76000
Negative Predicted Value	77.92000
Specificity	70.59000
F1 score	85.11000
MCC score	59.42000
Log Loss	-13.22461
Area under the curve	79.08974
Classification Error	18.91892

Figure 5.25: Model Evaluation Metrics

The test procedure form in Appendix A.4.3 provide the results and status of the tests conducted. The activities for verifying overfitting, calibration, and type 2 errors were indicated with the status "Warning". This was based on the actual results of the model observed from Figure A.3. An additional activity was included to verify the accuracy published by Kaggle. Though, we observed the accuracy of the chosen ML model was found to be almost equal. We indicated the test result status as "Warning". This to check whether the accuracy improves if we fixed the defects like overfitting, calibration, and type 2 errors.

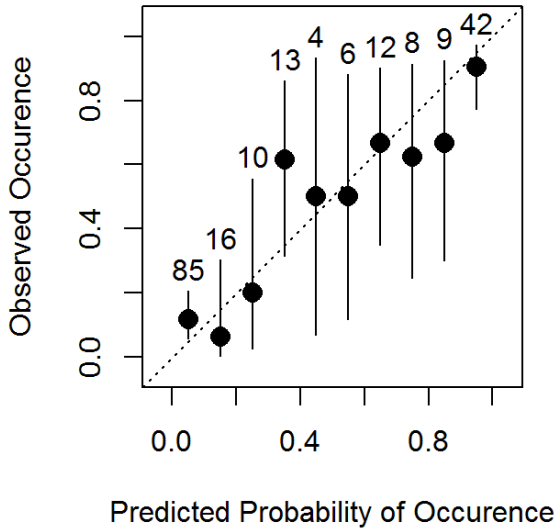


Figure 5.26: Calibration Plot

(vi) Test for Type 1 and Type 2 errors

Sc3TC9: Verify for Type 1 errors.

Percentage of positive predicted values is 82.76%.

Sc3TC10: Verify for Type 2 errors.

Percentage of negative predicted values is 77.92%.

```
## Check for Type 2 errors: More False negatives. !
```

Figure 5.27: Test Result for Type 1 and 2 errors

5.3.3 Feedback Interviews

The feedback helps to infer about the advantages and the disadvantages of the additional tests. It also provides information about the factors that motivate or demotivate the data scientists in using the additional tests, developed in this thesis. Finally, we infer the understandability and usability aspects of the additional tests along with the approach with which those tests are applied in the data science process pipeline. Seven data scientists were interviewed in semi-structure format to gather their feedback on the above mentioned aspects. Four senior data scientists and three data scientists were considered for the interview. The interviews were conducted in person and the responses were documented on a feedback form. Table 5.7 on the facing page provides the intent, over which questions were asked and the questions.

Of the seven participants, four had participated in the interviews conducted for both RQ1 and RQ2. These four participants had the complete idea about the research and their feedback was important in analyzing the overall progress of the research. Remaining three participants had participated in the interview conducted for RQ2 but not for RQ1. This kind of selection was done to analyze how understandable and usable the tests are for the data scientists, who were not familiar with the developed tests apriori. The demographics summary of the experts is shown in the following Table 5.8 on the next page.

Intent of the Questions	Questions
Advantages and disadvantages of the proposed testing approach	1. What are the advantages and disadvantages of the proposed testing approach in the domain of ML?
Motivating and demotivating factors	2. What would motivate you to use the proposed tests? 3. What would demotivate you to use the proposed tests?
Understandability	4. In your opinion, how understandable were the tests and the way in which it was performed?
Usability	5. In your opinion, are the proposed tests and the testing approach useful in establishing a QA phase in your work?

Table 5.7: Feedback Questions for RQ3.

The interviewees were first introduced to software testing aspects of the test design techniques, test types, phases, and methods. This was followed by an explanation about the additional tests considered and the phase in the data science process pipeline, at which these tests were applied. The intention was to demonstrate the simulation of a software testing practice to ML domain. The scenarios considered for the evaluation were explained and the results of each scenario were demonstrated. Finally, a feedback form was shared with them.

Attribute	Attribute Value	Number of experts
Type of organization	Services	7
Qualification of the interviewees	Doctorates (PhD) Master's degree	4 3
Age of Data Scientist (in years)	20-30 30-40 40-50	2 1 4

Table 5.8: Demographics data of the interviewees for RQ3.

Question 1: What are the advantages and disadvantages of the proposed testing approach in the domain of ML?

Responses from Experts

Six data scientists mentioned that the testing approach conducted in the thesis was systematic and efficient. Three data scientists mentioned that the automation script saves time and was easy to use. Three experts believed that the approach could be used as a standard to test the most important aspects of the ML model. In addition, two data scientists stressed the importance of the visual features, provided by the graphs. One data scientist mentioned that the approach includes several defects under its scope.

The important points, mentioned by the data scientists as disadvantages, were that the additional tests were not applicable to all use cases. Data scientistics also mentioned that they would need more information to be displayed on the test report. Lastly, it was mentioned that the additional tests would give an impression that the execution of these tests would cover all the defects and ensure 100% QA.

Question 2: What would motivate you to use the proposed tests?

Responses from Experts

Five data scientists mentioned that the tests and the approach was simple and useful. Two experts stated that the automation saves them time and hence they would like to use it. One data scientist mentioned that the tests can be easily integrated with their work practice. Another mentioned that the degree of generalization considered for the case of classification problem in ML.

