



Final report

CM3203 – One Semester Individual Project – 40 Credits

## Predicting Driver Insurance Claim

Author – Seb Squire

Supervisor – Yuhua Li

Moderator – Alun D Preece

## Abstract

Many automotive insurance providers are looking to improve their service for their customers, businesses are starting to adapt and implement machine learning and artificial intelligence methods of analysing data for performance, as a result giving better service for their customers from a better understanding of their needs. The main focus of this project therefore is targeted at automotive insurance providers looking to implement machine learning into their business, the project would also be beneficial to stakeholders and those who are looking to apply machine learning to improve their business.

Many businesses (in this case Porto Seguro) are looking into Kaggle competition platforms in order to gather alternative third party approaches for the hope of producing less simplistic models, this project aims to look at how machine learning concepts can be applied to insurance data, in particular for what makes a driver a risk for becoming insured (buying a new vehicle policy) with an insurance provider. Important key terms have been highlighted primarily in the background research, so there are definitions which are simple to find later on in the reading if they happen to be needed.

Alternate motivations for this project and what to expect include: working with anonymized data and further applications of machine learning, as well as how to handle imbalanced datasets with cross validation and sampling methods.

## Acknowledgements

I would like to express special thanks firstly to my supervisor Dr Yuhua Li, whose continuous support and expertise helped guide me throughout the duration of the project. I would also like to thank Porto Seguro for providing a real and reliable insurance claims dataset, as well as anonymizing the data for ethical reasoning. Lastly a thank you to my family and friends for providing me with insights to certain ideas, and giving support as a whole.

# Contents

---

## List of Figures

### 1. Introduction

#### 1.1.Preface

#### 1.2.Intended Audience and Ethical Concerns

#### 1.3.Project Aims and Scope

### 2. Background research

#### 2.1.Porto Seguro, the Problem and Potential Stakeholders

#### 2.2.The Wider Context

##### 2.2.1. Relevant Aspects of Porto Seguro

##### 2.2.2. What Could Be Beneficial to Porto Seguro or Stakeholders

#### 2.3.Constraints on the Approach

#### 2.4.Factors for Driver's Insurance Risk

#### 2.5.Machine Learning

##### 2.5.1. Supervised Learning Vs Alternative Algorithms

#### 2.6.Machine Learning Approach to Predicting Driver's Risk

2.6.1. Decision Trees

2.6.2. Naïve Bayes

2.6.3. Logistic Regression

2.6.4. Ensemble Methods

2.7. Evaluating the Model

2.7.1. Generalisation, Overfitting and Underfitting

2.7.2. Evaluation Metrics

2.8. Python Machine Learning Libraries

Contents

---

2.9. Existing Solutions

3. Design Specification & Approach

3.1. Business Model and Dynamic Behaviour

3.2. Requirements Specification

3.3. System Architecture

3.4. System Design Approach and Flow

3.5. Development Strategy and Methodology

4. Implementation

4.1. Understanding the Porto Seguro Dataset

4.2. The Algorithm Structure

4.3. Data Pre-Processing and Preparation

4.3.1. Loading and Visualising the Data

4.3.2. Feature Extraction

#### 4.3.3. Feature Selection and Dimension Reduction

#### 4.4. Cross-Validation and Imbalance Learning

#### 4.5. Classification and Evaluation for Models

#### 4.6. Fine Tuning the Models

### 5. Results & Evaluation

#### 5.1. Satisfying Pre-Processing Requirements

#### 5.2. Individual Classifier Performance

##### 5.2.1. Random Forest Classifier

##### 5.2.2. XGBoost Classifier

#### Contents

---

##### 5.2.3. Logistic Regression Classifier

#### 5.3. Testing Generalisation

#### 5.4. Validating the Chosen Classifier

#### 5.5. Conclusion of Results

### 6. Future Work

### 7. Conclusion

### 8. Project Reflection

### 9. References

## List of Figures

---

- 2.1 Machine learning Functionality [9]
- 2.2 Email Filter Classifier [17]
- 2.3 Fruit cluster [20]
- 2.4 Decision tree for a driver's risk
- 2.5 Naïve Bayes network structure [23]
- 2.6 Bernoulli probability density function [24]
- 2.7 Logistic regression predicting labels [24]
- 2.8 Ensemble Methodology [26]
- 2.9 Overfitting a Model [28]
- 2.10 Underfitting a Model [28]
- 2.11 Generalization of data [9]
- 2.12 Confusion matrix

2.13 Evaluation metrics: Accuracy, Precision & Recall [28]

2.14 Receiver operating characteristics graph (ROC) [30]

3.1 Machine learning pipeline for generating quotes

3.2 System architecture and directory diagram

3.3 System design flow-chart stage one

3.4 System design flow-chart stage two

3.5 System design flow-chart stage three

3.6 The agile methodology for software development (Water-Scrum-Fall)

4.1 Loading training and testing datasets

4.2 Visualising regional, calculated and car features

4.3 Missing data from train.csv and test.csv

4.4 Filling in missing feature values

4.5 One hot encoding categorical features

4.6 Pearson correlation coefficient of features

4.7 Removing 'calc' features

4.8 Calculating Gini scores for each feature individually

4.9 A histogram of the imbalanced distribution of target values

4.10 Cross validation with random over sampling code

4.11 Code to produce classifications

4.12 XGBoost classifier parameters

5.1 Output of function df.head() loading the training and test datasets

5.2 Evidence of training data missing values being imputed



- 5.3 Evidence of testing data missing values being imputed
- 5.4 Evidence of one-hot encoded features
- 5.5 Initial random forest classifier confusion matrix
- 5.6 Basic random forest Gini and auc score
- 5.7 Final random forest Gini and auc scores
- 5.8 Initial XGBoost classifier confusion matrix
- 5.9 Final XGBoost Gini and auc scores
- 5.10 Final logistic regression Gini and auc scores
- 5.11 Confusion matrix for a simple random forest classifier
- 5.12 First public and private test data Gini scores
- 5.13 Cross validation and random sampling results
- 5.14 Second public and private test data Gini scores
- 5.15 A roc graph for the random forest classifier
- 5.16 A roc graph for the XGBoost classifier

---

## CHAPTER 1

---

---

# Introduction

---

## 1.1. Preface

Vehicle owners seek out automotive insurance companies for insurance so that in the unfortunate event of an accident, they can mitigate the costs involved with coverage for: property (damage or theft to a car), liability (legal responsibility to others for medical or property) and medical (treating injuries) [1].

Insurance claims occur when the policyholder (the customer) creates a formal request to an insurer for coverage or compensation for an accident. The insurance company must validate this request and then decide whether or not to issue payment to the policyholder.

Automotive insurance quotes are determined by several factors and these factors can establish how much a driver will have to pay for their insurance contract. Some common examples [2] can be credit history, as there is research suggesting individuals with lower credit scores are more likely to file claims and potentially commit fraud or miss payments, this consequently provides a large problem financially for the insurance firm.

Another important example would be the drivers' location [2] due to evidence indicating areas which are highly populated with a lot of congestion tend to have accidents occurring more frequently, which then leads to a claim being made. This can raise the price of insurance for the customer significantly and you could argue from this that it would be unfair for a good driver to have to pay more just because of where they live, this creates a problem for the customer because if the insurance price is raised they may not be able to afford it and subsequently affects the insurance firm as they are losing potential customers.

Given these factors and their impacts on the insurance firm this creates a problem where there needs to be an efficient method that can determine the risk a driver poses to a company, in order for the firms to be able to adjust the insurance prices fairly to a drivers' ability and relevant personal information making automotive insurance more accessible to drivers, but also consider the insurance firm not losing money.

## Chapter 1. Introduction

---

Traditional methods for generating quotes would simply use these factors as variables and calculate the likelihood the driver would make a claim; a lot of insurance firms are adopting machine learning approaches that implement the traditional approach in a more detailed way which provides a more accurate and representative result. For this project I plan on exploring more

effective techniques in machine learning to try and produce as most an accurate prediction as possible for a drivers' risk level.

## 1.2. Intended Audience and Ethical Concerns

The intended audience and beneficiaries to come from this project would be individuals or organisations, which are interested in applying machine learning algorithms for predicting potential outcomes from automotive insurance data, with regards to the customers' personal information. More specifically, for determining the risk of a driver to be insured for an automotive insurance company, based on the machine learning model primarily factoring in whether or not a customer has requested an insurance claim in the past.

In this project I will be analysing a dataset provided by Porto Seguro, which is a large automotive company based in Brazil. In order to ensure that Porto Seguro customers' personal and private information is kept secure and secret, the dataset given by this company has been anonymized [3]. Certain data such as categorical data can be identified in the columns of the dataset by for example 'ps\_car\_04\_cat', which can give a generic indication of the expected data referring to the car such as vehicle type or car use (commercial or private), however this is not explicitly stated for protective purposes.

Lastly I have signed an agreement which means I cannot re-distribute the data and I can only use this data for academic research and non-commercial purposes.

## 1.3. Project Aims and Scope

The main overall scope for this project focuses on creating a machine learning algorithm which will provide an automotive insurance prediction as accurately as possible, in order for the client Porto Seguro to be able to improve their business by understanding their customers' information in more depth, as well as how they can use this information in their data science teams to tailor insurance prices for the better for their customers.

The main overall aims of this project are to firstly be able to generate an accurate prediction for whether or not a potential automotive insurance customer is likely to file an insurance claim. Secondly there must therefore be an accurate machine learning algorithm and model in order to produce suitable predictions, the basis of the model is that it should effectively factor in customers' information such as 'vehicle type' or 'cost of car', by doing so the predictions given by these features should then be representative to common statistical knowledge that refer to insurance claims, and as a result indicate which features are more likely to cause an insurance claim.

From the results of the model (and provided that the prediction is accurate), it should enable Porto Seguro to be able to make automotive insurance coverage more accessible to more drivers, this can be achieved by using the prediction to accommodate more reasonably affordable insurance prices, according to the drivers' risk assessment level and the cost of their road vehicle.

Lastly, given an accurate insurance prediction it will support Porto Seguro in tailoring insurance quotes according to the drivers' ability. Suppose that a driver has a clean driving record, it would be unfair for them to have to pay a similar insurance fee with a driver with a bad driving history. The model should therefore make it clearer which drivers are unlikely to make a claim and adjust their insurance quote lower, as well as accordingly increase the insurance cost for those who are likely to make a claim.

---

## CHAPTER 2

---

---

### Background Research

---

This section will detail a literature review to outline the context for this research project. It will specify information about the client as well as potential stakeholders and how this research is relevant to them, it will also provide information on how they could further build upon their current business components in the background of insurance predictions. This chapter will also go into further detail regarding the factors that affect car insurance quotes and their significance, then determining the most appropriate machine learning model to apply for predicting a drivers' risk level for becoming insured. Next is a brief description on the python libraries used and their purpose, then a description of existing solutions to distinguish the difference for the purpose of this research project in particular.

## 2.1. Porto Seguro, the Problem and Potential Stakeholders

The basis of this research project is based on the requirements for Porto Seguro [3], this however does not limit the possibilities and potential to apply the machine learning concepts to different areas in automotive insurance (see 2.2. The Wider Context). There can be a various amount of potential stakeholders which would primarily be other insurance providers looking to apply similar machine learning concepts for the same general purpose of tailoring quotes, or trying to understand their customers' information in a better way.

As described previously in the introduction (see 1.2. Intended Audience and Ethical Concerns) Porto Seguro is one of Brazil's largest automotive and homeowner insurance companies. Their aim for their automotive division is to tailor insurance quotes according to drivers' abilities, this can be accomplished through machine learning to learn from the customer information that they have been gathering over numerous years. Porto Seguro have been implementing machine learning into their business, although they believe that there may be more effective techniques that can be applied for more accurate results. Due to this they have provided two anonymized datasets, a training set and a test set in the expectation a better model can be created compared to their current model.

To then further elaborate from the preface, there is a need for a model which can read and interpret from Porto Seguro's large datasets containing customer information in the thousands. This creates a problem where there needs to be a suitable method and a more accurate machine learning model to determine the risk a driver poses to an insurance company and the probability they are going to produce a claim in the next year.

## 2.2. The Wider Context

### 2.2.1 Relevant Aspects of Porto Seguro

Porto Seguro is a company which operates in many lines of insurance [48] including: automobile, business, company health, investments, financing, residence services and telecommunications. This project will be looking specifically at the automobile insurance sector; however, this does not limit the machine learning concepts to just analysing automobile insurance. This project will hopefully give insight into how you can apply similar machine learning concepts to a range of different possibilities, given that you have access to a reasonably large enough dataset.

The aspects of Porto Seguro which are the most relevant for this project would be 'vehicle insurance' and 'automobile consortium'. The vehicle insurance department is simply automobile insurance coverage for protecting a vehicle. Porto Seguro have described their ethics to be that they diversify their packages or products considerably of sex or age, they also try to be mindful of vehicle types in a fair manner. With this in mind, the dataset provided by the company does anonymize the feature or column names so it is not possible to directly analyse sex or age, although the results from the machine learning solution would indicate to Porto Seguro themselves how they can compare their user data to common driver statistics.

Unfortunately, there is a lack of driving statistics in Portugal as well as Europe with the exception of the United Kingdom which provides a large amount of information. Due to this there will have to be an assumption made that the United Kingdom statistics can be relatively representative of Europe including Portugal itself, where Porto Seguro operates.

Confused.com are an insurance quote comparison provider, they have announced their driving statistics [4] for the United Kingdom, which tend to suggest women are safer drivers than men. One example in particular is that males made sixty-seven percent of accidents and insurance claims, and in general for the 'at fault claims' men are double as likely to be at fault. It is also indicated that males are more likely to make theft claims. This type of information can additionally become apparent through a machine learning solution and become very relevant for Porto Seguro to be aware of, as this knowledge can help their business and how they model their insurance quotes in future for fair results for their customers, this can be done through predictive modelling.

### 2.2.2 What Could Be Beneficial to Porto Seguro or Stakeholders?

There are many beneficial applications of a good machine learning model, as stated in [5] where there is a large range of data science use cases in insurance models that can be applied to. The main purpose of this project is to determine the risk a driver poses to the insurer and from this the insurer can tailor prices, therefore a model to actually predict a reasonable insurance quote estimate can be produced and become an upstream component in their business pipeline (as seen in figure 1). It is also additionally advantageous to the insurer to seek areas where they can use models to improve their business further, and understand the advantages of machine learning.

To expand on the fraud detection use case in [5] there are motivations to use machine learning algorithms for fraud detection and prevention [6]. An unsupervised learning algorithm could be used to examine data that does not contain identified fraud, and as a result reveals new irregularities and patterns from user data that identifies fraud and can be used to prevent and provide wariness of insuring potential fraudulent people. A supervised learning algorithm could be used to learn from historical data containing both fraud and non-fraud cases to identify patterns that an investigator at Porto Seguro may want to flag. It is also possible to implement a graph database [7] together with a machine learning model which can allow potential stakeholders or

the developers at Porto Seguro to more visibly see relationships between their customers and their information for clearer fraud analysis.

## Chapter 2. Background Research

---

### 2.3. Constraints on the Approach

The dataset which Porto Seguro are providing [3] has been **anonymized**, this means that there is no clear way of knowing what the features (such as location) represent specifically, they have however given certain categories of features [3]. In terms of aims and objectives It is not possible for me to directly correlate a feature such as ‘vehicle colour’ to provide information to Porto Seguro, however It is possible to correlate the feature to its category and make assumptions, regardless Porto Seguro can easily use the model and understand the data themselves.

It is also important to note that when customers fill in data for new policies, they tend to not always fill in all information [3], therefore the solution will have to accommodate to **missing values** which could have been informative for the learning algorithms performance.

### 2.4. Factors for Driver’s Insurance Risk

As previously stated, Porto Seguro split their data into three categories: region, individual and vehicle information. It is useful to have an abstract view of how these categories can affect car insurance premiums, this will make it easier to interpret the dataset and perform feature engineering more easily. ‘Compare the Market’ [8] are a car insurance premium comparison website for the UK, the information provided is therefore on the UK however

It can also be representative of Portugal and other countries in Europe. They suggest some factors are correlated firstly with **age**, young drivers tend to pay more as they are not as experienced as older drivers [8]. The policy holders **location** or postcode can have a large impact as more populated areas (cities) are more liable to theft and vandalism, therefore increasing the premium. Another factor can be a policy holders **occupation**, some are considered more high risk than others, for example a labourer compared to a secretary [8].



Other than age, location and occupation there are also other factors, some of which are more controversial. A good example is **gender**, as described in source [2] certain insurers in the United States adjust their premium rates based on gender due to the accident

## Chapter 2. Background Research

---

statistics associated (IIHS) [2]. Factors like **vehicle type** and **vehicle use** can have an increased premium as some cars have higher theft statistics, and those who drive more mileage are more likely to get into an accident leading to a higher premium [2].