Question 3: What would demotivate you to use the proposed tests?

Responses from Experts

Four data scientists mentioned that an additional effort is needed to conduct these tests and tune the thresholds to detect a defect, as it is use case dependent. One expert stated that running these tests requires additional time apart from the time required to train the ML model. Response received from one data scientist was that any hardware and software dependency would prevent from adapting these tests in their work.

Question 4: In your opinion, how understandable were the tests and the way in which it was performed?

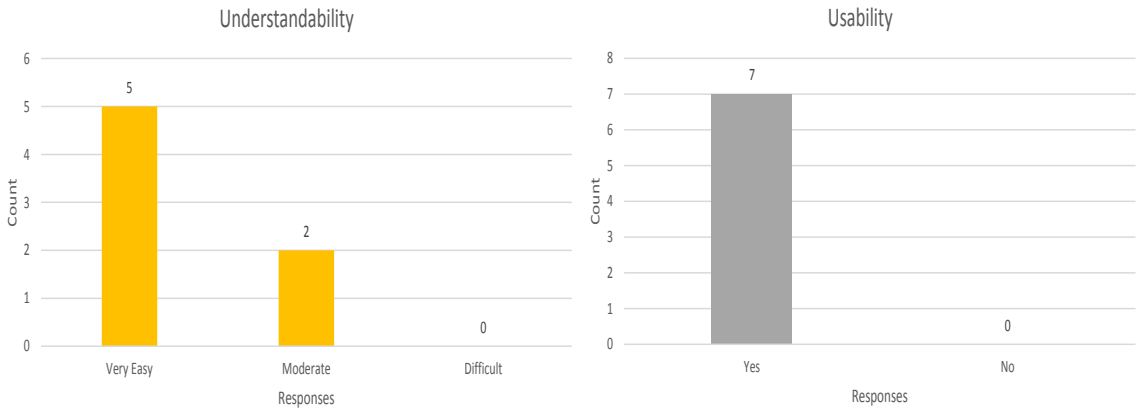
Responses from Experts

From the responses obtained as seen from Figure 5.28a, it was revealed that five data scientists found that the developed tests and approach of performing such tests were very easily understandable. Two experts found it moderately understandable. None of them found it difficult to understand.

Question 5: In your opinion, are the proposed tests and the testing approach useful in establishing a QA phase in your work?

Responses from Experts

Despite few disadvantages mentioned for the previous question, all the data scientists responded that they would like to use these tests as a part of performing QA as depicted from the Figure 5.28b.



(a) Responses for the Understandability aspect

(b) Responses for the Usability aspect

Figure 5.28: Responses from the experts regarding feedback of the proposed testing approach

The key findings from the expert interviews are summarized in the Table 5.9.

Advantages	1. Systematic and clear 2. Good start to consider system testing for data science 3. Saves time and easy to use 4. Graphs and plots helps in analysis
Disadvantages	1. Specific to certain use-case 2. Testing done at an abstract level 3. The results in the report need to be more informative 4. Might provide a false impression about the test coverage
Understandability	The additional tests and the approach are easily understandable
Usability	Data scientists wants to use this approach to perform testing

Table 5.9: Summary of the responses from the feedback interview.

5.3.4 RQ3: Result Summary

To summarize, two test scripts, namely, dataset_evaluation_test and model_quality_test were developed for RQ3. The script dataset_evaluation_test, checked the data to spot any data quality issues. This script included tests to detect 7 defects - class imbalance, signature check, completeness check, duplicates check, leakage, multicollinearity, and dataset shift.

We can detect the defect of class imbalance based on the very high distribution of one target class compared to the other target class. There needs to be a boundary value to indicate that the distribution of one target class is very high. Hence, we used the boundary value analysis technique [Tec15]. Defects like the signature check, completeness check, and duplicates check deals with the structure and syntax of the data. So, the syntax testing technique [Tec15] is used as the test design technique. Experience-based technique [Tec15] was used for defects like leakage, multicollinearity

and dataset shift. As these were extracted based on the experience of the data scientists during our case study for RQ1. We test the data even before the model is developed. So, testing the data was carried before the unit testing phase.

As a next step, the script model_quality_test was executed to identify the defects in the model. Defects like overfitting, underfitting, leakage, class imbalance, dataset shift, calibration, type 1, and type 2 errors were also extracted based on the experience of the data scientists during our case study for RQ1. Therefore, the experience-based technique was considered as the primary test design technique for these defects. All the defects except for calibration error were detected using boundary value analysis supplementing the experience-based testing technique [Tec15].

The system testing phase was employed since we tested the model after it was trained and before it was deployed [Pro13]. By checking the quality of the data and the model, we tested the functionality of the model. This explains the selection of the functional testing type [CD13]. The tests were conducted, using the black box method. We had developed automation scripts to achieve testing. Hence, the testing method was black-box automated testing [Lim09].

Figure 5.29 provides information about the test techniques, methods, phases, and type used to test the defects in the domain of ML.

Test Automation Script	Defects/Errors	Test Technique	Test Method	Test Phase	Test Type
dataset_evaluation_test	Class Imbalance Check	Boundary Value Analysis	Black-box Testing (Automated Testing)	Before Unit Testing (Data Quality Check)	Functional
	Signature Check	Syntax Testing			
	Completeness Check				
	Duplicates Check				
	Leakage Check	Experience-Based Technique			
	Multicollinearity Check				
	Dataset Shift Check				
model_quality_test	Overfitting	Experience-Based Technique Boundary Value Analysis		Sytem Testing	
	Underfitting				
	Leakage				
	Class Imbalance				
	Dataset Shift				
	Type 1 Error				
	Type 2 Error				
	Calibration Error	Experience-Based Technique			

Figure 5.29: Extension of software testing to test the defects related to ML or data science

The practicability of these automated tests was checked based on the three scenarios using the publicly available dataset from the Google-owned online forum for ML, Kaggle [Goo19]. The first scenario showed that the ML model when tested as a black box, without testing the data quality. The results of the model_quality_test showed that the predictions of the model were negatively affected by a certain class of defects. These defects could not be identified when tested in a manner similar to the testing of a traditional software application. In the second scenario, we executed the dataset_evaluation_test script to discover the inconsistencies in the data. After these inconsistencies in the data were handled, the ML model was developed and trained. Next, when the trained model was tested, we observed that the model evaluation metrics improved significantly and the implementation was free from most of the defects when compared with the results in the previous scenario. This explains the reason to conduct data quality and model quality tests. In the last scenario, we extended the application of our tests to a third party ML code implementation, submitted to Kaggle [elm19]. We tested that code implementation with the model_quality_test. We observed from the results that there were some defects in the ML model (third party code

implementation) considered for our evaluation. Therefore, with the additionally developed tests, we can conduct the QA process in the domain of ML just like it is performed in the software engineering domain.

Finally, we conducted feedback interviews to assess the tests we had developed. Seven data scientists assessed the understandability and usability aspects of the test scripts. Majority of the interviewed data scientists responded that the test scripts were easily understandable. All the data scientists participated in the interview responded that the additional tests were useful for them.

6 Conclusion

In this chapter, we summarize the thesis contribution and the overall evaluation results, the challenges faced while deriving the results of the research and possible directions for future work.

6.1 Summary

The master thesis addressed three research questions. First, to identify defects in the ML domain. Secondly, reveal the inadequacy of the software testing aspects to capture such defects and lastly, develop additional tests that would detect the ML related defects. The answers to these questions facilitate in evaluating whether the aspects of software testing like test phases, test techniques, test types [CD13] can be simulated for ML domain. As a start, we gathered the major defects affecting the quality of the results of the ML model. By expert interviews, literature review and a programmatic approach (see Section 5.1) we found that overfitting, underfitting, class imbalance, multicollinearity, leakage, dataset shift, type 1 and type 2 errors, outliers, and calibration errors were the common errors to the domain of ML. Furthermore, we gave the reasons for terming these as "Defects" based on the definitions given in [GVE08] and [Ter90]. This gives us a foundation to assert that software testing can be extended to the domain of ML.

To conclude the RQ2, fourteen data scientists were interviewed. Our first finding is that no formal requirements or specification was used, to conduct necessary tests. This translated into the fact that the majority of the data scientists conducted such tests as unit testing, data quality tests as well as analyzing model evaluation metrics. Few of the interviewed data scientists even performed integration testing as a part of following the QA process. Further, the unavailability of a formal specification posed difficulty for the data scientists from our case study, to validate the defects and the quality of the model. The next finding reveals that although the necessary tests were conducted by the data scientist, there was no standardization. The results of the conducted expert interviews showed that unit and integration tests are insufficient for the identification of the defects, suggested by the results for the RQ1. Data scientists from our case study suggested standardizing the way of evaluating the ML model metrics. As a result, we developed additional tests to detect the defects. Later, we evaluated these tests by simulating the "system testing phase" [Pro13] [Tec15] [CD13] as conducted in traditional software testing.

To accommodate the RQ3, additional tests, namely, "dataset_evaluation_test" and "model_quality_test" were developed to reveal possible defects in the data and the ML model respectively. These tests were considered specifically for the domain of ML. And, these tests simulated the experience-based test design technique [Tec15] [CD13], black-box test method [Lim09] which is automated, system testing phase [Pro13], and the functional testing [Pro13] type in software testing. Using the dataset from "Titanic: Machine Learning from Disaster" competition from Kaggle [Kag19a], we evaluated our developed test scripts. For this, we considered three scenarios. The objective of the first

scenario was to display the defects affecting the quality of the model, neglecting the data quality as a whole. In the second scenario, we evaluated whether the inconsistencies revealed from the data helps us to improve the model quality. Our aim in the third scenario was to consider a third-party code implementation for the ML model which submitted to Kaggle and evaluate the quality of this chosen model. This was to simulate the black-box testing method for the ML domain. The results showed that the two test scripts we had developed for this master-thesis identified the defects in the data and the ML model. We inferred that the quality of a model improved by knowing the inconsistencies in the data beforehand.

The two developed scripts and the approach of conducting such tests were evaluated through feedback interviews with the data scientists from our case study. The responses of the interviews suggest that this approach was easily understandable and can be used to perform QA effectively. Thereby, we extended software testing to the domain of ML.

6.2 Threats to Validity

Scientific research over combining two distinctive domains is a challenging task and the thesis work had several challenges with various aspects.

1. Receiving new data for testing the model

   Testing the ML model with a fresh set of data (testing data) helps to detect the defects in the model and determine the model quality. Since the model was not exposed to that data while generalizing the patterns. But, it is difficult to generate a new set of testing data, having the same features as the data used for training. Hence, we split the entire data considered for the evaluation in the ratio of 80:20 as training and testing data respectively.