To summarise, the individual dataset category is highly likely to include factors such as: gender, age and occupation. Region can be linked with location and vehicle category can be linked with vehicle type and use.

### 2.5. Machine Learning

Machine learning is the science of providing computers the ability to learn from data without being explicitly programmed [9]. It is a method of data analysis which has been increasing highly in demand over the last decade, as it enables individuals and organisations the ability to understand their datasets in further detail effectively. Forbes research even suggests that one in ten enterprises now use ten or more AI applications, some examples of where algorithms are effective include: 21% of applications focusing on fraud analysis, 26% on process optimisation and 12% for opinion-mining [10]. Machine learning is an extension of artificial intelligence and from the use of logic and conditions, it can enable machines to improve at certain tasks with experience, by learning from the data and identifying patterns of interest with minimal human intervention [11]. From the use of learning algorithms it is possible to produce a **prediction** based on a target value, or uncover underlying information previously unseen in the data with regards to something in the world [12].

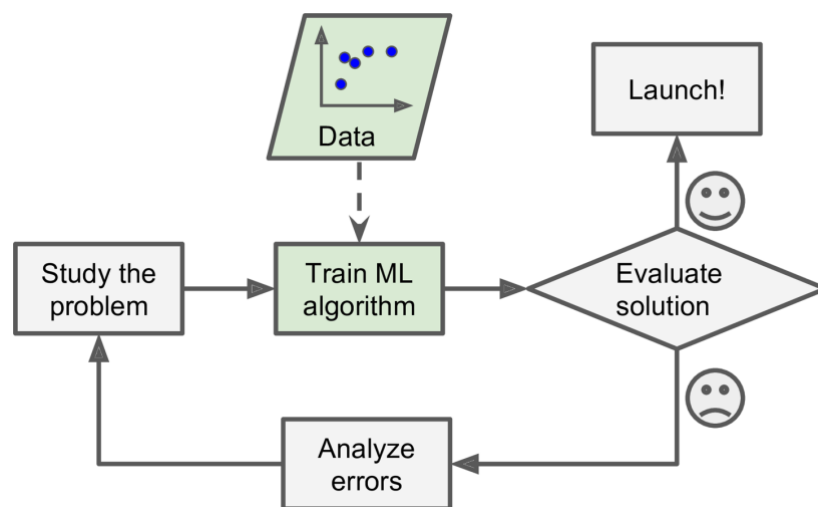
There are a large variety of uses for applying machine learning models in industries, some examples include the use of predictive models for product recommendations for online shopping, as well as fraud detection in banking or even spam filtering in email inboxes [13]. The

fundamental concept however behind these applications, is that the model needs to be able to generalise well in for it to produce accurate predictions. **Generalizing** provides a measure of how well your solution performs on unseen cases [14], for example how accurately a case of fraud is identified as being correct or not.

## Chapter 2. Background Research

---

There are also various models and approaches of learning algorithms, **supervised learning** and **unsupervised learning** are only a couple examples. These learning algorithms train the data to provide specific outputs dependent on the purpose of the task, this could be **regression** [15] that handles continuous outputs (perhaps predicting a housing property estimate) or **classification** [16] which handles categorical outputs, or discrete and unordered values (perhaps predicting a type of food category). The overall functionality and flow of machine learning can be represented as the following:



*Figure 2.1 - Machine Learning Functionality*

The generic machine learning algorithm will: receive input data, the data is trained through the model to make predictions, to determine its ability to generalize to external data it tests the model on the test data to determine if it predicts correctly or not, lastly there is an output of the prediction.

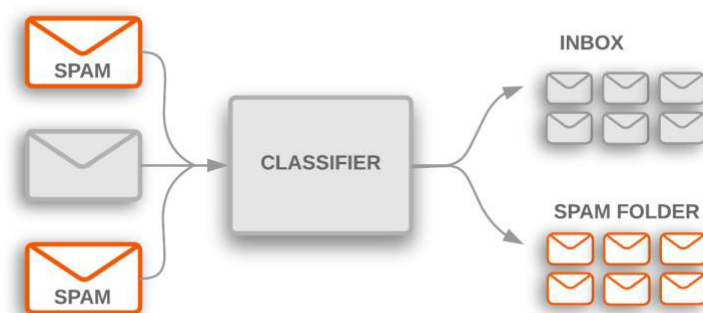
## 2.5.1 Supervised Learning Vs Alternative Algorithms

In **supervised learning**, the input training data includes instances with known **labels** [9], these labels represent the desired correct output. Typically supervised learning uses a classifier model, for example to train an email spam filter, this would train specific emails with their class (spam or ok) and it needs to be able to distinguish these classes and

Chapter 2. Background Research

---

classify new emails. Supervised learning can also be used for regression problems in order to predict a continuous **target** value, an example of this could be predicting the price of a house in a given geographical location. A house value can be determined by a set of **features** or (otherwise known as) **predictors** like location, condition or age. To train either a classification or regression problem, you need to provide as much example data as possible, including their predictors and labels.



*Figure 2.2 – Supervised Learning Email Filter Classifier*

In **unsupervised learning**, the input training data does not include instances with labels [9], instead the model tries to learn by itself. Some examples of learning algorithms could be clustering, visualisation and dimensionality reduction or association rule learning. Association rule learning [18] searches for frequent sets of items in data, from this new knowledge it produces correlations between sets which can be used for recommendations, an example is the

apriori algorithm [18] which can be used for transactional data to recommend buyers items such as clothing. A commonly used method is **clustering** [19], this organises a group of similar data items into a cluster that store different values compared to other clusters, a clustering algorithm will be able to find connections and distinguish groups without a human predefining them [9].

## Chapter 2. Background Research

---

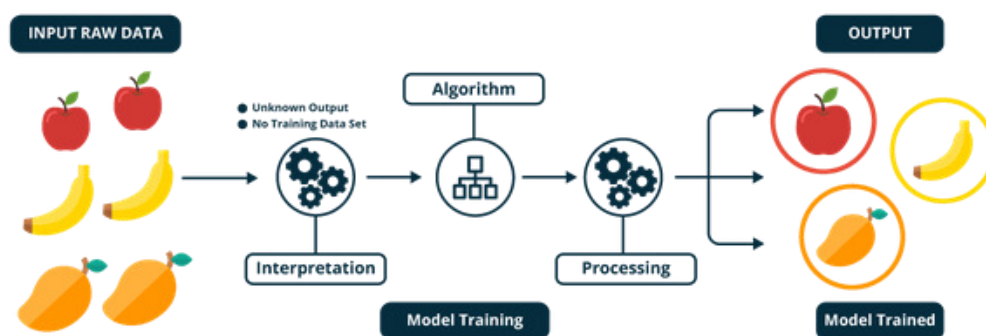


Figure 2.3 - Fruit Clustering Algorithm

Another largely used algorithm is **semi-supervised learning** [21] which learns in the presence of both labelled and unlabelled data. An area which is a good advantage for this method is photograph services, for example if you upload family photos it will recognise people's faces and provide an 'id' so that you could search an individual by this id [9]. Semi-supervised learning can also have great applications for improving model performance in supervised models where labelled data is insufficient, in order to provide an accurate output or prediction.

### 2.6. Machine Learning Approach to Predicting Driver's Risk

The purpose of this machine learning model is to predict the likelihood that a vehicle owner looking for an insurance policy, is a risk to the insurer. This algorithm is modelled on data representing a customer that has: a) made a claim or b) not made a claim. The problem can

therefore be identified as **binary classification** where a claim is a '1' and no claim made is a '0' [16]. There are various classification algorithms that could be used where some perform better or worse, considering the state of the data [22]. As mentioned by Porto Seguro [3] Insurance data tends to be incomplete and contain **noise**, (meaning irrelevant and randomly entered data) in the forms when creating a new driver policy. Different algorithms need different data representation, some examples of classification algorithms are: decision trees, naïve Bayes and neural networks. [22]

## Chapter 2. Background Research

---

### 2.6.1 Decision Trees

Decision trees are capable of performing both regression and classification tasks. They provide high-performing algorithms which can fit complex datasets, they also build the foundation of random forests which are one of the most powerful machine learning algorithms used for solving problems [9]. Decision trees are based off the concept of having a choice of decisions, and then following the path of the subsequent actions as a result. Decision trees are comprised from **nodes** and **branches**, there is also the **root node** at the top of the tree and **leaf nodes** at the bottom representing a class label or final decision [22]. Nodes in a tree represent a feature in an instance that needs to be classified, and each branch represents a value that the node can assume as an output, this can be demonstrated from the following figure [22].

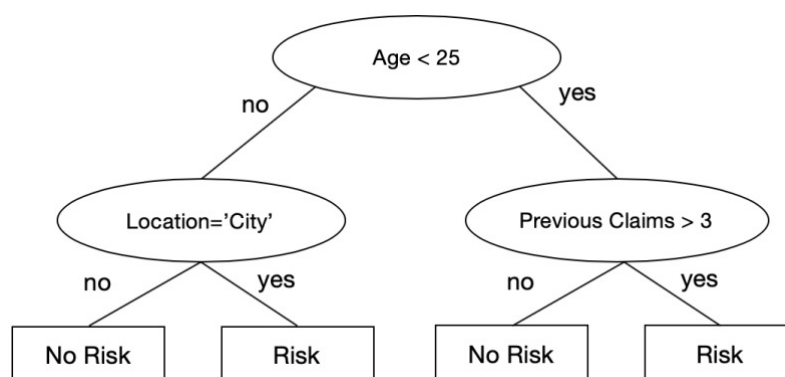


Figure 2.4 - Decision tree for a driver's risk

This decision tree represents a vehicle owner and their personal data given to an insurance provider. If a vehicle owner is seen as a risk, it means they can also be seen as a likely candidate for making a claim in the next year. Here the root node checks the feature 'Age', if this feature is not less than twenty-five it will then go through the path of the node location equal to 'City' (assuming the 'Age' feature is continuous). If the node location is equal to 'City' it will classify the vehicle owner as a risk, and otherwise not a risk.

---

## Chapter 2. Background Research

### 2.6.2 Naïve Bayes

It can be revealed that the naïve Bayesian network can have a good predictive performance compared to other classifiers such as neural networks or decision trees [23]. From the training data, naïve Bayes classifiers learn the conditional probability of each attribute  $A_i$  on the class label  $C$  [23]. For classification Bayes rule is applied to calculate the probability of  $C$  on the attribute instance  $A_1, \dots, A_n$ , the class which will be predicted is the class with the highest probability, for this to become feasible the classifier operates under the assumption that all attributes  $A_i$  are conditionally independent given the value of the class  $C$  [23].

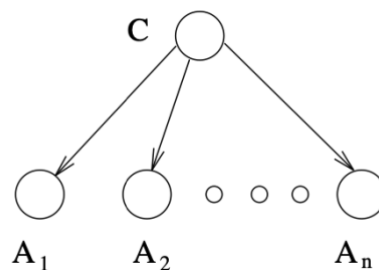


Figure 2.5 - Naïve Bayes network structure

Considering the fact that all attributes are processed independently, the performance can be surprising. This is due to the fact that it would ignore potentially important correlations between attributes [23]. There is a way however to include correlations and that is by using more general Bayesian networks, that way a user can set conditionally independent attributes [23].

### 2.6.3 Logistic Regression

Logistic regression is a **multivariable** learning algorithm for dichotomous results [24]. It is a classification method which has the best performance for models with two outputs, in particular ‘yes/no’ decision making [24], it is therefore suitable for predicting a vehicle insurance claim which would have two variables (claim or no claim). Logistic regression is

## Chapter 2. Background Research

---

similar to linear regression by its functionality, however linear regression provides a continuous output compared to the categorical output we desire [24]. Logistic regression works by having a single output variable  $y_i$ , where  $i = \{1, \dots, n\}$  and each  $y_i$  can hold one of two values ‘0’ or ‘1’ (but not both) [24]. This follows the Bernoulli probability density function [24]:

$$p(y_i) = (\pi_i)^{y_i} (1 - \pi_i)^{1-y_i}.$$

*Figure 2.6 - Bernoulli probability density function*

This takes the value ‘1’ where the probability is equal to  $\pi_i$ , and ‘0’ is equal to the probability as  $1 - \pi_i$ , the interest in this when  $y_i=1$  with an interest probability  $\pi_i$ . The classifier will then produce the output of the predicted label,  $\hat{\pi}_i$  is equal to ‘1’ if it is greater than or equal to its threshold (by default 0.5) [24].

$$\begin{aligned} \text{if } (p(y = 1)) \geq 0.5 & \quad \text{the instance} \in \text{class } (y = 1) \\ \text{if } (p(y = 1)) < 0.5 & \quad \text{the instance} \in \text{class } (y = 0) \end{aligned}$$

*Figure 2.7 - Logistic regression predicting labels*

## 2.6.4 Ensemble Methods

Typically in machine learning you tend to train multiple learning algorithms or models, and then choose the model with the best performance in terms of generalisation ability, ensemble methods however train multiple learning algorithms to solve a problem [25]. Ensemble learning operates by combining a set of learners referred to as **base learners**, some examples could be: decision trees, neural networks or other learning algorithms [25]. It is possible to either use learners of the same type, or you can heterogenous ensembles which include more variety of different types of learners [25]. As a whole ensemble methods are desirable because they can boost weak learners [25].

### Chapter 2. Background Research

---

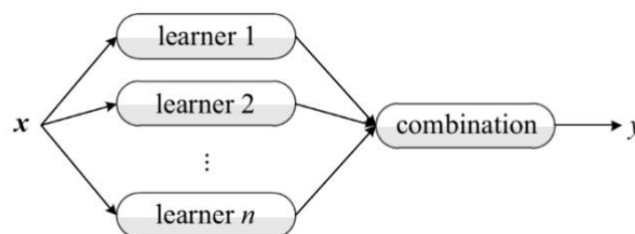


Figure 2.8 - Ensemble Methodology

It can be beneficial to improve the performance of the insurance risk classifier by combining multiple models, in particular because of the data imbalance problem of the ‘no claim’ class being a large majority. There are studies to show that **boosting** can boost weak learners to strong learners, observations have shown boosting is not as vulnerable to overfitting and has the possibility in some cases to reduce generalisation error [26].

## 2.7. Evaluating the Model



In order to analyse and evaluate the model properly, it is important to understand the concepts of generalization and splitting the dataset into a training and testing set to accomplish this [9].

### 2.7.1 Generalisation, Overfitting and Underfitting

Referring back to generalization (as described in 2.5. Machine Learning), it is an important concept to understand in order to determine whether or not a trained classifier model can perform well on unseen data [27]. There are certain concepts which show how data can be fitted to a model.

**Overfitting** can be described as when a learning algorithm fits the training data too well, as a result the noise and outliers which are not as useful information compared to the average data are memorised [28]. The consequence of overfitting data is that the performance can drop significantly on unseen data sets or test sets [28]. In general there are methods to reduce overfitting, some examples include: gathering more training data or reducing the

#### Chapter 2. Background Research

---

noise in the training data by removing the outliers [9]. When the model overfits there tends to be a high bias and low variance [28].

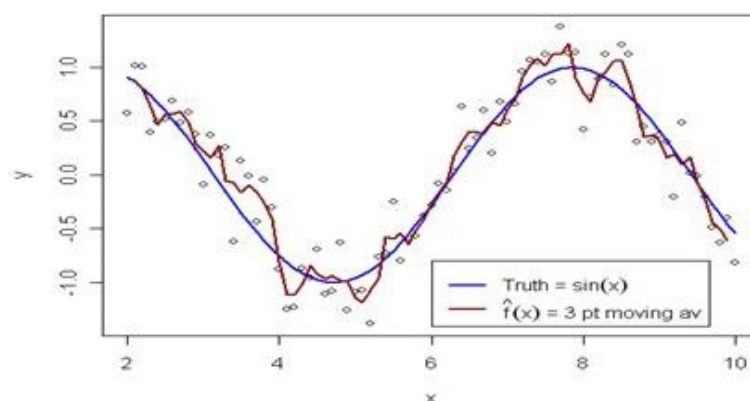


Figure 2.9 – Overfitting a model

**Underfitting** can be described as the opposite of overfitting, this occurs when the learning algorithm is not capable of analysing the variation of the training data [28]. The classifier will

not be capable of producing representative predictions of the data, the main reason behind this is that the model is too simple for the complexities of the training data [28]. There are methods for helping to prevent underfitting and overfitting collectively, some examples can be cross validation or early stopping techniques [28]. You can also select a higher-performance model with improved parameters or provide better performing features [9]. When the model underfits there tends to be a low bias and high variance [28].