2. Selecting appropriate thresholds for defect detection

   Several model evaluation metrics need to be analyzed to detect a defect. There are no requirements or formal specifications outlining the criteria to validate the quality of the model. We can summarize that determining the thresholds to qualify a deviation as a defect is based on the use case a data scientist handles. One cannot conclude that a deviation is a defect. Thus, the tester can mark the test results as "Warning" for the data scientists to analyze the deviations. Data scientists from our case study also suggested that determining whether the deviation is a defect should be left to them.

3. Participant's bias

   The study consisted of expert interviews to answer RQ1, RQ2, and RQ3. The results may be limited to the knowledge and expertise of the participants. Also, some of the interviewed data scientists were not from a software engineering background, hence less information was obtained from them related to the field of software testing. Therefore, the experience, knowledge and the number of experts considered for the study are important. Hence, the responses are subjective to the knowledge of the experts considered for our case study. Furthermore, the sample size of the data scientists considered for the interview is small, thus generalization of the results is hard.

4. Querying results in the keyword search program

   The results obtained by executing the program explained in Section 5.1.3.3, had variations depending on the keywords, which were used for verifying the frequency of occurrence. So the results depend on the keywords considered by us. We speculate that there were discussions in the Kaggle explaining other problems related to the domain of machine results other than the keywords used.

5. Complete automation not achieved

   The testing approach proposed in the thesis was semi-automated, where we rendered the necessary indicators like the model evaluation metrics and relevant graphs in a report. The data scientist analyzes the graphs to detect defects like class imbalance and calibration errors. So it was difficult to achieve complete automation of testing the ML model where we could have indicated a deviation for defects like class imbalance problem and calibration error as well.

Despite such potential limitations, we think that our research approach presented a valid way to extend the software testing onto the domain of ML.

6.3 Future Work

From Chapter 3, we observed that all the research works laid emphasis on carrying out QA for ML. The focus of our thesis was to evaluate the extension of software testing aspects like testing phases, techniques and types to ML. We developed two test automation scripts in this thesis. One script to spot the defects in the data and another to identify defects in the model. This was based on Breck, E et. al [BCN+17] and Braiek [BK18]. However, limited information was provided on software testing principles in the ML domain. And, Breck, E et. al had presented rules to award a score for each test based on which they decided upon deploying the model to production environment [BCN+16] [BCN+17]. This idea did not suit quite well for the thesis. The findings of this master thesis contribute to the research on extending the software testing aspects (like testing phase, techniques, and types) for spotting defects and ensuring the quality of the ML model.

The main scope considered in the thesis was the black-box testing method [Lim09]. The feedback of the data scientists from our case study suggested considering the gray-box testing method [Lim09]. So future research in the direction of comparing the black-box and gray-box testing methods for ML would provide useful insights in establishing a better QA.

The focus of the research was on the binary classification type of ML for a customer analytics use case. The same research can be extended to include the defects for other ML types like multiclass classification, regression, reinforcement learning, etc. under the scope of testing.

The logic for detecting defects related to the domain of ML, as implemented by the test automation script was based on the model evaluation metrics and graphs like the PR curve, AUC-ROC, and calibration plot. The decision to conclude a deviation as a defect relied on the data scientist. A good research direction can be the development of complex ML models or neural networks and embed those in the proposed testing approach to achieve complete test automation with no manual intervention for spotting a defect.

The approach was evaluated using the "Titanic: Machine learning from Disaster" competition published on Kaggle, which is a simple beginner's task. The same testing approach along with defect management tools like ALM[1], Jira[2], etc. can be applied to real-time use cases and evaluate the feasibility of efficient test execution and defect management phase. Thus, we hope that academics continue to explore on software testing for the domain of ML from a research perspective and contribute more towards assuring the quality of the ML applications.

[1]ALM: https://www.microfocus.com/en-us/solutions/software-development-lifecycle
[2]Jira software: https://www.atlassian.com/software/jira

Bibliography

[ABB+19] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, T. Zimmermann. "Software engineering for machine learning: a case study". In: *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice*. IEEE Press. 2019, pp. 291–300 (cit. on p. 15).

[AO16] P. Ammann, J. Offutt. *Introduction to software testing*. Cambridge University Press, 2016 (cit. on p. 31).

[BCN+16] E. Breck, S. Cai, E. Nielsen, M. Salib, D. Sculley. "What's your ML Test Score? A rubric for ML production systems". In: (2016) (cit. on pp. 38–40, 85).

[BCN+17] E. Breck, S. Cai, E. Nielsen, M. Salib, D. Sculley. "The ml test score: A rubric for ml production readiness and technical debt reduction". In: *2017 IEEE International Conference on Big Data (Big Data)*. IEEE. 2017, pp. 1123–1132 (cit. on pp. 38–40, 50, 85).

[Ber17] D. Berrar. "Confidence curves: an alternative to null hypothesis significance testing for the comparison of classifiers". In: *Machine Learning* 106.6 (2017), pp. 911–949 (cit. on p. 58).

[BK18] H. B. Braiek, F. Khomh. "On testing machine learning programs". In: *arXiv preprint arXiv:1812.02257* (2018) (cit. on pp. 15, 40, 43, 49, 62, 85).

[Buc19] Buck Woody. *The Keys to Effective Data Science Projects – Part 9: Testing and Validation*. [Online; accessed 14. Sep. 2019]. Sept. 2019. URL: https://blogs.msdn.microsoft.com/buckwoody/2017/10/24/the-keys-to-effective-data-science-projects-part-9-testing-and-validation (cit. on pp. 15, 16).

[CAS16] P. Covington, J. Adams, E. Sargin. "Deep Neural Networks for YouTube Recommendations". In: *Proceedings of the 10th ACM Conference on Recommender Systems*. New York, NY, USA, 2016 (cit. on p. 23).

[CCY98] T. Y. Chen, S. C. Cheung, S. M. Yiu. *Metamorphic testing: a new approach for generating next test cases*. Tech. rep. Technical Report HKUST-CS98-01, Department of Computer Science, Hong Kong . . ., 1998 (cit. on p. 43).

[CD13] I. Concepts, Definitions. "ISO/IEC/IEEE International Standard - Software and systems engineering –Software testing –Part 1:Concepts and definitions". In: *ISO/IEC/IEEE 29119-1:2013(E)* (Sept. 2013), pp. 1–64. DOI: 10.1109/IEEESTD.2013.6588537 (cit. on pp. 15, 16, 30–34, 39, 43, 58, 62, 80, 83).

[Doc13] I. T. Documentation. "ISO/IEC/IEEE International Standard - Software and systems engineering – Software testing –Part 3: Test documentation". In: *ISO/IEC/IEEE 29119-3:2013(E)* (Sept. 2013), pp. 1–138. DOI: 10.1109/IEEESTD.2013.6588540 (cit. on pp. 32, 62, 69).

[elm19] elmehdichouki. "Using Caret & RandomForest in R for (top 10%)". In: *Kaggle* (Aug. 2019). URL: https://www.kaggle.com/elmehdichouki/using-caret-randomforest-in-r-for-top-10 (cit. on pp. 74, 80).

[Fra14] Francisco J Martin. *A Simple Machine Learning Method to Detect Covariate Shift*. [Online; accessed 24. Sep. 2019]. Jan. 2014. URL: https://blog.bigml.com/2014/01/03/simple-machine-learning-to-detect-covariate-shift (cit. on pp. 65, 66).

[Gér17] A. Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. Ö'Reilly Media, Inc.", 2017 (cit. on pp. 15, 16, 24–27, 43, 50, 58).

[Goo19] Google. *Kaggle: Your Home for Data Science*. [Online; accessed 24. Sep. 2019]. Sept. 2019. URL: https://www.kaggle.com (cit. on pp. 18, 28–30, 80).

[GVE08] D. Graham, E. Van Veenendaal, I. Evans. *Foundations of software testing: ISTQB certification*. Cengage Learning EMEA, 2008 (cit. on pp. 45, 61, 83).

[GYD+08] X. Guo, Y. Yin, C. Dong, G. Yang, G. Zhou. "On the class imbalance problem". In: *2008 Fourth international conference on natural computation*. Vol. 4. IEEE. 2008, pp. 192–201 (cit. on p. 58).

[ISO10] ISO/IEC. *ISO/IEC 25010 System and software quality models*. Tech. rep. 2010 (cit. on pp. 33, 45).

[JAAA16] M. A. Jamil, M. Arif, N. S. A. Abubakar, A. Ahmad. "Software testing techniques: A literature review". In: *2016 6th International Conference on Information and Communication Technology for The Muslim World (ICT4M)*. IEEE. 2016, pp. 177–182 (cit. on p. 15).

[JHR18] A. H. M. Jakaria, M. M. Hossain, M. Rahman. "Smart Weather Forecasting Using Machine Learning: A Case Study in Tennessee". In: Nov. 2018 (cit. on p. 23).

[Kag19a] Kaggle. *Titanic: Machine Learning from Disaster*. [Online; accessed 24. Sep. 2019]. Sept. 2019. URL: https://www.kaggle.com/c/titanic/data (cit. on pp. 63, 64, 83).

[Kag19b] Kaggle Titanic Accuracy. *Titanic: Machine Learning from Disaster | Kaggle*. https://www.kaggle.com/c/titanic/overview/evaluation. (Accessed on 09/26/2019). 2019 (cit. on p. 74).

[KRPS12] S. Kaufman, S. Rosset, C. Perlich, O. Stitelman. "Leakage in data mining: Formulation, detection, and avoidance". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6.4 (2012), p. 15 (cit. on p. 65).

[Lim09] M. G. Limaye. *Software testing*. Tata McGraw-Hill Education, 2009 (cit. on pp. 31, 34, 35, 42, 63, 80, 83, 85).

[LRU14] J. Leskovec, A. Rajaraman, J. D. Ullman. *Mining of massive datasets*. Cambridge university press, 2014 (cit. on pp. 21, 50).

[Mic19] Michael Palmer. *Data is the New Oil*. [Online; accessed 24. Sep. 2019]. Sept. 2019. URL: https://ana.blogs.com/maestros/2006/11/data_is_the_new.html (cit. on p. 21).

[MKA06] C. Murphy, G. E. Kaiser, M. Arias. "A Framework for Quality Assurance of Machine Learning Applications". In: (2006) (cit. on pp. 15, 16, 37).

[MKA07] C. Murphy, G. E. Kaiser, M. Arias. “An approach to software testing of machine learning applications”. In: (2007) (cit. on pp. 37, 40, 62).