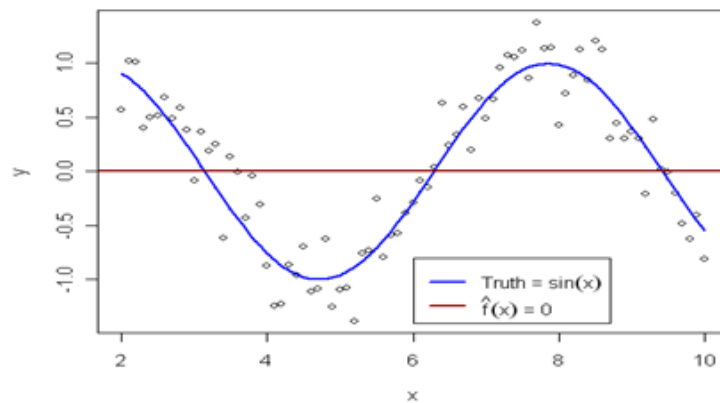


Figure 2.10 - Underfitting a model

## Chapter 2. Background Research

---

The main objective to achieve generalisation is by having the right balance between overfitting and underfitting, this means a balance between fitting the training data perfectly and training the model to be simple enough to generalize well [9]. As shown by the following figure, the solid blue line shows how the predictions are between the bulk of data points which should produce accurate predictions.

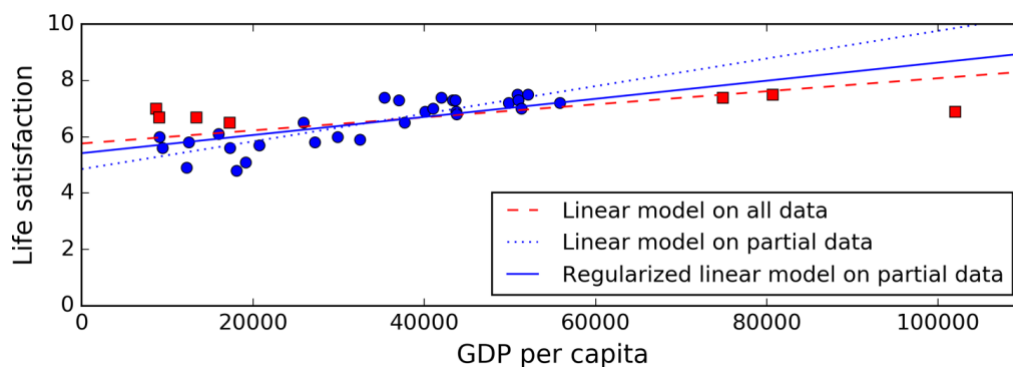


Figure 2.11- Generalization of data

In order to evaluate the generalization capability we use evaluation metrics and these will indicate how well the model fits the data [27]. To evaluate the model effectively it requires the dataset to be split into a training set and a testing set [9]. There is an error rate for new instances being classified incorrectly and this is referred to as generalization error, by evaluating the model on the test set (unseen cases), it can provide an estimate of the error [9].

## 2.7.2 Evaluation Metrics

There are several metrics to measure a classifier model and evaluate how well the model fits a dataset, as well as how it performs on unseen data [27]. A common misconception found when evaluating a model, is that the accuracy metric is favourable and for the most of the time a good measurement [27], this is usually because it is easier to understand and implement [29]. Accuracy alone for a classification problem cannot always be reliable because it can provide bias for a majority class giving high accuracy and weak accuracy for the minority class, making it less informative for predictions, especially in the case of

### Chapter 2. Background Research

---

imbalanced data [29]. Car insurance claims are a good example because it is known that the majority of policy holders do not make a claim, therefore if accuracy is used there would be a bias towards a 'no claim' class [29]. In order to understand how accuracy works it is important to understand firstly how a confusion matrix works.

A **confusion matrix** is used for binary classification problems and is a very useful method to distinguish which class outputs were predicted correctly or not. As shown in figure 2.12, the rows represent the predicted class while the columns represent the actual class [29]. In the matrix 'tp' and 'tn' signify the quantity of correctly classified positive and negative instances, whereas 'fp' and 'fn' represent the quantity of incorrectly classified positive and negative instances [29].

	Actual Positive	Actual Negative
Predicted Positive	True positive (tp)	False negative (fn)
Predicted Negative	False positive (fp)	True negative (tn)

Figure 2.12 – Confusion matrix

In the case of car insurance claims, true positive would represent no claim made and true negative would represent a claim. It would be ideal for the top left and bottom right squares to have a higher quantity (correctly predicted claim), compared to the bottom left and top right (incorrectly predicted claim), this would indicate a model that generalizes correctly and avoids overfitting or underfitting the data.

As stated previously, **accuracy** is a simple metric to understand by humans, it measures the ratio of correctly classified predictions divided by the total number of instances [27]. The **Precision** metric is used to measure how trustworthy the class is classified and it correctly belongs to the class [27]. Another useful metric is **recall** which is used to measure how well the fraction of a positive class become correctly classified [27], this essentially shows how well the model can detect the class type.

## Chapter 2. Background Research

---

Accuracy	Precision	Recall
$\frac{tp + tn}{tp + fp + tn + fn}$	$\frac{tp}{tp + fp}$	$\frac{tp}{tp + tn}$

Figure 2.13 – Evaluation metrics: Accuracy, Precision & Recall

There is a weakness to accuracy, precision and recall where they are not as robust to the change of class distribution, a popularly used ranking evaluation technique is to use the **area under the curve (auc)** metric or otherwise known as the **receiver operating characteristic (roc)** [30]. If the

test set were to have a change in its distribution of positive and negative instances, the previous metrics may not perform as well as when they were previously tested [30]. The roc curve however is insensitive to the change of the proportion of positive and negative instances and class distribution [30].

ROC graphs are two-dimensional where the y-axis represents the true positive rate, and the x-axis represents the false positive rate [31]. With reference to figure 2.14, the points on the roc graph are discrete classifiers which only output a class label, each classifier generates a false positive and true positive rate pair which correspond to a point in the roc graph [31]. There are a couple important points to consider, the coordinate (0,0) represents the solution never producing a positive classification, the reverse is where the coordinate (1,1) produces positive classifications unconditionally [31]. The coordinate (0,1) represents a perfect classification as the solution predicts all positives correctly with no false positives, it can be more ideal to have the solution towards the north-west area on the graph rather than north-east, this is because the north-west will have a higher true positive rate with less false positive [31].

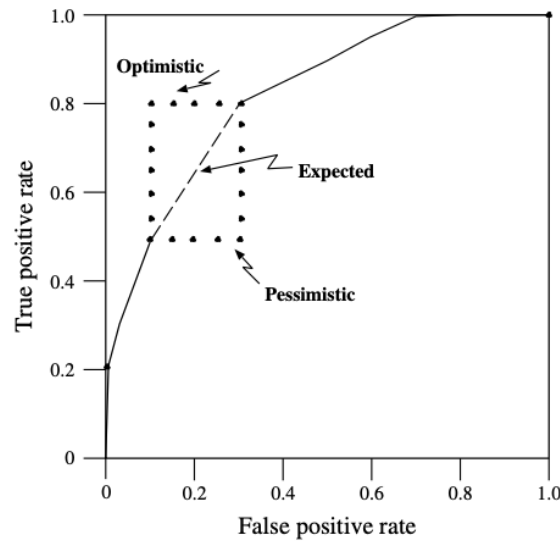


Figure 2.14 – Receiver operating characteristics graph (ROC)

With consideration to the insurance problem and as show by figure 2.14, it is ideal for the classifiers to be above the curve for better performance of predicting a claim being made, if the classifiers are in the pessimistic range below the curve they are not performing well [31].

It is useful to utilise different methods of evaluation, however It is stated by Porto Seguro that they are using the **Normalized Gini coefficient** therefore it is possible to use the other metrics for testing but the final testing will be done using normalized Gini [3]. This scoring metric is almost identical to the area under the curve metric, the distinguishing feature is that the range is between (0 and 0.5) whereas auc is between (0 and 1) [3]. They can be converted between each other by the following algorithm:  $[G + 1 = 2 * AUC]$ .

## 2.8. Python Machine Learning Libraries

There are various machine learning libraries which will assist this project in producing a predictive machine learning algorithm. The following libraries will be used in order to save development time and utilise pre-written functions which are designed principally for machine learning purposes:

- **Scikit-learn** [32] is a machine learning library that supports supervised and unsupervised learning. There are numerous tools fundamentally surrounding fitting models to the target variable, data pre-processing, model selection and evaluation.
- **Matplotlib** [33] is an extension of the numpy library for handling large arrays of datasets. It is a plotting library for Python that provides functions to plot and visualise datasets in many forms such as: scatter plots, histograms, line plots as well as various other formats. This library facilitates static, animated and interactive images in Python, with customizable features to present the data clearly.
- **Pandas** [34] is a Python software library for data manipulation and analysis. It provides tools for reading and writing data for in-memory data structures, as well as formats such as CSV or text files. Columns in data structures can be inserted or deleted for size mutability. Pandas also supports high performance merging and joining of datasets, including various other features.
- **XGBoost** [35] is a gradient boosting library designed to be highly efficient and flexible, it implements machine learning algorithms through the gradient boosting framework. For speed and accuracy, XGBoost takes advantage of its parallel tree boosting.

- **Numpy** [36] is a package for handling powerful n-dimensional arrays, it also provides: numerous mathematical functions and a high performance multidimensional array object with tools to operate the arrays.

## Chapter 2. Background Research

---

- **Imbalanced-learn** [43] is a Python package which has various different sampling techniques and methods, these include under sampling and over sampling as well as combinations and ensembles of under and over sampling. This package involves strategic or random sampling methods, these are useful to choose between time and computational power.

### 2.9. Existing Solutions

There is a lot of motivation for automotive insurance companies to implement machine learning algorithms in their business, three major categories of where they are used can be: chatbots, driver performance monitoring as well as insurance market analytics [37].

An example of insurance market analytics is a model that predicts a claim severity, essentially the amount of funds needed to repair vehicle damage [38]. The difference between this project and the project to be created, is that it will focus on distinguishing a policy holder likelihood on filing a claim, compared to predicting the cost of a claim and the funds required for the claim damage. This type of example represents how insurance providers are looking into many different forms of applying machine learning on their customer data.



This project is focusing on insurance market analytics, as described by Porto Seguro they are looking for alternative and more complex machine learning models to predict a policy holder likelihood to produce a claim in the next year [3]. An example of a similar and good solution to the same problem is the thesis “Research on Probability-based Learning Application on Car Insurance Data” [39], where they use a Bayesian network to classify either a claim or no claim. The motivation behind Porto Seguro is to produce a machine learning solution which is more complex or varied in the approach, therefore this project will have more variety in combining models and ensemble methods for an accurate claim prediction.

---

## CHAPTER 3

---

---

### Design Specification & Approach

---

This section details the plan and structure for the design and development of the predictive model. I will be discussing Porto Seguro’s business model and how my driver risk prediction component is associated with the overall dynamic behaviour of this business model. I will also outline the requirements specification to satisfy the clients’ needs, as well as the system design and architecture to define how I intend on implementing the solution. Lastly I will describe how the development methodology chosen will help to achieve this in the given timeframe.

#### 3.1. Business Model and Dynamic Behaviour

It is important to firstly understand how Porto Seguro’s business model would operate on an abstract level, by breaking down what the components and data stores are in order to see where the solution would be implemented, this is also useful to gain insight on how more upstream components can be added to the business model in future (for example predicting if the policy holder may commit fraud). Figure 3.1, outlines the dynamic behaviour of how their system will operate, as well as what type of data will flow through my component (driver risk predicting) and how it will do so.

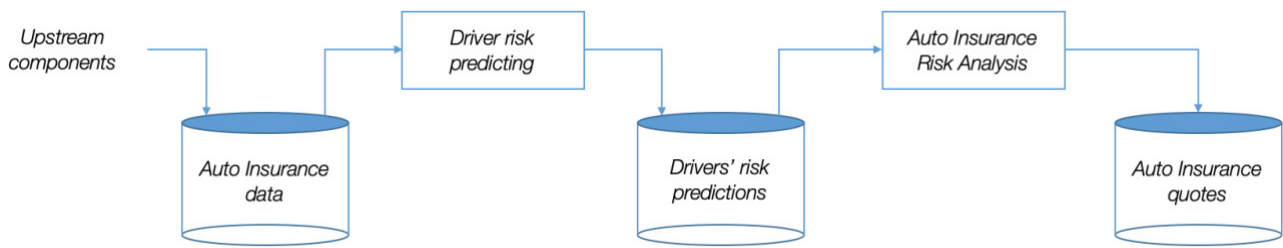


Figure 3.1 - Machine learning pipeline for generating quotes

Figure 3.1 can be interpreted by understanding the following: upstream components can refer to anything that is used in the pipeline, relating to generating automotive insurance quotes., and as previously mentioned that can include anything such as a predictive model for fraud detection. The driver risk predicting is where the model trains the data and then

## Chapter 3. Design Specification & Approach

---

produces driver risk predictions, the predictions are analysed and the risk of the driver making a claim is determined as a factor towards the insurance quote.

### 3.2. Requirements Specification

Referring back to chapter 1.3, the overall aims and objectives of this project were set with regards to the clients general requirements and to summarise the points, Porto Seguro require a machine learning solution which provides accurate predictions for their policy holders on whether or not they are likely to make a claim. This can be broken down more specifically into software functional requirements, and these requirements will be evaluated later in the results and evaluation chapter to test the validity of the model, the solution can be tested specifically on its completion of the acceptance criteria. The solution can be categorised into three main sections: Data preparation, training the model and testing the model.

**Data Preparation requires the following:**

- **The system can load in the training and test datasets:** The training dataset is required for training the classifiers, and the test set is required for testing the classifiers generalization ability.
  - **Acceptance criteria:** Using the pandas library, the system recognises the data and can output simple functions such as *df.info()* or *df.head()*.
- **Both the training and test datasets are pre-processed:** Certain models require data formatted in a specific way before being able to function and train the data. This involves handling matters such as missing data, according to Porto Seguro [3] a lot of their customers tend to not fill in all their personal information in the forms, therefore this has to be accounted for. It also requires handling of outliers, converting data types and categorical values (one-hot encoding).
  - **Acceptance criteria:** All models being: logistic regression, random forest and XGBoost are able to operate and run the data for their algorithms.

### Chapter 3. Design Specification & Approach

---

#### **Training the model requires the following:**

- **The best solution is chosen from multiple classifiers:** The system should train the models on the training data for the classifiers: logistic regression, random forest and XGBoost models.
  - **Acceptance criteria:** Each model's predictions are compared as a final result.
- **Each classifier can produce a 'claim' or 'no claim' prediction:** The models need to train the data and distinguish the classes.
  - **Acceptance criteria:** Claims classified as: claim equal to '1' and a no claim equal to '0'.
- **The system needs to be able to handle a data imbalance:** Predictions must be appropriate and accurate, they cannot produce biased predictions towards a majority class, i.e. predicting a 'no claim' because most customers do not produce a claim.
  - **Acceptance criteria:** The proportion of the confusion matrix has higher true positive and true negative values, than false positive and false negative values.

### **Testing the model on unseen evaluation data requires:**

- **The system needs to be able to produce accurate predictions similarly on training data, to unseen data:** If the predictions are only good for the training data and not real use cases, it does not generalize and is not useful.
  - **Acceptance criteria:** For each 'k' fold, the Gini score and the confusion matrix true positive and true negative values are within a small margin. The Gini score of the test dataset held on Kaggle should also hold a similar score to the folds.
- **The system needs to be able to appropriately evaluate a good prediction and overall solution:** To determine whether or not a classifier produces a reliable and accurate claim, it will be evaluated based on certain factors.

### Chapter 3. Design Specification & Approach

---

- **Acceptance criteria** is relative to the following:
- **Confusion matrix:** The proportion of the confusion matrix has higher true positive and true negative values, than false positive and false negative values.
- **Normalized Gini coefficient:** As described by Porto Seguro, a perfect model scores 0.5 meaning every prediction is correct, and a score of 0 implies the model randomly guesses predictions values, the minimum Gini score should be at least 0.24 [3].
- **Area Under the Curve score:** similar to the Gini score, this ranges between 0 for random guessing and 1 for a perfect score, the minimum score should be 0.62. Using the ROC curve diagrams, it provides a clear visualisation to test and determine the model's predictive capabilities.

The solution is built primarily on functional features as it is more of an algorithm rather than a software product (such as a website). There are however a couple of non-functional requirements to describe how the solution should behave:

### Non-functional requirements:

- **The system should be robust:** The predictions cannot significantly begin to vary when tested against new cases of unseen data, neither can the performance and time of the model itself, this is mainly reliant on the data imbalance issue.
- **The system should be reliable:** The predictions should be consistent and stay within a certain threshold when the model is run multiple times.
- **The system should have re-usable components:** For stakeholders who may be interested in the solution, the implementation must be structured and organised so they can easily use sections which are needed, with this in mind the datasets are not provided in the solution for stakeholders due to the copyright contract signed with Porto Seguro.
- **The system performance should not take a too significant amount of time:** Predictions should be produced within in a reasonable time when the program is run.

## Chapter 3. Design Specification & Approach

---

### 3.3. System Architecture

The solution will be structured by firstly having a file *main.py* in the central directory *ML\_project* which acts as the main loop for calling functions from other packages by importing them. The package *classifiers* inside the *ML\_project* directory will store all the classifiers which have been trained and tested on the data inside the *datasets* package, which stores the training and testing datasets. For all data pre-processing tasks there is a package named *pre-processing*, this will store all the relevant files to handle areas such as missing data and feature scaling.

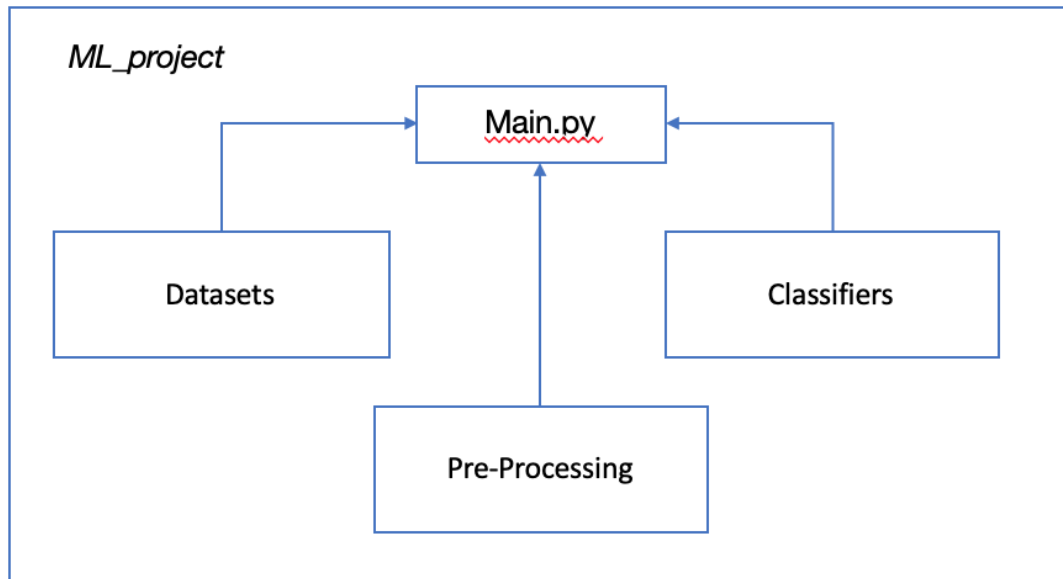


Figure 3.2 – System architecture and directory diagram

This figure represents the system architecture content structure. The solution has been structured in this format so that components are easily re-usable, as well as to avoid cluttering which would make the code difficult to read. It means that files such as *feature\_scaling* can apply their functions to other datasets in future without having to re-write the code, as a result saving development time.

## Chapter 3. Design Specification & Approach

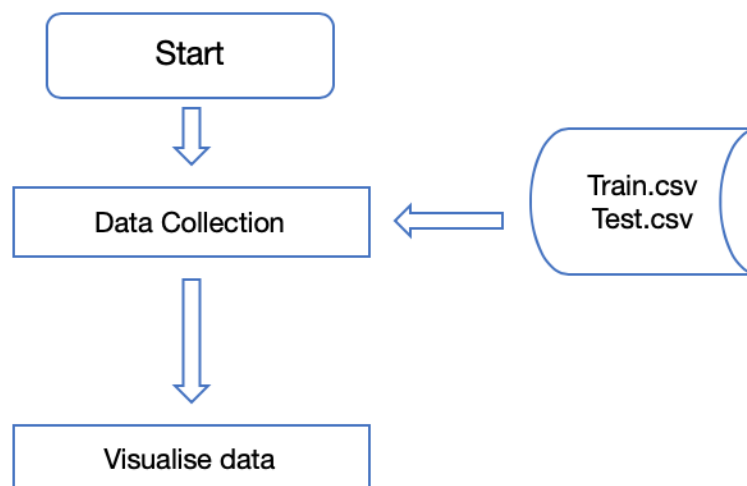
---

### 3.4. System Design Approach and Flow

In order to understand the structure and approach for implementing the machine learning solution, the system design can be described by a flowchart on an abstract level which is simple to understand and read. This will help to visualise what is required at each stage of the implementation, and be able to compare it to the overall aims and objectives and the previously discussed functional and non-functional requirements and how they are satisfied at each stage. The system approach is split into three stages.

### Stage one

The first stage involves reading in both the training and testing datasets, and then visualising each feature and its values. This will provide insight on which features have outliers and how feature scaling can be used to handle outliers. Visualising the data will also highlight features which have imbalanced data. By recognising features with data imbalance, we can later verify whether or not the feature is useful to the machine learning algorithm for performance, data imbalance can be recognised by noticing for example, some categorical feature values having a far larger frequency compared to the others.



*Figure 3.3 – System design flow-chart stage one*

## Chapter 3. Design Specification & Approach

---

### Stage two

The second stage involves reading in both the training and testing datasets, and then carrying out pre-processing by: imputing in feature's missing values with a new value or deleting the feature, one-hot encoding for changing categorical feature data into a readable format and feature scaling

with normalisation so that different features have the same value ranges, as machine learning algorithms perform better with normalized data.

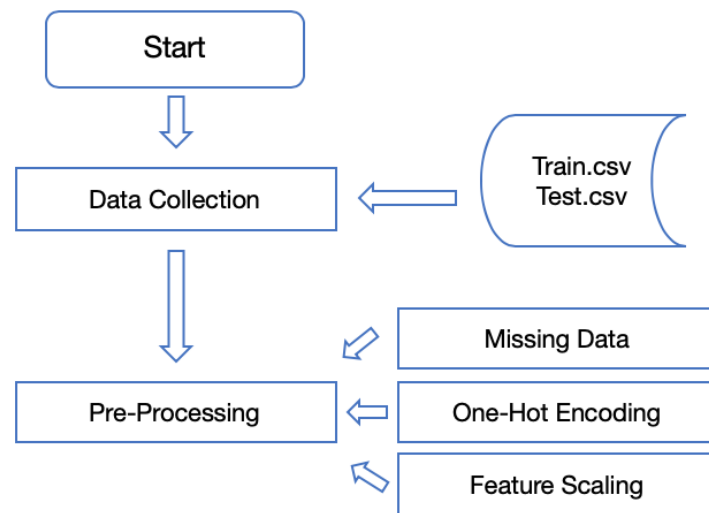


Figure 3.4 – System design flow-chart stage two

The feature scaling method which will be used is min-max scaling. This normalises the feature's values to a fixed range between zero and one, which leads to smaller standard deviations and helps to suppress outliers. One-hot encoding works by taking a integer encoded feature and producing a vector with an equal length to the number of categories for that feature, now in binary format. For features which have missing values (indicated by a -1 value), these need to be accounted for in order for a learning algorithm to be able to read the data. If the feature is a categorical it should use the mode function and if the feature contains continuous values it should use the mean function to impute values based on the averages of the feature.

## Chapter 3. Design Specification & Approach

---

### Stage three

The third stage involves the bulk of implementing the solution. It works by firstly splitting the training dataset into another training and test dataset combination, then further pre-processing is



carried out on both the original training and testing datasets for feature selection and dimensionality reduction. In order to determine which features are the most useful for prioritising, as well as the features that are not informative to the machine learning algorithms and should therefore be dropped, each individual feature will be tested against the training dataset using the normalized Gini coefficient score, additionally the Pearson correlation coefficient will be used to identify which features correlate well with each other and with regards to the target variable. Features with a better correlation to the target variable (which determines a claim being made or not) are preferable and should therefore be prioritised for choosing which features to keep.

It is important to note that cross validation will need to be applied during the next steps in this stage. This involves splitting the training data into a training and testing split by a specified 'k' amount of times, this is used to track the generalisation ability of each model as they are being trained. Each individual learning algorithm or model needs to be trained on the training data, this includes training for the following classifiers: naïve Bayes, random forest, XGBoost and logistic regression. The performance of classifiers will be tested with a confusion matrix as well as their Gini score, parameter tuning will be used in order to improve the performance of the classifiers predictive ability.

After the models performance is acceptable there will need to be a final evaluation on which model performs the best, this will compare the results from the confusion matrix and Gini scores as well as a roc graph visualisation. Once this has been completed there will be a final solution which can produce better and more reliable predictive results compared to the other models.

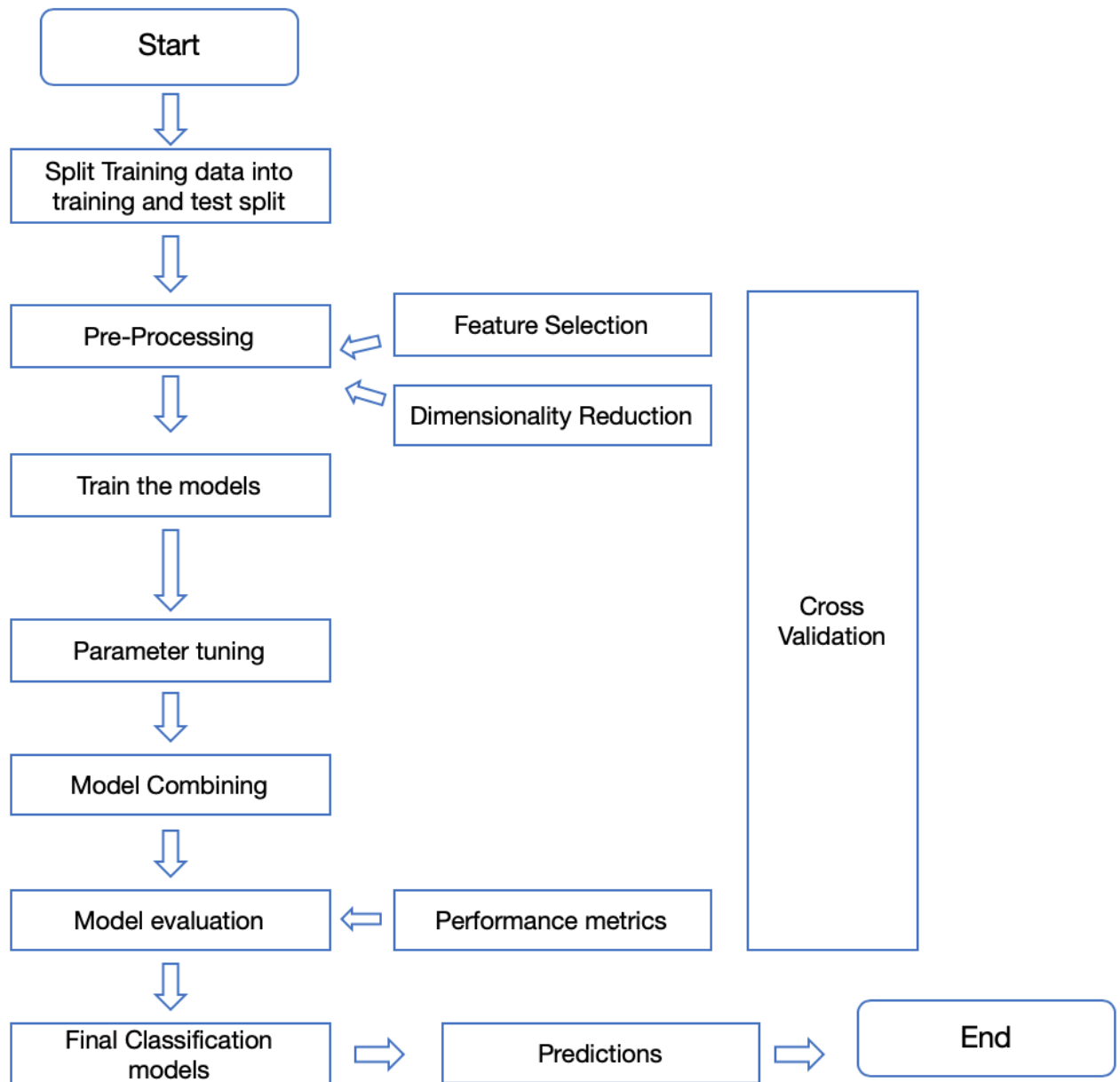
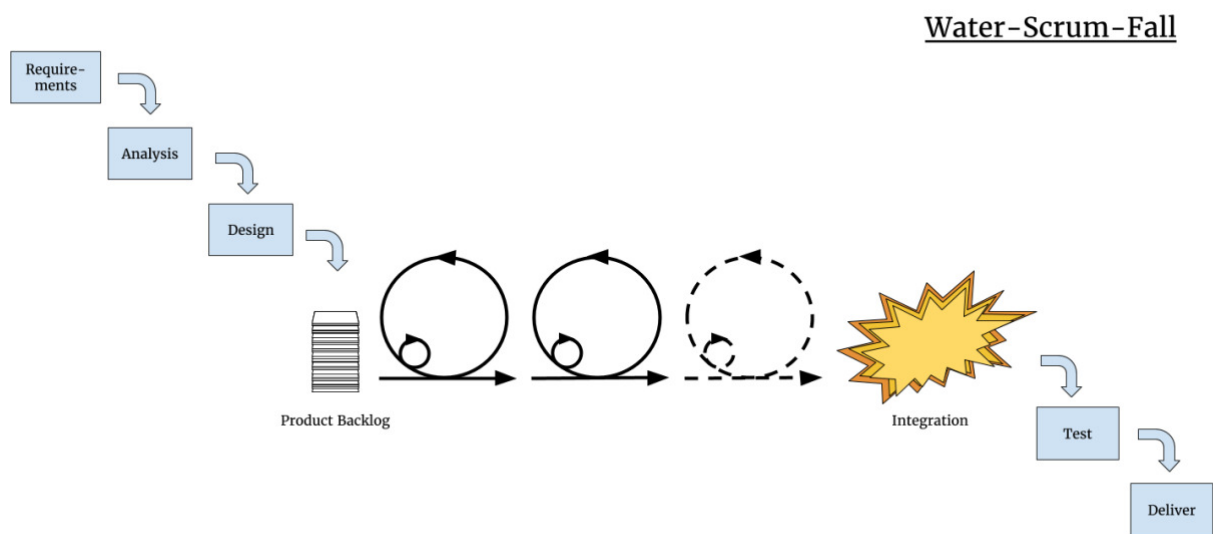


Figure 3.5 – System design flow-chart stage three

### 3.5. Development Strategy and Methodology

It is important to maintain a consistent structure for the development of the solution. This is to ensure that the solution created is to a high standard and with the intention that within the design, development and testing stages errors can be prevented, or become more easily reversible by adopting the right development methodology. It is also important to be able to show deliverables more regularly to make sure that Porto Seguro's general requirements will be met. Due to these reasons the agile methodology [45] is an appropriate method to use.



*Figure 3.6 - The agile methodology for software development (Water-Scrum-Fall)*

The agile methodology operates differently from the traditional approach of a waterfall; this is because the focus is more on being adaptive rather than predictive. Models may not perform as well as intended, therefore it is important to be able to adapt and design alternative models that provide better predictive results. The agile approach does however adopt the concept of the waterfall, however uses the stages in incremental delivery and adaptable planning by iterations. The development and testing of this project can therefore be broken down into the following iterations:

- Iteration One: Before the machine learning model can be made, data preparation and pre-processing must take place on the data so that it can be in the correct format for a machine to interpret. As a result of this iteration, there will be a dataset which can begin learning to classify because the model will be able to operate on the new data format.
- Iteration Two: This focuses on developing three classifiers: logistic regression, random forest and XGBoost. These classifiers will need to be trained on the training dataset and then tested on unseen data to determine its generalization ability. The result of this iteration is a group of classifiers which can produce predictions of a claim being made or a claim no being made.
- Iteration Three: The models or classifiers will need to have their parameters tuned and once their best possible prediction accuracy is determined, the classifiers can be combined with ensemble methods to further enhance their performance.

After full completion of these iterations or stages, the solution should produce a more reliable or accurate set of predictions compared to a standalone classifier.

---

## Implementation

---

This section will describe and outline the most important parts of the developed solution, this section will go into the finer details of the developed code that is required to satisfy the client functional and non-functional requirements (as mentioned in chapter 3.2). It will also provide justifications for certain design decisions made. There will also be descriptions of how the classifiers generate predictions, as well as how classifiers are evaluated to test their performance.