[Nak18] S. Nakajima. “Quality assurance of machine learning software”. In: *2018 IEEE 7th Global Conference on Consumer Electronics (GCCE)*. IEEE. 2018, pp. 601–604 (cit. on pp. 39, 40).

[NMOU18] Y. Nishi, S. Masuda, H. Ogawa, K. Uetsuki. “A test architecture for machine learning product”. In: *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE. 2018, pp. 273–278 (cit. on pp. 39, 40).

[OS13] C. O’Neil, R. Schutt. *Doing data science: Straight talk from the frontline*. Ö’Reilly Media, Inc.", 2013 (cit. on pp. 21, 22, 24, 41, 43, 50, 58, 65, 66).

[Pau06] R. K. Paul. “Multicollinearity: Causes, effects and remedies”. In: *IASRI, New Delhi* (2006) (cit. on pp. 58, 65).

[PF13] F. Provost, T. Fawcett. *Data Science for Business: What you need to know about data mining and data-analytic thinking*. Ö’Reilly Media, Inc.", 2013 (cit. on pp. 21, 62, 66).

[Pro13] I. T. Process. “ISO/IEC/IEEE International Standard - Software and systems engineering –Software testing –Part 2:Test processes”. In: *ISO/IEC/IEEE 29119-2:2013(E)* (Sept. 2013), pp. 1–68. DOI: 10.1109/IEEESTD.2013.6588543 (cit. on pp. 17, 31, 32, 58, 61–63, 80, 83).

[QSSL09] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, N. D. Lawrence. *Dataset shift in machine learning*. The MIT Press, 2009 (cit. on pp. 58, 66).

[Sam00] A. Samuel. “Some Studies in Machine Learning Using the Game of Checkers. II—Recent Progress”. In: *IBM Journal of Research and Development* 3 (Feb. 2000), pp. 206–226. DOI: 10.1147/rd.441.0206 (cit. on p. 24).

[Sea99] C. B. Seaman. “Qualitative Methods in Empirical Studies of Software Engineering”. In: *IEEE Trans. Software Eng.* 25 (1999), pp. 557–572 (cit. on p. 17).

[SKM13] B. R. Sharma, D. Kaur, A. Manju. “Review on Data Mining: Its Challenges, Issues and Applications”. In: *International Journal of Current Engineering and Technology* 3.2 (2013) (cit. on pp. 50, 58).

[SR13] P. M. Shields, N. Rangarajan. *A playbook for research methods: Integrating conceptual frameworks and project management*. New Forums Press, 2013 (cit. on p. 17).

[SR15] T. Saito, M. Rehmsmeier. “The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets”. In: *PloS one* 10.3 (2015), e0118432 (cit. on p. 66).

[Sta19] Stack Exchange. *Covariate shift detection*. [Online; accessed 24. Sep. 2019]. Sept. 2019. URL: https://datascience.stackexchange.com/questions/8278/covariate-shift-detection (cit. on p. 65).

[Ste15] Stephanie Glen. *Multicollinearity: Definition, Causes, Examples - Statistics How To*. [Online; accessed 24. Sep. 2019]. Sept. 2015. URL: https://www.statisticshowto.datasciencecentral.com/multicollinearity (cit. on pp. 58, 65).

[Tec15] I. T. Techniques. "ISO/IEC/IEEE International Standard - Software and systems engineering–Software testing–Part 4: Test techniques". In: *ISO/IEC/IEEE 29119-4:2015* (Dec. 2015), pp. 1–149. DOI: 10.1109/IEEESTD.2015.7346375 (cit. on pp. 17, 32, 33, 58, 62, 63, 69, 79, 80, 83).

[Ter90] I. S. E. Terminology. "IEEE Standard Glossary of Software Engineering Terminology". In: *IEEE Std 610.12-1990* (Dec. 1990), pp. 1–84. DOI: 10.1109/IEEESTD.1990.101064 (cit. on pp. 45, 83).

[TFL18] X. Tong, Y. Feng, J. J. Li. "Neyman-Pearson classification algorithms and NP receiver operating characteristics". In: *Science advances* 4.2 (2018), eaao1659 (cit. on p. 58).

[Wic16] N. Wickramage. "Quality assurance for data science: Making data science more scientific through engaging scientific method". In: *2016 Future Technologies Conference (FTC)*. IEEE. 2016, pp. 307–309 (cit. on pp. 38, 40).

[Yur18] B. Yurdakul. "Statistical Properties of Population Stability Index". In: (2018) (cit. on p. 65).

[Zhe15] A. Zheng. "Evaluating machine learning models: a beginner's guide to key concepts and pitfalls". In: (2015) (cit. on pp. 43, 50, 58, 65, 66).

All links were last followed on September 25, 2019.