### 4.1. Understanding the Porto Seguro Dataset

Before making any modifications to the dataset for creating the machine learning model, it is important to understand how the datasets have been structured. Porto Seguro have provided two datasets where the first set is a training set and the second is a test set, the distinguishing feature here is that the training set has an additional 'target' column which indicates whether a claim was made (value equal to one) or a claim has not been made (value equal to zero). In the datasets each row represents an independent customer and their personal information. There is also a data description provided [40] which provides important information on the data preparation which has already been processed, the main points to observe are the following:

- Values of '-1' indicate the feature was missing from the observation.
- Feature names include the postfix 'bin' for binary features and 'cat' for categorical features.
  - Binary data has two possible values '0' or '1'.
  - Categorical data (one of many possible values) have been processed into a value range for its lowest and highest value respectively.
- Features without a postfix are either continuous or ordinal.
  - The value range appears as a range which has used feature scaling, therefore feature scaling is not required.
- Features belonging to similar groupings are tagged as 'ind', 'reg', 'car' and 'calc'.
  - 'ind' refers to customers personal information such as their name.

- 'reg' refers to a customer's region or location information.
- 'calc' are Porto Seguro's calculated features.

### 4.2. The Algorithm Structure

It is important to ensure that the classifiers, datasets and pre-processing functions are stored separately. This provides encapsulation by giving one file one meaning as we only need the output, rather than the specific details. This makes it easier when producing a package so that the functions can become re-usable, as well as this the code is far simpler to understand and read. The functions and files which build the solution follow the same structure as described in chapter 3.3, to break this down further each package or directory includes the following:

- **ML\_project:** This is the main directory which stores all the sub-packages of classifiers, datasets and pre-processing. It also stores the file *main.py* which imports and calls the functions in these sub-packages. Lastly it stores the file *calculate\_gini.py* for calculating the normalized Gini coefficient.
- **Classifiers:** This package stores the learning algorithms functions for producing claim predictions for logistic regression, random forest and XGBoost. The files stored are *logistic\_regression.py*, *random\_forest.py* and *xgboost.py*.
- **Datasets:** This package simply stores the training dataset *train.csv*, as well as the testing dataset *test.csv*.
- **Pre\_processing:** This package stores the files *feature\_scaling.py*, *feature\_selection.py*, *missing\_values.py* and *one\_hot\_encoding.py*.

The file *feature\_scaling.py* was not used in the end classifier model because it did not impact the predictive performance, from this we can only assume that Porto Seguro have done some pre-processing on the continuous values for the final classifier for an import not to be necessary.

## 4.3 Data Pre-Processing and Preparation

### 4.3.1. Loading and Visualising the Data

The first step is to load in both the training and testing datasets with the `load_data()` function, as shown by figure 4.1 these are inputted from csv files where the columns are the features, this includes the target variable for the training dataset (labels). This can be done through the pandas function `pd.read_csv()` which converts the csv file into a pandas data frame so the data can be manipulated with in Python. Values in the dataset that are missing are identified as '1' and by setting the parameter `na_values="-1"`, it changes missing values to a data type of NaN, this is needed later for the pandas function `fillna()` to simply handle and replace missing values with a new suitable value.

```
def load_data():  
    train_df = pd.read_csv('../datasets/train.csv', na_values="-1")  
    test_df = pd.read_csv('../datasets/test.csv', na_values="-1")  
    return train_df, test_df
```

*Figure 4.1 – Loading training and testing datasets*

After reading in the data it is important to gain an understanding of how the different features have an impact on each other through correlation, as well as visualising the distribution of data such as the feature's variance to determine if feature scaling may be required. Figure 4.2 represents well the distribution of data for the other features of its category and how they have a bell shaped curve to show the data is normalised, due to this it is not necessary to apply feature scaling. This is important as normalized data tends to show better performance for the machine learning algorithms [9], it may also have an impact on certain models weights if features are on different scales as some may update faster than others [41].

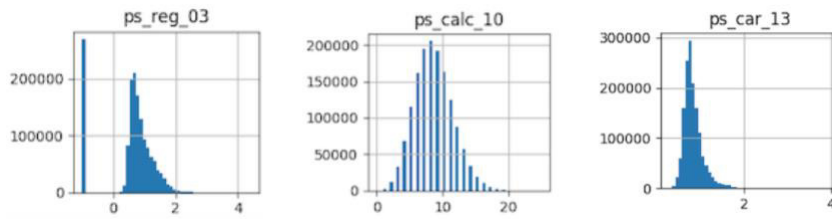


Figure 4.2 – Visualising regional, calculated and car features

## Chapter 4. Implementation

---

### 4.3.2 Feature Extraction

Before it is possible for a machine learning model to train data, the data must first be in a format which is machine-readable. This primarily includes: handling missing values, feature scaling (as previously mentioned) with outliers and converting categorical features into a numerical value. After testing the results from simple classifiers, I found that feature scaling did not have much of an impact, therefore I have removed the feature scaling functions. Typically this stage also covers: handling duplicates, correcting errors and converting data types, however this has already been covered by Porto Seguro. For the purpose of separating the different sections of code, each file will save the datasets id and target features before removing them and then pursuing their task, for example this happens with the `missing_data.py` and `one_hot_encoding.py` files. From removing the id and target features the dimensionality of both the training and testing datasets are the same, this makes data manipulation more straightforward.

#### Handling and filling in missing values

As described previously, the missing values in the datasets have been converted to the NaN data type, this enabled me to be able to identify the proportion of missing data for each feature which has a NaN value in both the datasets. Using the function `isna().sum()*100/n` where n is equal to the column length of the dataset, the following figures show the sum of the percentage of missing or NaN values for each feature in the training and testing datasets respectively:



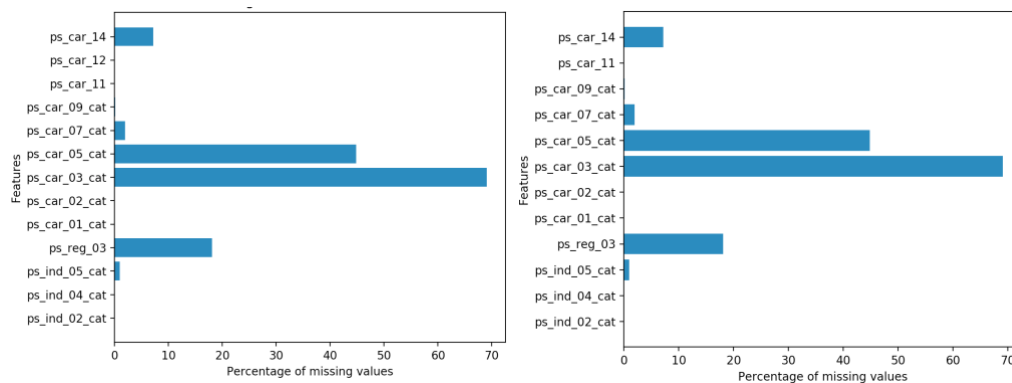


Figure 4.3 - Missing data from train.csv and test.csv

## Chapter 4. Implementation

As shown by this figure the features have a large proportion of missing values being roughly 70% for *ps\_car\_03\_cat* and 45% for *ps\_car\_05\_cat*, therefore the features are not that reliable as there are too few values to represent the features true meaning. Assigning new values which are missing to each customer record for these features may also not be representative of the feature's meaning, and negatively impact the learning algorithm's performance. Due to these reasons the features have been dropped and removed from the datasets. The following figure is the function *handle\_missing\_values()* which passes the parameters of the datasets:

```
def handle_missing_values(train_df, test_df):
    print("handling missing values...")
    train_id = train_df.id
    test_id = test_df.id
    target = train_df.target

    #Drop ps_car_03_cat and ps_car_05_cat columns, [id,target] re-added later
    train_df.drop(["id", "target", "ps_car_03_cat", "ps_car_05_cat"], axis=1, inplace=True)
    test_df.drop(["id", "ps_car_03_cat", "ps_car_05_cat"], axis=1, inplace=True)

    #Imputation of missing values, cat and bin use mode, ordinal/normal values use median
    for col in train_df.columns:
        if 'cat' in col:
            train_df[col].fillna(value=train_df[col].mode()[0], inplace=True)
            test_df[col].fillna(value=test_df[col].mode()[0], inplace=True)
        elif 'bin' in col:
            train_df[col].fillna(value=train_df[col].mode()[0], inplace=True)
            test_df[col].fillna(value=test_df[col].mode()[0], inplace=True)
        elif 'bin' or 'cat' not in col:
            train_df[col].fillna(value=train_df[col].median(), inplace=True)
            test_df[col].fillna(value=test_df[col].median(), inplace=True)

    train_df.insert(loc=0, column="target", value=target)
    train_df.insert(loc=0, column="id", value=train_id)
    test_df.insert(loc=0, column="id", value=test_id)

    return (train_df, test_df)
```

Figure 4.4 - Filling in missing feature values

After removing *ps\_car\_03\_cat* and *ps\_car\_05\_cat*, this function iterates through the pandas training and testing data frames and checks to see if the columns are either: categorical, binary or a continuous feature. The *fillna()* function allows the NaN value to be replaced by a specific value by imputation methods. Categorical and binary feature values are replaced by the mode, continuous feature values are replaced by the mean of their column values. This is because categorical data works well using the mode and continuous data works well using the mean, both methods are also simple and quick for imputing values [42].

## Chapter 4. Implementation

---

### **One-hot encoding categorical data**

The next step is to convert categorical features into a state which the learning algorithms can read. Porto Seguro's anonymization converts the original categorical data strings into a numerical value, an example being *vehicle type=vauxhall* is now equal to 5. This method is known as integer encoding however it does not take into account ordinal data where there may be an order if one value has more importance than another. As the data is anonymized there is no way of knowing whether there is an order and therefore it is appropriate to use one hot encoding as this uses binary representation instead to assist with this issue [44]. The following figure 4.5 is a snippet of code describing the main purpose from the function *one\_hot\_encode()*. It encodes the features by using the pandas function *get\_dummies()* which creates extra pandas columns (or series) up to the maximum of the columns maximum numerical value, for example if there are five vehicles in a feature there are now five new columns which are binary encoded.

```

if 'cat' in col:
    train_df[col] = train_df[col].astype('category')
    test_df[col] = test_df[col].astype('category')

    #OHE categorical features
    train_ohe = pd.get_dummies(pd.Series(train_df[col]),prefix=col)
    test_ohe = pd.get_dummies(pd.Series(test_df[col]),prefix=col)

    #Add new OHE features onto dataframes
    train_df = pd.concat([train_df,train_ohe],axis=1)
    test_df = pd.concat([test_df,test_ohe],axis=1)

    #Remove old categorical feature
    train_df.drop([col], axis=1, inplace=True)
    test_df.drop([col], axis=1, inplace=True)

```

*Figure 4.5 - One hot encoding categorical features*

### 4.3.3 Feature Selection and Dimension Reduction

As seen previously there has been some feature selection that has already been processed, this is the removal of the features *ps\_car\_03\_cat* and *ps\_car\_05\_cat* because they had too many missing values to be useful to a learning algorithm. It is important at this stage to further visualise and identify which features are more useful than others by using the Pearson correlation coefficient as shown in the following figure:

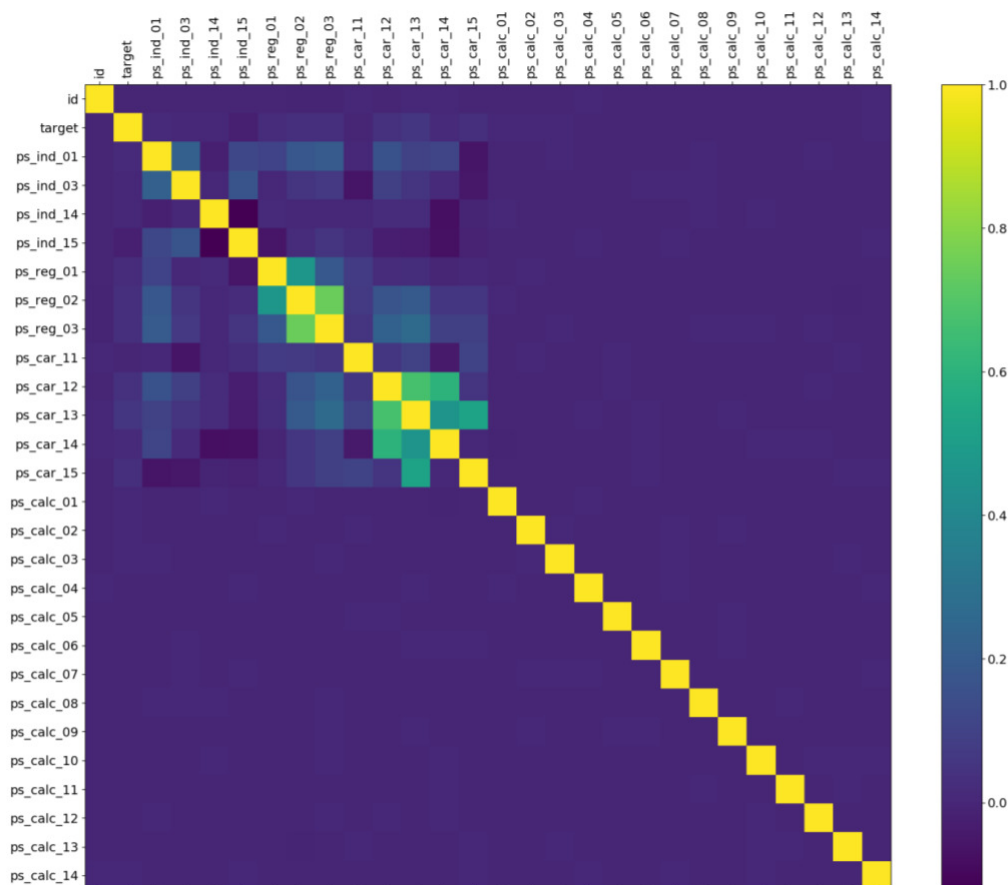


Figure 4.6 - Pearson correlation coefficient of features

From these results it is clear to see that the 'calc' features do not correlate well other features, and despite the other features having a fairly weak correlation with the target variable, the 'calc' features do not have any correlation with the target variable. This is important because the target variable (label) determines if there is a claim being made or

#### Chapter 4. Implementation

---

not, therefore there should be some correlation. Due to this the 'calc' features have been dropped from the data frames for both datasets.

Removal of the calc features is carried out in the file *feature\_selection.py* in the function *feature\_removal()*, this operates by iterating through the training and dataset and checking if a 'calc' feature is the feature included in the string, if so then it drops the column or feature from the datasets. The following figure is the code for removing these features:

```

def feature_removal(train_df, test_df):
    print("Removing features...")

    #Remove calc features - not useful
    for col in train_df:
        if 'calc' in col:
            train_df.drop([col], axis=1, inplace=True)
            test_df.drop([col], axis=1, inplace=True)

    return (train_df, test_df)

```

Figure 4.7 - Removing 'calc' features

After the removal of these features I then created a random forest classifier and ran the function *feature\_importances()* to determine the best features in terms of performance, this however gave non-informative results and therefore I had to try a different method. The following figure operates by calculating the Gini score for each feature individually on the training data:

```

f_score = {}
for f in train_cols.columns:
    f_score[f] = abs(calculate_gini.gini_normalized(train_target.values, train_cols[f].values))

i = 0 #index of feature ranking
for key, value in sorted(f_score.items(), key=lambda x: x[1], reverse=True):
    print (i, key, value)
    i+=1

```

Figure 4.8 – Calculating Gini scores for each feature individually

The variable *f\_score* creates a dictionary where the feature names and values will be stored. The loop then iterates through the columns of the training data and adds the feature name to the feature score, with its respective calculated normalized Gini coefficient score. This is

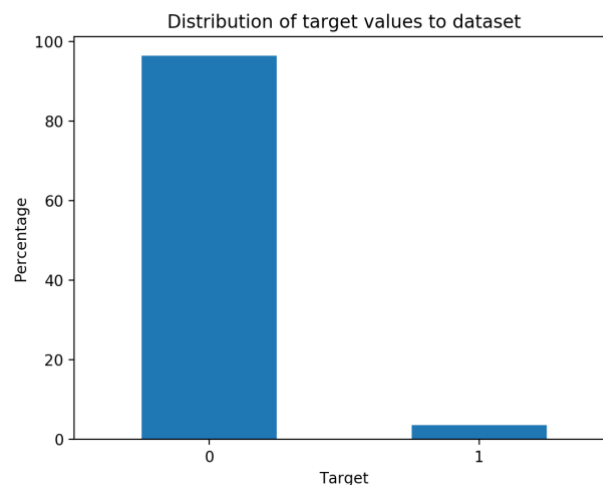
Chapter 4. Implementation

---

done by passing the training dataset target values *train\_target.values* with the columns feature values *train\_cols[f].values*. From these results I decided to also drop three poorly performing binary features in the file *feature\_selection*, this is because it will prioritise the features which perform better that are the most effective features for the classifiers to train with.