A Appendix

A.1 Python script making API calls to Kaggle

```
# import libraries
data_list = []

for i in range(<start_number>, <end_number>):
        # sending requests to kaggle
        data = requests.get("https://www.kaggle.com/topics/"+str(i)+".json")
        print(str(i)+":"+str(data.status_code))
        time.sleep(0.5)
        if data.status_code == 200:
                try:
                        # collect data
                        json_data = data.json()
                except ValueError:
                        print("Wrong status code for topic: "+str(i))
                else:
                        data_list.append(json_data)


with open("<file location to store the collected data>", 'w') as outfile:
        json.dump(data_list, outfile)
```

A.2 Python script to structuring the discussion data of Kaggle

```
# importing libraries
import json
import os

from utils import getItemDetails

if __name__ == '__main__':
        # obtaining the files from the path
        file_path = # <mention file_path>
        json_files = os.listdir(file_path)
        result = []

        for json_file in json_files:
                print(json_file)

                with open(file_path + json_file) as fp:
                        json_array = json.load(fp)

                        for elements in json_array:
                                # code for data extracting
                                item_details = getItemDetails(elements)
```

```
                                print(type(item_details))
                                data_json = {"id":item_details.item_id,
                                        "competition": item_details.competition,
                                        "author": item_details.name_author,
                                        "date_time": item_details.datentime,
                                        "overview": item_details.overview,
                                        "comments": item_details.comments,
                                        "votes": item_details.votes}

                                result.append(data_json)

                with open("data_merge.json", "w") as json_elements:
                        json.dump(result, json_elements)
```

A.3 Frequency of Keyword Occurrence - code snippets

A.3.1 Loading data to Microsoft Azure

```
## Loading the filtered discussion data

# File location and type
file_location = "/FileStore/tables/data_merge.json"
file_type = "json"

# CSV options
infer_schema = "false"
first_row_is_header = "false"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

display(df)
```

A.3.2 Merging the text data

```
df_combine = df.select(concat(df['overview'], lit(' '), df['comments']).alias('combine_text'))
display(df_combine)
```

A.3.3 Cleaning the text data

```
def removePunctuation(column):
    """Removes punctuation, changes to lower case, and strips leading and trailing spaces.

    Note:
        Only spaces, letters, and numbers should be retained.  Other characters should should be
        eliminated (e.g. it's becomes its).  Leading and trailing spaces should be removed after
        punctuation is removed.

    Args:
        column (Column): A Column containing a sentence.
```

```

    Returns:
        Column: A Column named 'sentence' with clean-up operations applied.
    """

    regex_result = regexp_replace(column,r'<.*?>|&([a-z0-9]+|#[0-9]{1,6}|#x[0-9a-f]{1,6});', '')
        #regexp_replace(str, pattern, replacement)
    trim_result = trim(regex_result)  #Trim the spaces from both ends for the specified string
        column.
    lower_result = lower(trim_result) #Converts a string column to lower case.
    return lower_result
```

A.3.4 Querying the keywords of defects

```
regexTokenizer = RegexTokenizer(inputCol = "refined_text", outputCol = "words", pattern =
    "overfit|underfit|missing values|imbalance|covariate shift|outlier|leakage|calibration|dataset
    shift|drift", gaps=False)
```

A.3.5 Aggregating the results

```
wokey = countdf[countdf['size'] == 0].count()
wkey = countdf[countdf['size'] != 0].count()

result1_DF = [Row(Discussions = 'Discussions without defect keywords', Count = wokey),
    Row(Discussions = 'Discussions about defects', Count = wkey)]

schema = StructType([StructField('Discussions', StringType()), StructField('Count',IntegerType())])
result1 = spark.createDataFrame(result1_DF, schema)

display(result1)

schemao = StructType([StructField('keys_wodupes', StringType()),
    StructField('count_wodupes',IntegerType())])
result2 = spark.createDataFrame(countsdupes, schemao)

display(result2)
```

A.4 Test Procedure Form

A.4.1 Scenario 1: Initial Functional Test

Test Procedure				
Test Procedure ID	**Objective**			**Estimated Duration**
S-1	The purpose of this test procedure is to test the way the ML model provides the results on a testing data when realized as a black box and check for the all possible defects in the model.			
Start up: The stages of preprocessing, feature engineering and modelling is performed as a pre-requisite. We have the ML model which is trained with the training data.				
Relationships to other procedures: None				
Test Log				
Date:	Initials:	Test item:		Ok/Not Ok
9/2/19		System Testing of ML model.		
Comments:				
Procedure				
Step no.	**Activities**	**Examination of result**	**Actual results**	**Test result**
1	Validate Model functionality.	The trained ML model predicts when applied on the testing data without any failure.	The trained model predicts without any failure.	**Pass**
2	Validate the format of the prediction results.	The trained ML model predicts the results in the form of "0"s and "1"s.	The results are in the form of "0"s and "1"s.	**Pass**
3	Verify for Overfitting.	The model is not overfitted.	The model is not overfitted. But the train and test accuracies are very low	**Warning**
4	Verify for Underfitting.	The model is not underfitted.	The model is not underfitted. But the train accuracy is very low	**Warning**
5	Verify for Leakage.	The model does not suffer from Leakage.	The model is not suffering from Leakage.	**Pass**
6	Verify for Class imbalance.	The model does not suffer from Class imbalance.	The model is not suffering from Class imbalance.	**Pass**
7	Verify for Calibration errors.	The model does not suffer from Calibration errors.	The model results are moderately calibrated.	**Warning**
8	Verify for Dataset shift.	The model does not suffer from Dataset shift.	The model is not suffering from Dataset shift.	**Pass**
9	Verify for Type 1 errors.	The model does not suffer from Type 1 errors.	The model suffers from Type 1 errors.	**Warning**
10	Verify for Type 2 errors.	The model does not suffer from Type 2 errors.	The model suffers from Type 2 errors.	**Warning**
Stop: End of testing				