#### 4.4. Cross-Validation and Imbalance Learning

In order to determine whether the learning algorithms would overfit the data, I first had to visualise the quantity of each class (class zero and class one) compared to the total of the target variable. From the following figure it is clear that there is a bias towards class zero (no claim made) with a majority percentage of over 95%, compared to class one (claim made) with a minority percentage of less than 10%:



*Figure 4.9 - A histogram of the imbalanced distribution of target values*

The learning algorithms ability to generalise and prevent overfitting is simple to test through cross validation. For all models I have implemented a stratified 'k' fold which splits the training data into five folds, this means there are now four equally sized training sets and one validation testing set which the trained data can evaluate the predictive performance on. Stratified 'k' fold has been used instead of the generic 'k' fold as it will attempt to replicate the distribution of target values for each fold, this will help generalisation. I have used oversampling within each 'k' fold in order to keep consistency with the main training

#### Chapter 4. Implementation

---

and testing dataset, this will reduce overfitting and leading to poor generalisation. A random oversampling has been carried out rather than a stratified sampling method, this is due to the fact that the computational power and time was too significant. The following code snippet represents how I implemented cross validation and over sampling:

```

skf = StratifiedKFold(n_splits=5, shuffle=True)
skf.get_n_splits(X, y)

for train_index, test_index in skf.split(X, y):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    ros = RandomOverSampler(random_state=42)
    X_res, y_res = ros.fit_resample(X_train, y_train)

```

*Figure 4.10 – Cross validation with random over sampling code*

The function `skf.get_n_splits(X,y)` is where the training data and its target values are split five times. There is then a loop to iterate through each split and assign a range of training data, testing data and target values. The function `ros.fit_resample(X_train,y_train)` essentially uses the random over sampler to create more values of class one, to reduce the bias towards class zero and balance the data. It then fits this data to two new instance of the folds training data and target values.

There is a cross validation score which determines the generalisation ability of the model, this is calculated by calculating the mean or average of the Gini scores from each fold. Each this average is calculated from a variable called *results*, which stores the Gini score from each fold.

## 4.5. Classification and Evaluation for Models

Classification is required in order for the learning algorithm to determine which class an individual's data belongs to, there are two possible classes, class '1' represents a claim being made

and class '0' represents no claim being made. Figure 4.11 describes the approach in training a model and then producing predictions for the target variable.

```
model = LogisticRegression()
model.fit(X_res, y_res)

y_pred = model.predict_proba(X_test)
y_pred = y_pred[:,1]
y_cm_pred = model.predict(X_test)

gini_score = calculate_gini.gini_normalized(y_test,y_pred)
matrix = confusion_matrix(y_test, y_cm_pred)

print("Fold",i,"\n")
i+=1
print("ROC score =",roc_auc_score(y_test, y_pred))
print("Gini Score =",gini_score)
print("matrix = \n",matrix)

results.append(gini_score)
cv_score = sum(results)/len(results)
```

Figure 4.11 – Code to produce classifications

The method of implementing classification uses the same approach for each classifier. The variable called *model* describes the learning algorithm (in this case logistic regression), in this figure to keep the code visualisation simple there are no parameters. This model must then train the data using the function *model.fit()*, the parameters used in this function are the training dataset and its target values resampled using oversampling. The variable *y\_pred* will use the function *predict\_proba()* passing the training data as the parameter, this works by assigning itself an array containing the probabilities of obtaining a class '0', as well as the probabilities of obtaining the class '1'. Then the function *y\_pred[:,1]* uses indexing to store the list of only the probabilities of obtaining class '1'. Next the function *predict()* is used instead of the function *predict\_proba()* to store the values needed for the confusion matrix, the difference with the *predict()* function is that It only outputs the predicted class, this is needed later for the confusion matrix to compare with the original target classes.

## Chapter 4. Implementation

---

Once the predictions have been produced there needs to be evaluation metrics to test their accuracy. My solution implements the normalized Gini coefficient, the area under the curve



method as well as the confusion matrix. I have placed the calculation for the normalized Gini coefficient inside the package *calculate\_gini*, an advanced and recommended calculation method by Kaggle was used from the Kaggle website [46]. The function *gini\_normalized()* takes in the original (true target values) and predicted target value probabilities as parameters, it then outputs the overall prediction score. The function *roc\_auc\_score* uses the same approach. Lastly the *confusion\_matrix()* function is used to output the confusion matrix by taking in the parameters of the original target values with the predicted target values.

## 4.6. Fine Tuning the XGBoost Model

Due to the large size of the training and testing datasets, the computational effort and time were quite long and because of this I took a manual parameter selection approach over the period of the project timeline. If I were to use a combined random and grid search to select parameters, it would have taken too long to retrieve results due to the volume of different parameter combinations which are possible. My manual approach involved selecting recommended parameters based on various sources online, each parameter has a recommended range of values to choose from to see if it improves the models performance.

I have taken a manual approach on parameter choices for the random forest and logistic regression models, however the priority has been on XGBoost as it yielded the best results from each test I had carried out.

The parameter combinations I used for XGBoost came from an analytics source [47]. This provided me some knowledge on which parameters to select for general and booster parameters:

- **n\_estimators** [200,400,800]: this specifies the number of trees to build before taking the average of the predictions, the higher this value the better the performance (dependent on the computer computation ability).

- **eta** [0.1,0.2,0.3]: this helps make the model perform more robust by shrinking the weights at each step.
- **min\_child\_weight** [1,5,8]: this parameter sets the minimum weights of all observations in a child node, It is used to control overfitting.
- **max\_depth** [3,6,10]: this sets the maximum depth of the tree, the final value six was used after testing the results in each 'k' fold. This variable is important for controlling overfitting the data.
- **scale\_pos\_weight** [1,1.5,3]: this variable is useful for faster convergence, it is useful as there is a high class imbalance where the majority points towards class '0'.
- **colsample\_bytree** [0.5,0.7,0.9]: this specifies the number of columns to be random samples for each trees.
- **subsample** [0.5,1,1.8]: this specifies the number of observations to be random samples for each tree, lower values are more stable and prevent overfitting however if it is too small it can underfit the data.
- **gamma** [1,3,5]: nodes are split only if It produces a positive reduction in the loss function, the gamma value determines the minimum loss reduction required to make a split. This variable is used for robustness and stability.

The final parameters which were used are in the following code figure:

```
model = XGBClassifier(
    n_estimators=400,
    max_depth=4,
    eta=0.05,
    learning_rate=0.08,
    subsample=0.9,
    min_child_weight=6,
    colsample_bytree=0.9,
    scale_pos_weight=1.6,
    gamma=9
)
```

*Figure 4.12 – XGBoost classifier parameters*

---

## Results & Evaluation

---

This section will be comparing the results obtained from each classifier and comparing these results according to the functional and non-functional requirements to see if it satisfies Porto Seguro's requirements. For each classifier it will be tested by following a structure. Confusion matrices, roc graphs and Gini scores will be compared from the initial classifiers performance compared to its final best state. Then the classifiers will be compared amongst each other to show why XGBoost was chosen as the final classifier. These tests indicate their need to test the model's predictive capacity, as well as provide insight as to why some classifiers may have performed better to their initial state.

### 5.1. Satisfying Pre-Processing Requirements

Before any learning algorithm can train data, the data must be imported using the pandas library, and the training and testing datasets must be pre-processed so the computer is able to read the data. There are two main functional requirements and one non-functional requirement this must satisfy.

#### 1. The system can load in the training and test datasets.

- **Acceptance Criteria:** Using the pandas library, the system recognises the data and can output simple functions such as *df.info()* or *df.head()*.

This requirement has been met as it is the basis of the project for being able to produce predictions and train a learning algorithm. The following figure shows the output of the function *df.head()*, this shows the first five driver instances and their anonymised personal information indicated by their feature names, for example *ps\_ind\_01* represents a value referring to the individual such as their name. This figure also provides the binary target feature in the training data, this acts as the

label which represents the class '0' as no claim being produced and the class '1' of a claim being produced.

## Chapter 5. Results & Evaluation

---

```
df_train.head()
  id  target  ps_ind_01  ...  ps_calc_18_bin  ps_calc_19_bin  ps_calc_20_bin
0   7       0         2  ...              0              0              1
1   9       0         1  ...              0              1              0
2  13       0         5  ...              0              1              0
3  16       0         0  ...              0              0              0
4  17       0         0  ...              1              1              0

[5 rows x 59 columns]

df_test.head()
  id  ps_ind_01  ps_ind_02_cat  ...  ps_calc_18_bin  ps_calc_19_bin  ps_calc_20_bin
0   0         0              1.0  ...              0              0              1
1   1         4              2.0  ...              1              0              1
2   2         5              1.0  ...              0              0              0
3   3         0              1.0  ...              0              0              0
4   4         5              1.0  ...              0              0              1
```

Figure 5.1 – Output of function `df.head()` loading the training and test datasets

Now that the training and testing datasets are imported into the solution, The next requirement is that the data is pre-processed. This includes: handling missing values, ensuring outliers do not impact the predictive performance and one-hot encoding categorical features for improved performance and the ability of all the learning algorithms to be able to read and manipulate the data.

### 2. Both the training and test datasets are pre-processed.

- **Acceptance criteria:** All models being: logistic regression, random forest and XGBoost are able to operate and run the data for their algorithms.

#### Handling missing values

The first way to satisfy this acceptance criteria is by handling the missing values. As described earlier in the implementation section there is a function called `handle_missing_values()`, this will iterate through each feature searching for missing values. Features which are either binary or

categorical have their values imputed with the mode of that feature's values, features which are continuous have their values imputed with the mean of that feature's values. To demonstrate that the feature values have been imputed with another value, I created a function `plot_missing_values()` which checks if there are any missing values in the dataset. The following figures represent the plotted missing data,

## Chapter 5. Results & Evaluation

---

followed by the missing value imputation and then lastly plotting the missing data again to show there are not any missing values for both the training dataset and the testing datasets:

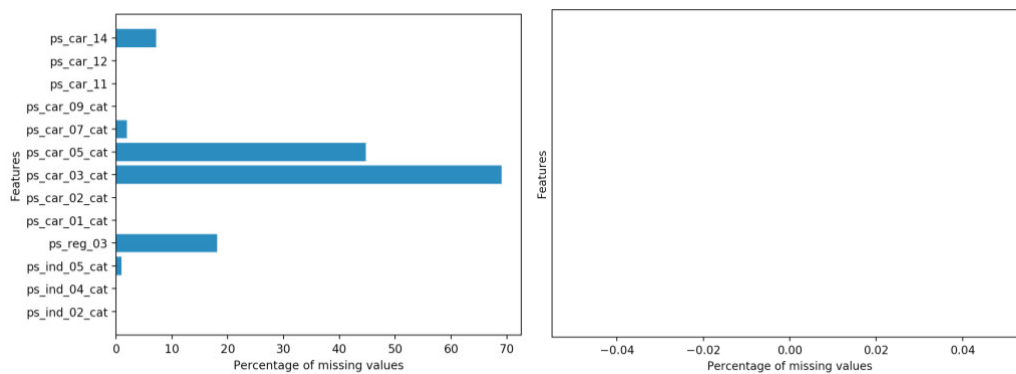


Figure 5.2 – Evidence of training data missing values being imputed

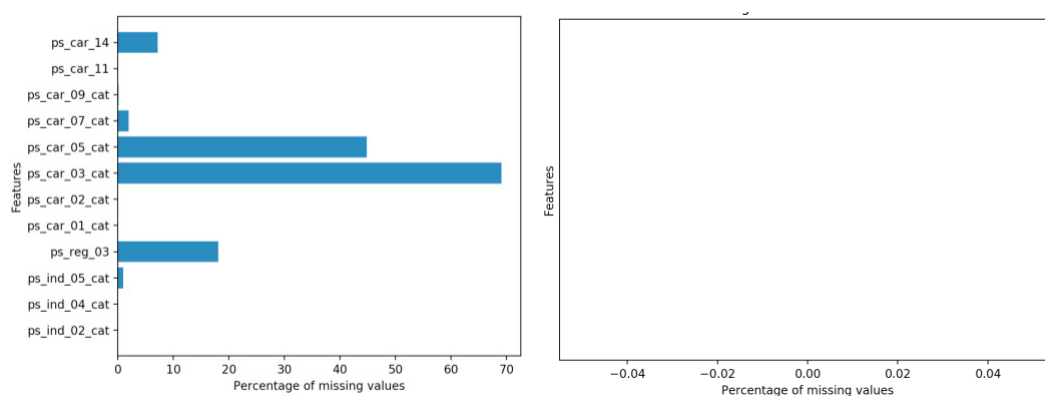


Figure 5.3 – Evidence of testing data missing values being imputed

### One-hot encoding categorical data

One-hot encoding is necessary for learning algorithms to be able to read and manipulate the data. Figure 5.4 receives its output by iterating through all the features in both the training and testing datasets, and then performing the pandas function `get_dummies()` on the feature:

```
feature = ps_ind_02_cat
1.0    432075
2.0    123573
3.0     28186
4.0     11378
Name: ps_ind_02_cat, dtype: int64
```

	ps_ind_02_cat_1.0	ps_ind_02_cat_2.0	ps_ind_02_cat_3.0	ps_ind_02_cat_4.0
0	0	1	0	0
1	1	0	0	0
2	0	0	0	1
3	1	0	0	0
4	0	1	0	0

*Figure 5.4 – Evidence of one-hot encoded features*

This example of one-hot encoding applies for all other features, it operates on the integer encoded values. In this case the feature is `ps_ind_02_cat` and it has four different possible values, as a result from the one-hot encoding the output is now four separate binary encoded features. These features are then added to the training and testing datasets respectively, and the original feature is dropped as it is no longer needed. This figure satisfies this part of the requirement due to these reasons.

### Feature scaling

Feature scaling is required to make sure the predictive performance of all learning algorithms perform effectively, I have implemented a min-max scaling package however when I tested the Gini score before applying the scaling and then after, the Gini score results had either none or minimal difference. Due to this I have commented out the functions which call this package inside the main file *main.py*. I can only make the assumption that the reasoning behind the minimal evaluation scoring, is that Porto Seguro have done some pre-processing of their own for the continuous values. The feature scaling file is still useful to have stored, in case in future there is further testing on different learning algorithms and it can be used

## Chapter 5. Results & Evaluation

---

to suppress outliers impacting learning performance, due to this the file contributes towards the non-functional requirement of having a re-usable system.

### 5.2. Individual Classifier Performance

To distinguish the difference between a well performing learning algorithm and a poorly performing algorithm, I will first show the results obtained from each learning algorithm when they were first tested on the data. These initially classified results will be briefly analysed according to the functional and non-functional requirements, after displaying the final classifier results I will compare the difference in results and justify the need for the tests carried out. The critical functional requirements these tests focus on are that:

1. **Each classifier can produce a 'claim' or 'no claim' prediction.**
  - **Acceptance criteria:** Claims classified as: claim equal to '1' and a no claim equal to '0'.
2. **The system needs to be able to appropriately evaluate a good prediction and overall solution.**
  - **Acceptance criteria** is relative to the following:
  - Confusion matrix: The proportion of the confusion matrix has higher true positive and true negative values, than false positive and false negative values.

- Normalized Gini coefficient: As described by Porto Seguro, a perfect model scores 0.5 meaning every prediction is correct, and a score of 0 implies the model randomly guesses predictions values, the minimum Gini score should be at least 0.24 [3].
- Area Under the Curve score: similar to the Gini score, this ranges between 0 for random guessing and 1 for a perfect score, the minimum score should be 0.62. Using the ROC curve diagrams, it provides a clear visualisation to test and determine the model's predictive capabilities.

## Chapter 5. Results & Evaluation

---

### 5.2.1 Random Forest Classifier

The first learning algorithm which was trained on the training data was a random forest classifier. This random forest classifier fails to meet the first requirement because it can only correctly classify predictions to class '0', whereas there are no correct predictions to class '1'. This partially meets requirement two because the majority of class '0' predictions have been classified correctly, however the model cannot appropriately classify class '1' and therefore it is neither producing good predictions for this class nor is the overall solution appropriate. This is demonstrated by the confusion matrix in figure 5.5:

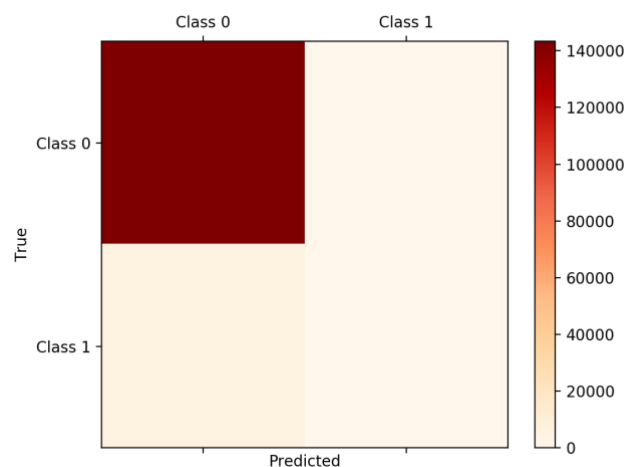
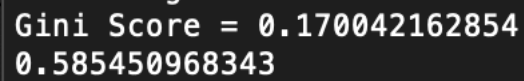


Figure 5.5 - Initial random forest classifier confusion matrix



The next test is to generate the normalized Gini coefficient score, an appropriate score to indicate a good learning algorithm on the data should provide a minimum score of 0.24. The auc score will also need to be calculated, a good score should be above 0.62 to indicate the model can predict classes well. Figure 5.6 represents the scores acquired for Gini and auc:



```
Gini Score = 0.170042162854  
0.585450968343
```

*Figure 5.6 – Basic random forest Gini and auc score*

These results clearly indicate the solution does not meet any of the requirements, as the Gini and auc scores are less than the acclaimed base scores.

Chapter 5. Results & Evaluation

---

### **Final random forest results**

I then built upon this solution to include cross-validation and random over sampling on the minority class '1' representing a no claim class. The following results are shown in figure 5.7:

```

Fold 1

Gini Score = 0.246677287884
0.623338643942
matrix =
[[109136  5568]
 [  3842   497]]
fitting
fitted
Fold 2

Gini Score = 0.271559023221
0.635779511611
matrix =
[[109564  5140]
 [  3888   451]]
fitting
fitted
Fold 3

Gini Score = 0.262031943378
0.631015971689
matrix =
[[109549  5155]
 [  3841   497]]
fitting
fitted
Fold 4

Gini Score = 0.272142791444
0.636071395722
matrix =
[[109554  5149]
 [  3840   499]]
fitting
fitted
Fold 5

Gini Score = 0.247225006087
0.623612503044
matrix =
[[109285  5418]
 [  3888   451]]
— 233.44083259503046 minutes —
0.259927210403

```

Figure 5.7 – Final random forest Gini and auc scores

## Chapter 5. Results & Evaluation

---

This optimised model clearly satisfies the acceptance criteria at a far higher level. With regards to the first requirement on average you can see from the confusion matrix results that almost all of the class '0' predictions have been predicted correctly, and although there are more incorrectly

classified class '1' predictions, there are a good number of classified predictions for this class considering the noise in the dataset.

With regards to the second requirement you can see firstly that the proportion of confusion matrix results, are representative of the data. The normalized Gini coefficient receives a score of 0.2599 which is higher than the base recommended result of 0.24. Lastly the auc score for each fold averages to a value of 0.63, as 0.63 is larger than 0.62 we can conclude the model appropriately evaluates a good prediction and overall solution.

### 5.2.2 XGBoost Classifier

The next learning algorithm which was trained on the data was the XGBoost classifier. This learning algorithm has mostly failed the first requirement as it is only capable of predicting the class '1' class once, and incorrectly classifying it four times. The class '0' predictions however have been classified significantly well, from these results however we can conclude the base model of XGBoost is incapable of classifying a claim or a no claim class, from this we can infer it will not be able to generalize on unseen data as it will overfit on the class '0' due to its large majority of predictions. Figure 5.8 shows the results of the initial XGBoost:

```
ROC score = 0.632397742859
Gini Score = 0.264795483709
matrix =
[[114699    4]
 [  4338    1]]
```

*Figure 5.8 - Initial XGBoost classifier confusion matrix*

With regards to whether or not this initial XGBoost can produce good predictions for a good overall solution, it is possible but unlikely due to the lack of predicted class '1' classes. The

Gini and roc score are fairly good results, however this initial XGBoost model is unlikely to generalize well on new unseen test data.

### Final XGBoost results

```
Performing XGBoost
Fold 1
ROC score = 0.632575932349
Gini Score = 0.265151866708
matrix =
[[28847 85857]
 [ 573  3766]]
Fold 2
ROC score = 0.634274188098
Gini Score = 0.268548394278
matrix =
[[28957 85747]
 [ 575  3764]]
Fold 3
ROC score = 0.639932529687
Gini Score = 0.279865031238
matrix =
[[28829 85875]
 [ 538  3800]]
Fold 4
ROC score = 0.641229740103
Gini Score = 0.282459448057
matrix =
[[28329 86374]
 [ 530  3809]]
Fold 5
ROC score = 0.638650022399
Gini Score = 0.277300056854
matrix =
[[29463 85240]
 [ 552  3787]]
--- 33.68544221719106 minutes ---
0.274664959427
```

Figure 5.9 – Final XGBoost Gini and auc scores

As seen by the final results it satisfies the acceptance criteria because it is able to distinguish between the class '0' and the class '1'. This is demonstrated by the confusion matrix results in each fold where the majority of class '0' predictions are classified correctly, the class '1' predictions have a good number of correctly classified predictions, there is a larger number

of incorrectly classified class '0' predictions but this will be the case considering the dataset is noisy. This may have been able to be suppressed given there was a stratified approach to identifying class predictions, however due to the complexity of computational effort I would have to explore this in future.

With regards to the second functional requirement, the 'k' folds comes to an average Gini score of 0.2747. This is higher than the requirement Gini score of 0.24. The auc score comes to an average of 0.637, which beats the base requirement score of 0.62. From these results we can conclude they provide great predictions in terms of the evaluation metric scores, and therefore provides a good solution for a model that can generalize well for unseen data.

### 5.2.3 Logistic Regression

The approach towards training the logistic regression classifier was slightly different because the parameters used provided similar Gini scores and auc scores compared to the base classifier. Therefore for this example I will evaluate the final logistic regression model result.

As shown by the results in the next figure, this model has shown to satisfy the non-functional requirements of being robust and providing reliable results as shown by the average of the 'k' folds, where it is a good score.

### Final logistic regression results

```
Fold 1
ROC score = 0.625991033454
Gini Score = 0.251982066907
matrix =
[[71688 43016]
 [ 1924  2415]]
Fold 2
ROC score = 0.63154993109
Gini Score = 0.263099862179
matrix =
[[71087 43617]
 [ 1879  2460]]
Fold 3
ROC score = 0.625559065221
Gini Score = 0.251118130441
matrix =
[[71315 43389]
 [ 1895  2443]]
Fold 4
ROC score = 0.628698600556
Gini Score = 0.257397201113
matrix =
[[71575 43128]
 [ 1903  2436]]
Fold 5
ROC score = 0.622288675687
Gini Score = 0.244577351373
matrix =
[[71076 43627]
 [ 1953  2386]]
— 5.3840694983800255 minutes
0.253634922403
```

Figure 5.10 – Final logistic regression Gini and auc scores

For requirement one and from these results, we can see that the model can successfully classify class '0' and class '1', similarly to the other models it cannot predict the 'class 1' as accurately because of the data imbalance and noise. The average Gini score retrieved is 0.2536 which is

greater than the base value of 0.24, the auc or roc score is 0.625 on average which means the results satisfy these requirements.

### 5.3. Testing Generalization

Referring back to chapter 3.2 there are functional and non-functional requirements which are relevant towards testing the generalization ability of the solution. Each relevant requirement will be tested against its acceptance criteria to determine whether or not the solution satisfies these requirements. In order to easily understand how generalization has been accomplished, I will train a simple random forest classifier with no sampling methods implemented or cross validation, these results will then be compared to a solution with cross validation and over sampling.

#### 1. The system needs to be able to handle a data imbalance.

- **Acceptance Criteria:** The proportion of the confusion matrix has higher true positive and true negative values, than false positive and false negative values.

This means that when predicting class values, the number of predictions are not biased towards any class type. To illustrate an example of data imbalance taking place, the following figure represents the results of a simple random forest classifier with no modifications:

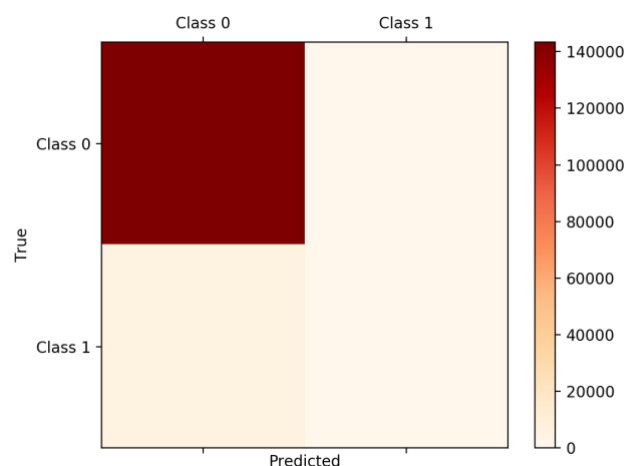


Figure 5.11 - Confusion matrix for a simple random forest classifier

As shown by the colour bar the basic random forest classifier does not meet the acceptance criteria, this is because the results indicate that the majority of predictions were classified as class '0' indicating that no claim was made, as well as the fact that class '1' made no correct classification of a claim being made at all. This test is required because if the solution cannot handle a data imbalance, it will have an immediate impact on its ability to make reliable predictions on unseen test data, this subsequently leads on to requirement two.

### 2. The system needs to be able to produce accurate predictions similarly on training data, to unseen data.

- **Acceptance criteria:** For each 'k' fold, the Gini score and the confusion matrix true positive and true negative values are within a small margin. The Gini score of the test dataset held on Kaggle should also hold a similar score to the folds.

To determine if this requirement is satisfied I tested the model on unseen data provided by Porto Seguro's data store in Kaggle, by submitting a file *submission.csv*. This file contains all the predictions made on the test dataset. The predictions are tested on two external sections of the test dataset being the public and private test data, the public test data contains 30% of the total test data and the private test data contains the remaining 70%. The following figure represents the scores that the basic random forest classifier produced:

---

submission.csv	0.07192	0.08166
----------------	---------	---------

---

*Figure 5.12- First public and private test data Gini scores*

Due to the fact that the maximum Gini score possible is '0.5' for a perfect score, and '0' for random guessing, these results clearly indicate that the model has severely underfitted the data and has not



made the acceptance criteria. The model cannot make classifications properly and therefore the predictions are inaccurate and unreliable.

### **3. The system should be robust**

#### Chapter 5. Results & Evaluation

---

This test also indicates the model is not robust and does not satisfy the non-functional requirement, this is because both the performance and predictions significantly changed in a negative way.

To summarise, a model that does not incorporate cross validation or any sampling will result in the data remaining imbalanced. A machine learning algorithm will not perform well with Imbalanced data as shown by these results. To be able to satisfy these functional and non-functional requirements, the final solution implements cross validation with five 'k' folds. Cross validation is needed in the solution to test its generalization accuracy, this means that there are four training data sets being trained on one test data set. If the results are similar and there is a relatively good prediction it means that it should work well on unseen test data. To enhance cross validation's potential, I have implemented random oversampling for each fold. Oversampling will make sure there are more class '1' examples so the likelihood the model will predict a class '1' increases. The following figure represents the results from a simple random forest, however this time there is cross validation and random oversampling implemented:

```

Fold 1
ROC score = 0.615443857281
Gini Score = 0.230887732645
matrix =
[[80219 34485]
 [ 2314  2025]]

Fold 2
ROC score = 0.615455421461
Gini Score = 0.230910830867
matrix =
[[79245 35459]
 [ 2259  2080]]

Fold 3
ROC score = 0.615588157722
Gini Score = 0.231176317454
matrix =
[[79793 34911]
 [ 2311  2027]]

Fold 4
ROC score = 0.617016295903
Gini Score = 0.234032587788
matrix =
[[79111 35592]
 [ 2270  2069]]

Fold 5
ROC score = 0.612308617908
Gini Score = 0.224617235815
matrix =
[[79862 34841]
 [ 2290  2049]]
--- 63.40536116361618 minutes ---

```

Figure 5.13 – Cross validation and random sampling results

## Chapter 5. Results & Evaluation

---

From these five folds the results demonstrate that the enhanced model is capable of handling a data imbalance and to prove this further, the next figure represents the model being tested on the Kaggle test data:

submission.csv	0.23588	0.22907
----------------	---------	---------

Figure 5.14 - Second public and private test data Gini scores

### **Satisfying requirement one**

This solution satisfied requirement one because the results from the confusion matrix have a high rate of true negative predictions and a far better result of true positive values, and although there are quite a high number of false positive predictions, the Gini score has significantly improved. From looking at the confusion matrix you can see that there is not a complete bias majority for class '0' and that class '1' is receiving more predictions.

### **Satisfying requirement two**

The solution illustrates how each Gini score for each 'k' fold holds a similar value interval, the average can be calculated as '0.2303'. This essentially means that for each training data set which was tested on the test data set, it had received a good average Gini score. Because they are very similar scores it can show the classifier generalises its predictions well, this is also proved by the scores obtained in the Kaggle test data which averages a Gini score of '0.2325'.

### **Satisfying the robustness and reliability requirement**

These last requirements are satisfied by the classifier because the predictions and Gini scores do not vary significantly, as proved by the average of the 'k' folds and the comparison between these fold scores to the Kaggle tested Gini score result. The model can be considered reliable and a good representative classifier for making predictions accurately, due to its consistency.

## Chapter 5. Results & Evaluation

---

To conclude, my solution can now ensure that any classifier trained will not underfit or overfit unseen data it is provided with. The solution will generalize by classifying a claim or a no claim to a good quality. A disadvantage however to this solution could be that the over sampling is randomised instead of using a stratified approach, this however was too computationally intensive to perform and in future could be implemented considering there is more computational power.

### **5.4. Validating the Chosen Classifier**

Due to the fact that the Gini score and auc score are mathematically similar, In order to choose which model will be selected for the final classification model proposal, I will be comparing the auc and Gini scores relative to their visualisations on a receiver operating characteristic graph (roc).

The first figure represents the roc graph for the random forest classifier. Its curve is above the dotted red line which indicates 'random guessing', considering its auc score is 0.61 in this instance it is showing an unreliable score compared to its previously scored auc predictions.

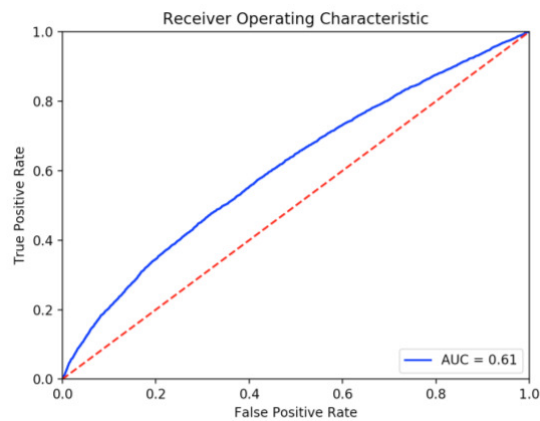


Figure 5.15 - A roc graph for the random forest classifier

## Chapter 5. Results & Evaluation

---

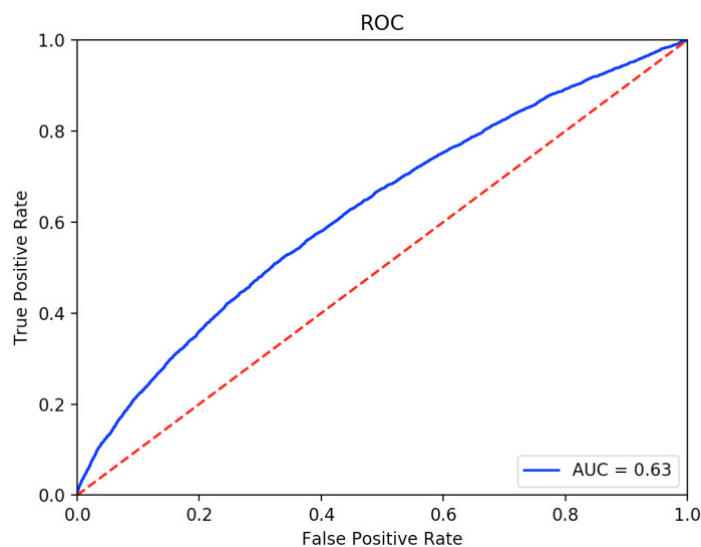


Figure 5.16 - A roc graph for the XGBoost classifier

As shown by this XGBoost classifier, the auc results are better and therefore I decided to use XGBoost as the final classifier.

## 5.5. Conclusion of Results

To conclude the results given from the classifiers produced, it is fair to say that the XGBoost model met the functional and non-functional requirements in the best detail. This is because it could correctly classify the class '0' results very accurately and although there was a range of incorrectly classified class '1' results, it predicted a good quantity of this class considering the noise of this dataset and how heavily imbalanced it is. Comparing the results in each 'k' fold to the random forest classifier, it showed that the output Gini scores were more reliable and this gave the XGBoost model the upper hand of making it more robust and generalizing better on unseen data. The random forest classifier showed promising results, in particular due to the fact that the classification of class '1' appeared to classify and distinguish classes better than the XGBoost model, unfortunately the Gini score average was not as strong in comparison. Lastly the logistic regression model appeared to produce good base results, however in comparison to the random forest and XGBoost models it was out-performed.

## Chapter 5. Results & Evaluation

---

In general I believe the requirements set by Porto Seguro were satisfied as all the acceptance criteria has been met for the functional and non-functional requirements. The XGBoost model is robust, reliable and the code used to generate this model (as well as the other models) has been structured with a package system so the components are re-usable.

---

## CHAPTER 6

---

---

### Future Work

---

From carrying out this project I was intrigued by the results I had obtained from using various different learning algorithms. Due to the short time given however, I did not explore in depth all the aspects of machine learning I would have liked to. In future I would like to train unsupervised and semi-supervised models, to determine whether or not their results could be used to improve the insurance claim predictions. There are also certain aspects I would go further in depth for researching and selecting, such as the choice of categorical encoding methods like target encoding,

as well as alternative data imputation methods such as the ‘k nearest neighbour’ algorithm which is more advanced for selecting more realistic replacement values, for values which are missing. The addition of more models for testing and parameter tuning would also be a good way to improve on the project in future, in particular neural networks because they can yield great results on various projects.

It would also be a good idea to train a live system (perhaps a batch system) which takes in batches of new customer information, trains the data and then produces predictions automatically within a reasonable time. With a live system the data can thereby increase with more customer information and this could potentially lead to more accurate predictions if there is more data for the learning algorithms to work with.

Lastly due to the fact the data was anonymised I was unable to make discoveries about the features meaning in its real world applications, I was only able to make assumptions with regards to common driving statistics. If there had been more time I would have liked to try and specifically cross-reference feature values to certain statistics with more research, for example a feature that represents a car colour with regards to vehicle accidents and producing claims.

---

## CHAPTER 7

---

---

### Conclusion

---

The overall aims of this project were to produce a machine learning solution for predicting whether or not a new policy holder would produce an insurance claim in the next year. This is in the hopes that it can make automotive insurance more accessible to more drivers, through a solution that produces a more accurate prediction.

To make predictions it would have to effectively apply the feature information such as the customer's location or region, as well as their personal and vehicle information. While it may have been slightly difficult to directly reference real-world features that were anonymized, this was possible to overcome by analysing feature's importance through a correlation matrix and testing features normalized Gini coefficient score's individually on the dataset. Using feature selection and training multiple classifiers I was able to produce a solution that was more complex than a simple classifier with parameters, and provide reliable claim predictions. It is important to factor in that the datasets provided were very noisy and imbalanced, and so the evaluation scores would not be possible to retrieve a nearly perfect score.

This solution provides a good approach, however there are more methods and different approaches which could have been chosen to produce better predictions and lead to providing Porto Seguro an improved solution, to help them tailor their prices for their policy holders which would have been researched further given there was more time. My solution used boosting ensemble methods, however an alternative could have been stacking where each model combines its predictions for multiple levels or stages, this could produce an improved Gini score.

---

## CHAPTER 8

---

---

### Project Reflection

---

This project has come to be one of my most challenging projects I have done. I undertook this project because I am highly interested in machine learning and its real-world applications and possibilities. In the beginning of undertaking this project I found it difficult trying to understand



all the different concepts of machine learning, as I had no prior knowledge to the topic. I found that many sources of information did not explain the concepts of machine learning too clearly, especially for handling imbalanced data and distinguishing between evaluation and generalization techniques. Due to this I spent too much of my time researching the “best” methods to solve certain problems, the advantage of this was that I believe my knowledge of machine learning concepts and applications has very significantly developed, the disadvantage is that from my research I was reluctant to implement code as quickly as I should have to provide more time to write the report. Another challenge was interpreting the data at first glance, I was under the assumption that the data needed pre-processing such as feature scaling due to the range of values, this however was misinterpreted as categorical data which is different to continuous data.

I would say that my time management for prioritising tasks could be improved as although I was making brief notes as I was coding certain sections, I should have been developing code more quickly to get simple classifier predictions and then build upon it rather than trying to solve the problem all in one go. In particular the computational time for training a model became considerably large for my computer, to be able to handle in a fairly reasonable time. After acknowledging this from a few long attempts of training, I switched to the free ‘Google Colab’ service and made use of their GPU’s. This was confusing to implement my Python code at first, however it saved me a great amount of time testing my classifiers prediction accuracies due to its computational power.

From research and applying code directly myself, I understood the value of separating certain sections or functions of code, for example one hot encoding categorical variables or computing the mean values for a pandas data column. This means the code can be simply adapted to other projects and become re-usable. To conclude I believe that I discovered various ways of solving problems with machine learning, and I can learn from my mistakes to improve on my next project.

## References

---

[42] Badr, Will. (2019). *6 Different Ways to Compensate for Missing Values In a Dataset (Data Imputation with examples)*. Accessed 20/05/2020, from <https://towardsdatascience.com/6-different-ways-to-compensate-for-missing-values-data-imputation-with-examples-6022d9ca0779>

[5] Bobriakov, Igor. (2018). *Top 10 Data Science Use Cases in Insurance*. Accessed 12/02/2020, from <https://medium.com/activewizards-machine-learning-company/top-10-data-science-use-cases-in-insurance-8cade8a13ee1>

[44] Brownlee, Jason. (2020). *Why One-Hot Encode Data In Machine Learning*. Accessed 02/05/2020, from <https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/>

[10] Columbus, Louis. (2020). *Roundup Of Machine Learning Forecasts And Market Estimates*. Accessed 12/05/2020, from <https://www.forbes.com/sites/louiscolumbus/2020/01/19/roundup-of-machine-learning-forecasts-and-market-estimates-2020/#7ff2b10f5c02>

[8] Compare The Market. (2020). *Why is my car insurance so expensive?* Accessed 2/04/2020, from <https://www.comparethemarket.com/car-insurance/content/why-is-car-insurance-expensive/>

[4] Confused.com. (2020). *The Gender Gap in 100 Drivers*. Accessed 03/02/2020 from, <https://www.confused.com/car-insurance/gender-gap-100-drivers>

[46] Extremely Fast Gini Computation. Accessed 15/05/2020, from <https://www.kaggle.com/cmpmml/extremely-fast-gini-computation>

- [12] Faggella, Daniel. (2020). *What is Machine Learning*. Accessed 10/03/2020, from <https://emerj.com/ai-glossary-terms/what-is-machine-learning/>
- [31] Fawcett, Tom. (2005). *An introduction to ROC analysis*. Accessed 19/05/2020, from <https://people.inf.elte.hu/kiss/13dwhdm/roc.pdf>
- [45] Flewelling, Paul. (2018). *The Agile developer's handbook: get more value from your software development: get the best out of the Agile methodology*. Accessed 17/03/2020.
- [23] Friedman, N., Geiger, D. & Goldszmidt, M. (1997). *Bayesian Network Classifiers*. Accessed 20/04/2020, from <https://doi.org/10.1023/A:1007465528199>
- [29] Ganganwar, V. (2012). *An overview of classification algorithms for imbalanced datasets*. Accessed 17/05/2020, from [https://www.researchgate.net/publication/292018027\\_An\\_overview\\_of\\_classification\\_algorithms\\_for\\_imbalanced\\_datasets](https://www.researchgate.net/publication/292018027_An_overview_of_classification_algorithms_for_imbalanced_datasets)
- [9] Géron, Aurélien. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: concepts, tools, and techniques to build intelligent systems*. Accessed 06/02/2020.
- [14] Gonçalves I., Silva S., Melo J.B., Carreiras J.M.B. (2012). *Random Sampling Technique for Overfitting Control in Genetic Programming*. Accessed 13/05/2020, from [https://doi.org/10.1007/978-3-642-29139-5\\_19](https://doi.org/10.1007/978-3-642-29139-5_19)
- [17] Google Developer. (2020). *Text classification*. Accessed 14/05/2020, from <https://developers.google.com/machine-learning/guides/text-classification>

- [19] Grira, N., Crucianu, M. and Boujemaa, N. (2004). *Unsupervised and semi-supervised clustering: a brief survey. A review of machine learning techniques for processing multimedia content*. Accessed 14/05/2020, from <http://cedric.cnam.fr/~crucianm/src/BriefSurveyClustering.pdf>
- [2] Gusner, Penny. (2018). *13 things that affect your car insurance*. Accessed 03/02/2020, from <https://www.insure.com/car-insurance/car-insurance-factors.html#claims>
- [27] Hossin, M. and Sulaiman, M.N. (2015). *A review on evaluation metrics for data classification evaluations*. Accessed 16/05/2020, from [https://www.researchgate.net/publication/275224157\\_A\\_Review\\_on\\_Evaluation\\_Metrics\\_for\\_Data\\_Classification\\_Evaluations](https://www.researchgate.net/publication/275224157_A_Review_on_Evaluation_Metrics_for_Data_Classification_Evaluations)
- [43] Imbalanced-learn. Accessed 01/05/2020, from <https://imbalanced-learn.readthedocs.io/en/stable/>
- [28] Jabbar, H. and Khan, R.Z., 2015. *Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study)*. Accessed 17/05/2020, from [https://www.researchgate.net/profile/Haider\\_Allamy/publication/295198699\\_METHODS\\_TO\\_AVOID\\_OVER-FITTING\\_AND\\_UNDER-FITTING\\_IN\\_SUPERVISED\\_MACHINE\\_LEARNING\\_COMPARATIVE\\_STUDY/links/56c8253f08aee3cee53a3707.pdf](https://www.researchgate.net/profile/Haider_Allamy/publication/295198699_METHODS_TO_AVOID_OVER-FITTING_AND_UNDER-FITTING_IN_SUPERVISED_MACHINE_LEARNING_COMPARATIVE_STUDY/links/56c8253f08aee3cee53a3707.pdf)

[47] Jain, Aarshay. (2016). Complete Guide to Parameter tuning in XGBoost with codes in Python. Accessed 20/05/2020, from <https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/>

[13] Jonathan Schmidt, Mário R. G. Marques, Silvana Botti & Miguel A. L. Marques. (2019). *Recent advances and applications of machine learning in solid-state materials science*. Accessed 12/05/2020, from <https://www.nature.com/articles/s41524-019-0221-0>

[1] Kagan, Julia. (2018). *Introduction to Auto Insurance*. Accessed 03/02/2020, from <https://www.investopedia.com/terms/a/auto-insurance.asp>

[38] K. C. Dewi, H. Murfi, S. Abdullah. (2019). *Analysis Accuracy of Random Forest Model for Big Data – A Case Study of Claim Severity Prediction in Car Insurance*. Accessed 08/02/2020, from <https://ieeexplore.ieee.org/abstract/document/8987520>

[16] Kotsiantis, S.B., Zaharakis, I.D. & Pintelas, P.E. (2006). *Machine learning: a review of classification and combining techniques*. Accessed 13/05/2020, from <https://doi.org/10.1007/s10462-007-9052-3>

[22] Kotsiantis, S.B., Zaharakis, I. and Pintelas, P. (2007). *Supervised machine learning: A review of classification techniques*. Accessed 15/05/2020, from [https://books.google.co.uk/books?hl=en&lr=&id=vLiTXDHR\\_sYC&oi=fnd&pg=PA3&dq=machine+learning+classification+algorithms&ots=CYtrxs2Fhl&sig=QBxBE\\_90](https://books.google.co.uk/books?hl=en&lr=&id=vLiTXDHR_sYC&oi=fnd&pg=PA3&dq=machine+learning+classification+algorithms&ots=CYtrxs2Fhl&sig=QBxBE_90)

f5FwJk3O\_OJepCOGnNU&redir\_esc=y#v=onepage&q=machine%20learning%20classification%20algorithms&f=false

[39] Longhao Jing, Wenjing Zhao, Karthik Sharma, Runhua Feng. *Research on Probability-based Learning Application on Car Insurance Data*. Accessed 15/04/2020, from <https://www.atlantis-press.com/proceedings/macmc-17/25888526>

[7] Manuel, Natasha. (2018). *Fraud Detection Using Graph Technology*. Accessed (24/03/2020) from, <https://neo4j.com/blog/fraud-detection-using-graph-technology/>

[33] Matplotlib. Accessed 07/02/2020, from <https://matplotlib.org>

[24] Musa, A.B. *Comparative study on classification performance between support vector machine and logistic regression*. Accessed 22/04/2020, from <https://doi.org/10.1007/s13042-012-0068-x>

[36] numpy. (2020). Accessed 08/02/2020, from <https://numpy.org>

[20] OneModel. (2020). *AI Academy: What is Machine Learning?* Accessed 16/05/2020, from <https://www.onemodel.co/blog/ai-academy-what-is-machine-learning>

[34] Pandas. Accessed 07/02/2020, from <https://pandas.pydata.org>

[3] Porto Seguro. (2018). *Porto Seguro's Safe Driver Prediction*. Accessed 03/02/2020, from <https://www.kaggle.com/c/porto-seguro-safe-driver-prediction/overview>

[40] Porto Seguro. (2018). *Data Description*. Accessed 25/02/2020 from, <https://www.kaggle.com/c/porto-seguro-safe-driver-prediction/data>

[48] Porto Segura Insurance Sectors. (2020). Accessed 20/03/2020, from <https://www.portoseguro.com.br/en/institutional>

[41] Raschka, Sebastian. (2014). *About Feature Scaling and Normalization – and the effect of standardization for machine learning algorithms*. Accessed 20/05/2020, from [https://sebastianraschka.com/Articles/2014\\_about\\_feature\\_scaling.html](https://sebastianraschka.com/Articles/2014_about_feature_scaling.html)

[15] S.M.Weiss, N.Indurkha. (1995). *Rule-based Machine Learning Methods for Functional Prediction*. Accessed 13/05/2020, from <https://doi.org/10.1613/jair.199>

[6] SAS. (2020). *Fraud Prevention*. Accessed 24/03/2020 from, [https://www.sas.com/en\\_gb/insights/fraud/fraud-prevention.html](https://www.sas.com/en_gb/insights/fraud/fraud-prevention.html)

[11] SAS. (2020). *Machine Learning, what it is and why it matters*. Accessed 08/03/2020, from [https://www.sas.com/en\\_gb/insights/analytics/machine-learning.html](https://www.sas.com/en_gb/insights/analytics/machine-learning.html)



- [32] Scikit-learn. Accessed 07/02/2020, from <https://scikit-learn.org/stable/>
- [37] Sennaar, Kumba. (2020). *How America's Top 4 Insurance Companies are Using Machine Learning*. Accessed 20/04/2020, from <https://emerj.com/ai-sector-overviews/machine-learning-at-insurance-companies/>
- [30] Wu Shaomin. (2005). *A scored AUC Metric for Classifier Evaluation and Selection*. Accessed 18/05/2020, from <http://dmip.webs.upv.es/ROCML2005/papers/wuCRC.pdf>
- [35] XGBoost. Accessed 07/02/2020, from <https://xgboost.readthedocs.io/en/latest/index.html>
- [18] Zheng Zijian, Kohavi Ron, Mason Llew. (2001). *Real world performance of association rule algorithms*. Accessed 14/05/2020, from <https://dl.acm.org/doi/abs/10.1145/502512.502572>
- [25] Zhou, Z.H. (2012). *Ensemble methods: foundations and algorithms*. Accessed 23/04/2020, from [https://books.google.co.uk/books?hl=en&lr=&id=BDB50Ev2ur4C&oi=fnd&pg=PP1&dq=ensemble+methods&ots=OxKAHpnORD&sig=lKZzLAd78yBGAW-Z-5o5yjra9ws&redir\\_esc=y#v=onepage&q=ensemble%20methods&f=false](https://books.google.co.uk/books?hl=en&lr=&id=BDB50Ev2ur4C&oi=fnd&pg=PP1&dq=ensemble+methods&ots=OxKAHpnORD&sig=lKZzLAd78yBGAW-Z-5o5yjra9ws&redir_esc=y#v=onepage&q=ensemble%20methods&f=false)
- [26] Zhou, Z.H. (2009). *Ensemble Learning*. Accessed 25/04/2020, from <https://cs.nju.edu.cn/zhoush/zhoush.files/publication/springerEBR09.pdf>

[21] Zhu, X. and Goldberg, A.B. (2009). *Introduction to semi-supervised learning*. Accessed 15/05/2020, from <https://www.morganclaypool.com/doi/abs/10.2200/S00196ED1V01Y200906AIM006?journalCode=aim>