Figure A.1: Test Procedure Form of Scenario 1

A.4.2 Scenario 2: Data and Model Quality Test

Test Procedure				
Test Procedure ID	**Objective**			**Estimated Duration**
S-2	The purpose of this test procedure is to test the dataset provided to analyze the possible errors and thereby construct the model and then again check the model for possible errors on testing data when realized as a black-box.			
Start up: 1: For dataset_evaluation_test: Data is provided. 2. For model_quality_test: The stages of preprocessing, feature engineering and modelling is performed as a pre-requisite. We have the ML model which is trained with the training data.				
Relationships to other procedures: None				
Test Log				
Date:	Initials:	Test item:		Ok/Not Ok
9/3/19		Testing data and model.		
Comments:				
Procedure				
Step no.	**Activities**	**Examination of result**	**Actual results**	**Test result**
1	Check the data for imbalance class problem.	The data provided should be free from imbalance class problem.	No class imbalance problem	**Pass**
2	Check the data for any problems concerning signature checks.	The data provided should not have any irregularities in terms of structure.	No problem in signature checks.	**Pass**
3	Check the data for any missing data.	The dataset provided should not contain any missing values.	"Age" column has 139 "NA" values. And, "Cabin" column has 553 "empty" values respectively.	**Warning**
4	Check the dataset for any duplicated data.	The dataset provided should contain any duplicated data.	No duplicated rows.	**Pass**
5	Check the size of the dataset.	The dataset provided should be large enough.	Dataset is comparitively large.	**Pass**
6	Check the data for any possibility of leakage.	The data provided should be free from leakage	Possibility of target leakage due to columns "Name" and "Ticket".	**Warning**
7	Check the data for any problem concerning multicollinearity.	The data provided should be free from multicollinearity problems.	Dataset is free from multicollinearity problems.	**Pass**
8	Check the labels in the data for any possibility of dataset shift.	The labels in the data are free from dataset shift.	10 labels show the possibility of dataset shift.	**Warning**
9	Validate Model functionality.	The trained ML model predicts when applied on the testing data without any failure.	The trained model predicts without any failure.	**Pass**
10	Validate the format of the prediction results.	The trained ML model predicts the results in the form of "0"s and "1"s.	The results are in the form of "0"s and "1"s.	**Pass**
11	Verify for Overfitting.	The model is not overfitted.	The model is not overfitted. Better train and test accuracies are observed.	**Pass**
12	Verify for Underfitting.	The model is not underfitted.	The model is not underfitted. Better train accuracy is observed.	**Pass**
13	Verify for Leakage.	The model does not suffer from Leakage.	The model is not suffering from Leakage.	**Pass**
14	Verify for Class imbalance.	The model does not suffer from Class imbalance.	The model is not suffering from Class imbalance.	**Pass**
15	Verify for Calibration errors.	The model does not suffer from Calibration errors.	The model results are well calibrated.	**Warning**
16	Verify for Dataset shift.	The model does not suffer from Dataset shift.	The model is not suffering from Dataset shift.	**Pass**
17	Verify for Type 1 errors.	The model does not suffer from Type 1 errors.	The model is not suffering from Type 1 errors.	**Pass**
18	Verify for Type 2 errors.	The model does not suffer from Type 2 errors.	The model is not suffering from Type 2 errors.	**Pass**
Stop: End of testing				

Figure A.2: Test Procedure Form of Scenario 2

A.4.3 Scenario 3: Testing the Third Party Code

Test Procedure				
Test Procedure ID	**Objective**			**Estimated Duration**
S-3	The purpose of this test procedure is to consider a ML code submitted to kaggle for "Titanic: Machine learning from Disaster" competition and execute "model_quality_test". Then, we realize the quality of the model by checking for defects.			
Start up: The stages of preprocessing, feature engineering and modelling is performed as a pre-requisite. We have the ML model which is trained with the training data.				
Relationships to other procedures: None				
Test Log				
Date:	Initials:	Test item:		Ok/Not Ok
9/3/19		Testing the model.		
Comments:				
Procedure				
Step no.	**Activities**	**Examination of result**	**Actual results**	**Test result**
1	Validate Model functionality.	The trained ML model predicts when applied on the testing data without any failure.	The trained model predicts without any failure.	**Pass**
2	Validate the format of the prediction results.	The trained ML model predicts the results in the form of "0"s and "1"s.	The results are in the form of "0"s and "1"s.	**Pass**
3	Verify for Overfitting.	The model is not overfitted.	The model may be overfitted.	**Warning**
4	Verify for Underfitting.	The model is not underfitted.	The model is not underfitted.	**Pass**
5	Verify for Leakage.	The model does not suffer from Leakage.	The model is not suffering from Leakage.	**Pass**
6	Verify for Class imbalance.	The model does not suffer from Class imbalance.	The model is not suffering from Class imbalance.	**Pass**
7	Verify for Calibration errors.	The model does not suffer from Calibration errors.	The model results are not well calibrated.	**Warning**
8	Verify for Dataset shift.	The model does not suffer from Dataset shift.	The model is not suffering from Dataset shift.	**Pass**
9	Verify for Type 1 errors.	The model does not suffer from Type 1 errors.	The model is not suffering from Type 1 errors.	**Pass**
10	Verify for Type 2 errors.	The model does not suffer from Type 2 errors.	The model suffers from Type 2 errors.	**Warning**
11	Verify the accuracy	The accuracy is "80.382" as scored by kaggle.	The test accuracy is 81.08 and the balanced test accuracy is 79.09.	**Warning**

Figure A.3: Test Procedure Form of Scenario 3

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature