|  |  |
| --- | --- |
|  |  |

Statistical Learning Methods

# Preventive Maintenance

#### Student\_ID: S398713

#### Department: Applied Artificial Intelligence

#### Date: 16/11/2022

|  |  |  |
| --- | --- | --- |
|  |  | Predictive Maintenance  Statistical Learning Methods |

# Aim

The aim of the analysis is to enhance the maintenance operations and planning of time-based preventive maintenance of the aircraft engine. This is supposed to be done by applying statistical learning methods: regression and classification

# Problem Definition

Failure prediction is a major topic in predictive maintenance in many industries. Aircraft manufacturers, OEM’s and end users are highly interested in prediction of component failures during the operation so that they can plan maintenance operations and reduce losses due to the time aircraft has spent on the ground.

Monitoring of the engine health and current condition is based on sensor data analysis and telemetry from the engine sub-systems. It is supposed to promote predictive maintenance by estimating Time-To-Failure (TTF) or remaining Useful Life (RUL) for aircraft components that are currently in-service and may be fully functional at the time of testing.

Based on the measurements from the sensors of the aircraft engine, the developed analysis framework should provide the following predictions, which are the objective of this assignment:

1. Time-To-Failure (TTF) prediction for the engine.
2. Classify which engine will fail in the analysed time period.

# Data Given

The data given consists of training data and testing data for aircraft engines with their simulated time to failure events along with four sensor measurements responsible for the ttf. The dataset was originally taken from NASA C-MAPPS.

* Training data contains of engine id for the 100 aircraft engine, cycle per engine sequence starting from 1 to cycle number where failure happened, sensor measurements of 4 sensors, time to failure of the engines along with a label\_bnc required during classification
* Testing data consists of measurements of 4 sensors performed at a randomly selected cycle of engine operation. TTF is found to be different from the ones found in the training data.

# Exploratory Data Analysis

First, we check if the data has any null values and if the data is clean and upon verification it was found that the data is free of null values. Hence, it’s a clean dataset.

Next step is to understand the data given to us and the first step of understanding data is by visualizing the data given to us. We have various independent variables such as cycle, s1, s2, s3, s4 and the dependant variable in the form of ttf. Let us visualize the data to understand where each and every feature lies.

Histogram

Description automatically generatedA picture containing histogram

Description automatically generated

A picture containing shape

Description automatically generatedChart, histogram

Description automatically generated

Chart

Description automatically generatedHistogram

Description automatically generated with medium confidence

Kdeplot is a Kernel Distribution Estimation Plot which depicts the probability density function of the continuous or non-parametric data variables. We see the range in which these features lie and it’s nature. We also see that cycle and ttf have a similar visualization.

Let us analyse the ttf variable by plotting a boxplot and see what more the dependant variable ttf tells us. Boxplot is an extremely useful way to interpret data as it tells us whether there are any outliers present, if our data is symmetrical, how tightly our data are grouped and if our data is skewed.

Chart

Description automatically generated

The boxplot tells us that the minimum value of our data lies at 0, the lower 25% of our data (Q1) lies between 0 and 50, it also shows us where the median lies (Q2) ,the upper 25% our data (Q3) and the maximum value of the ttf variable. We can also see that there are some outliers present above the maximum value. We can’t just eradicate the outliers as they may have some importance to it and after inspection in the dataset it was found that engine 69 showed these values and that means that it may be a bigger engine meaning that the time taken for the engine to fail will be large. Hence, we can’t remove these outliers present in the ttf variable.

Now let us understand what relation the independent variable and dependant variable have with each other and themselves.

Background pattern

Description automatically generated

From this figure we see that s1 and s3 show a nonlinear relationship with the ttf variable and s2 and s4 show a linear relationship with the ttf variable.

The next step of our exploratory data analysis will be to understand correlation of our independent variables and see whether they are correlated to each other and also corelation between the independent and target variable.

Text

Description automatically generated

Let us use a heatmap to better represent this data.

A picture containing graphical user interface

Description automatically generated

From this heatmap we can clearly see that s1 and s3 have high corelation and the same goes for s2 and s4 i.e., they have a correlation of (>0.8). We see that multicollinearity can have a negative impact in the performance of our regression models. We can’t just get rid of features that are highly collinear with each other as they may hold some information. PCA Analysis is one of the ways where we can get rid of multicollinear features by assigning them to a vector space.

# Prediction of Time-To-Failure (ttf)

I used both Linear as well as Non-Linear regression models to predict the time-to-failure of the engines. The regression models that were tested and tried were Multiple Linear Regression, Polynomial Regression and Random Forest Regression. Each of the regressions are explained in short along with their implementation in Python.

## Multiple Linear Regression

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables (features) to predict the outcome of a response variable (ttf). The goal of multiple linear regression is to model the [linear relationship](https://www.investopedia.com/terms/l/linearrelationship.asp) between the independent variables and dependent variables. In essence, multiple regression is the extension of ordinary least-squares (OLS) [regression](https://www.investopedia.com/terms/r/regression.asp) because it involves more than one explanatory variable.

*Y*i​=*β*0​+*β*1​*xi*1​+*β*2​*xi*2​+...+*βp*​*xip*​+*ϵ*

**where, for***i*=*n***observations:**

*yi*=dependent variable

*xi*​=explanatory variables

*β*0​=y-intercept (constant term)

*βp*=slope coefficients for each explanatory variable

*ϵ*=the model’s error term (also known as the residuals)​

### Implementation of the Multiple Linear Regression

I implemented the Multiple Linear Regression on python using the sklearn library.

I first defined my X\_train and y\_train by assigning Cycle, s1, s2, s3, s4 to X\_train and ttf to y\_train.

data\_train is the variable name that stores the entire training csv table.

**X\_train = data\_train.drop(['id','ttf', 'label\_bnc'],axis=1).values**

**y\_train = data\_train['ttf'].values**

The same was done for the test dataset by preparing two variables X\_test which consisted of cycle, s1, s2, s3, s4 column values and y\_test consisted of the ttf column values.

data\_test is the variable name that stores the entire testing csv table.

**X\_test = data\_test.drop(['id','ttf', 'label\_bnc'],axis=1).values**

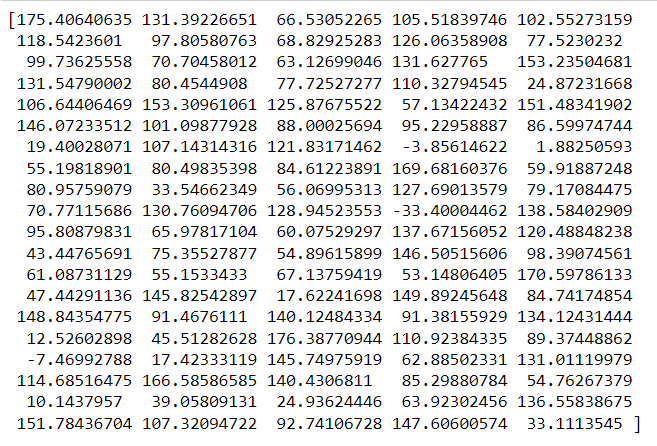
**y\_test = data\_test['ttf'].values**

Now as both are X\_train , y\_train and X\_test and y\_test variables are created we fit our Linear Regression model for the training dataset from the sklearn library.

**Graphical user interface, text, application

Description automatically generated**

These are the predicted values for the Linear Regression:



Here is a graph of Multiple Linear Regression that plots the actual vs predicted values

Chart

Description automatically generated

### Analysis of Multiple Linear Regression and why I chose it

* The reason I chose Multiple Linear Regression because it is the most basic technique to understand the relationship of two independent variables with a dependant variable and to know a value of the dependant variable at a certain independent variable.
* Upon completion of the analysis and plotting of the graph we can see a huge difference between the predicted and actual values this is because Linear Regression has two assumptions:

1. The regression model must be free of features having multicollinearity
2. The features should have a linear relationship with the dependant variable (ttf).

* In our Exploratory Data Analysis, we found that s1 and s3 , s2 and s4 are highly corelated with each other and also, we saw the sensors s1 and s3 show a nonlinear relationship w.r.t ttf.
* Hence, the model performed poorly.
* We will use performance metrics on our regression models to evaluate their performance and to confirm our analysis.

## Polynomial Regression

**Polynomial regression** is a technique we can use to fit a regression model when the relationship between the predictor variable(s) and the response variable is nonlinear.

A polynomial regression model takes the following form:

Y = β0 + β1X + β2X2 + … + βhXh + ε

**where, for***i*=*n***observations:**

*yi*=dependent variable

*xi*​=explanatory variables

*β*0​=y-intercept (constant term)

*βp*=slope coefficients for each explanatory variable

*h=*degree of power

*ϵ*=the model’s error term (also known as the residuals)​

### Implementation of the Polynomial Regression

We have already defined the X\_train, y\_train , X\_test, and the y\_test variables.

Graphical user interface, text, application, email

Description automatically generated

These are the predicted values for the Polynomial Regression

**Table

Description automatically generated**

Here is a graph of Polynomial Regression that plots the actual vs predicted values

**A picture containing graphical user interface

Description automatically generated**

### Analysis of Polynomial Regression and why I chose it

* The reason I chose Polynomial Regression is because here it is not required for the features to be linear as compared to Linear Regression. In our Exploratory Data Analysis, we had found that some of the variables displayed a nonlinear relationship with the target variable hence I chose this technique.
* Upon plotting the graph for actual vs predicted values we can see that the predicted values are almost as the actual values.
* This regression performs better because :-

1. By adding polynomial transform to the features showing a nonlinear relationship we improve the model’s performance.
2. It can also deal with features having high multicollinearity with each other.

* Hence, Polynomial Regression performs way better as compared to Linear Regression.
* We will use various **performance metrics** on this model to evaluate its performance.

## Random Forest Regression

Random forest is both a supervised learning algorithm andan ensemble algorithm.

Ensemble algorithms combine multiple other machine learning algorithms, to make more accurate predictions than any underlying algorithm could on its own. In the case of random forest, it ensembles multiple decision trees into its final decision.

The random forest algorithm follows a two-step process:

1. Builds *n* decision tree regressors (estimators).
2. Random forest regression takes the average of those predictions as its ‘final’ output.

Diagram

Description automatically generated

### Implementation of the Random Forest Regression

Graphical user interface, text, application, email

Description automatically generated

These are the predicted values for the Random Forest Regression

Text

Description automatically generated

Here is a graph of Random Forest Regression which plots the actual vs predicted values.

A picture containing chart

Description automatically generated

### Analysis of Random Forest Regression and why I chose it

I chose Random Forest regression because it was an ensemble level algorithm involving various decision trees and I expected the outcome to be the best as compared to other regression techniques. This is because it uses a number of decision trees and the average of the results of each decision trees results in the final output. But it was found that the Regression performed poorly as compared to Polynomial Regression.

I even tried Standardizing and Normalising the data as our independent variables lie in various data space. But after performing such techniques it was found that the model’s performance deteriorated. Upon further reading I realised that as this Regression technique is a Tree based model it doesn’t require feature scaling. Because tree-based models do not require the absolute value of the feature.

Hence, the performance of the regression was compromised.

I also feel like using this Regression on this problem leads to the problem of overfitting. If we are using a Random Forest with only one tree it will overfit to data because it is the same as a single decision tree. When we add trees to the Random Forest then the tendency to overfitting should decrease. However, the generalization error will not go to zero. The variance of generalization error will approach to zero with more trees added but the bias will not.

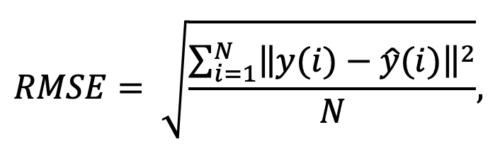
Hence, this regression performed poor as compared to Polynomial Regression.

# Evaluation of the Regression Models

We use various ways to evaluate the accuracy of the model and the regression metrics used to test the accuracy of the model were Root Mean Squared Error (RMSE), Mean Squared Error(MSE) and R square error.

## Root Mean Squared Error

Root mean squared error is a common technique that is used to evaluate the quality of prediction and how far it is from the measured true values based on the Euclidean distance. It is given by



where N is the number of data points, y(i) is the i-th measurement, and y ̂(i) is its corresponding prediction.

### Implementation of the RMSE in python

From the sklearn.metrics library import the mean\_squared\_error function.

While implementing RMSE make sure you set the squared=False otherwise it will calculate the Mean Squared Error.

Text, Word

Description automatically generated

## Mean Squared Error

MSE is calculated by taking the average of the square of the difference between the original and predicted values of the data.

A larger MSE indicates that the data points are dispersed widely around its central moment (mean), whereas a smaller MSE suggests the opposite. A smaller MSE is preferred because it indicates that your data points are dispersed closely around its central moment[(mean)](https://www.simplilearn.com/tutorials/data-analytics-tutorial/measures-of-central-tendency).

It is given by:

A picture containing graphical user interface

Description automatically generated

where:

* Σ – a symbol that means “sum”
* n – sample size
* actual – the actual data value
* forecast – the predicted data value

### Implementation of the MSE in python

From the sklearn.metrics library import the mean\_squared\_error function.

Text

Description automatically generated

## R-Squared Error

It is also known as the **coefficient of determination**. This metric gives an indication of how good a model fits a given dataset. It indicates how close the [regression line](https://www.studytonight.com/post/classification-problem-introduction-to-logistic-regression) (i.e. the predicted values plotted) is to the actual data values. The **R squared value lies between 0 and 1** where 0 indicates that this model doesn't fit the given data and 1 indicates that the model fits perfectly to the dataset provided.

It is given by:

Diagram

Description automatically generated with low confidence

### Implementation of the R Squared Error in python

Text

Description automatically generated

poly stands for Polynomial Regression the same was done for other models as well

## Comparing RMSE,MSE and R-Squared error of the Regression Models

|  |  |  |  |
| --- | --- | --- | --- |
|  | **RMSE** | **MSE** | **R Squared** |
| Multiple Linear Regression | 32.361 | 1047.228 | 0.394 |
| Polynomial Regression | 13.869 | 192.355 | 0.888 |
| Random Forest Regression | 29.005 | 841.307 | 0.512 |

It is found that Polynomial Regression performs better than Multiple Linear Regression and Random Forest Regression scoring a **RMSE of 13.869** which means that it predicts the ttf with an error range of **+-13.869.**

RMSE is an important feature in analysing the model as it explains change or error in prediction, and it is also able to give information on unseen data or new test values whereas R squared doesn’t give any information about unseen data. It explains the variability present in the data and Polynomial Regression was better to explain the variability in data by explaining around 88% of it.

Hence, Polynomial Regression is the best method to predict the ttf values for the engine.

# Classification Methods

To classify which engine would fail within a given time-period is defined by Classification methods. Classification methods used in this assignment are Naïve Bayes, KNN Classification and Logistic Regression.

## Naïve Bayes

Naive Bayes applies the Bayes' theorem to calculate the probability of a data point belonging to a particular class. Given the probability of certain related values, the formula to calculate the probability of an event *B*, given event *A* to occur is calculated as follows.

P(B|A) = (P(A|B) \* P(B) / P(A))

This theory is considered naive, because it assumes that there is no dependency between any of the input features.

### Implementation of Naïve Bayes in Python

From the sklearn.naive\_bayes library import Multinomial NB and fit the training model and predict the target variable that is **label\_bnc** based on the features of training set.

Here,

alpha = smoothening parameter

class\_prior = Prior probabilities of the classes. If specified, the priors are not adjusted according to the data.

fit\_prior= Whether to learn class prior probabilities or not. If false, a uniform prior will be used

Text

Description automatically generated

### Analysis of Naïve Bayes Classification and why I chose it

The reason I chose Naïve Bayes Regression is because:

* It can handle both continuous and discrete data.
* It is easily scalable with the number of predictors and data points.
* It is fast and it is easily scalable.

Upon Analysis it was found that it performed poor as compared to other Classification models and it was understood that:

* This classification assumes that all are the features are independent of each other.
* Its predictions were not always true.

We use various performance metrics to evaluate its performance and confirm our theory.

## KNN Classification

The general idea behind K-nearest neighbours (KNN) is that data points are considered to belong to the class with which it shares the most number of common points in terms of its distance. K number of nearest points around the data point to be predicted are taken into consideration. These K points currently belong to a class.

The data point under consideration is said to belong to the class with which the most number of points from these K points belong. The most popular formula to calculate the distance between the points is the Euclidean distance.

### Implementation of KNN Classification in Python

From sklearn.neighbors import KNeighborsClassifier function and fit your training model i.e., X\_train and y\_train. Then predict the classification label on the basis of the features of the testing dataset.

**Text

Description automatically generated**

### Analysis of KNN Classification and why I chose it

The reason I chose KNN Classification is because:

* It is one of the most simplest algorithm to implement
* It has high accuracy as there are no predictions made about the model what it is trained on it will give an output depending on that.

But one downside of KNN Classification is that is the data it is trained on turns out to have errors its errors will also be reflected in its predictions.

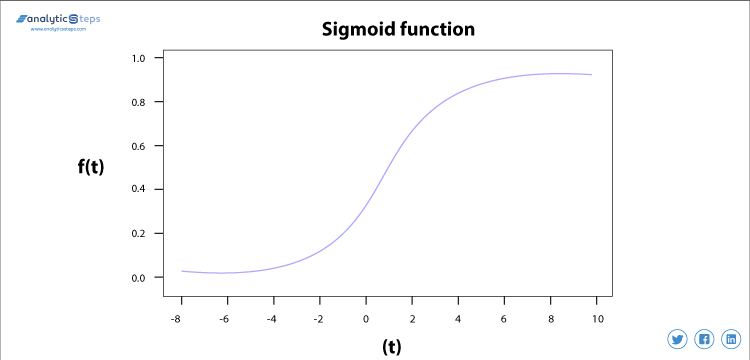
Other than that, it is one of the most widely used Classification technique. Hence I incorporated it in my analysis.

## Logistic Regression

In logistic regression algorithm, instead of predicting the actual continuous value, we predict the probability of an outcome. To achieve this, a logistic function is applied to the outcome of the linear regression. The logistic function is also referred to as a *sigmoid function*. This outputs a value between 0 and 1. Then, we select a line that depends on the use case. Any data point with a probability value above the line is classified into the class represented by 1. The data point below the line is classified into the class represented by 0.

Sigmoid Function – It is a type of function that converts the real values and maps it between 0 and 1. It is given by

***Y = 1 / 1+e -z***



So, if the value of z goes to positive infinity then the predicted value of y will become 1 and if it goes to negative infinity then the predicted value of y will become 0. And if the outcome of the sigmoid function is more than 0.5 then we classify that label as class 1 or positive class and if it is less than 0.5 then we can classify it to negative class or label as class 0

### Implementation of Logistic Regression in Python

From the sklearn.linear\_model import LogisticRegression and fit the training dataset then predict the target variable **label\_bnc** from the training data set depending on the features of the training data set

Text

Description automatically generated with medium confidence

### Analysis of Logistic Regression and why I chose it

The reason I chose this technique it’s because.

* It is a regression technique if the target variable has either true or false value and depending on the problem statement it was a perfect example to incorporate this technique.
* It serves as a base of all other Classification techniques such as SVM etc and I expected its results to be better given how easy it is to train and set up this technique with no extra tuning required.

Hence, I chose this method. We will further perform Performance metrics on our Classification model to evaluate it’s performance and compare it with our other Classification models.

# Accuracy of the Classification Method

Let us discuss the performance of our Classification Methods and why it was chosen.

To check the accuracy of the classification, we use sklearn library and import the metrics function. We then check the accuracy\_score against the actual and predicted.



Lrc stands for Logistic Regression the same was done for other models as well.

To analyse the result of our Classification technique using accuracy as the only Performance metric we will have to use ROC, Confusion Matrix and Recall as additional metrics to evaluate the performance of our Classification Techniques.

First let us see the Confusion Matrix of our Classification techniques

Chart

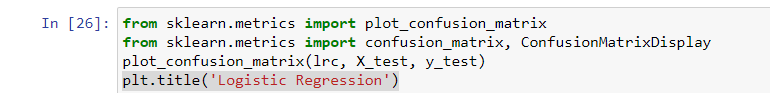
Description automatically generated Chart

Description automatically generated

Chart, treemap chart

Description automatically generated

In classification technique the goal is to reduce the number of False Negative Predictions i.e. The prediction says that the engine is okay but in reality the engine is a failure and this leads to a major issue. And we can see that our Logistic Regression predicts the least number of False Negatives.



The next method is using Recall Score and it is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

The best value is 1 and the worst value is 0.

Text

Description automatically generated

ROC and AUC plots the performance of Classification Model at all Classification thresholds. It plots the True Positive Rate and False Positive Rate.

AUC provides an aggregate measure of performance across all possible classification thresholds.

Higher the value of AOC better is the model’s prediction.

Text

Description automatically generated

Chart, line chart

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Accuracy** | **Recall Score** | **ROC and AUC** |
| Naïve Bayes Classification | 0.75 | 0.0 | 0.500 |
| KNN Classification | 0.87 | 0.56 | 0.767 |
| Logistic Regression | 0.89 | 0.68 | 0.820 |

It is found that Logistic Regression performs the best with the accuracy score of 89%, a recall score of 0.68 and an AUC value of 0.820

Basically, it tells us that TTF were divided into periods with different cycles for example

**period 0 : 0-15 cycles**

**period 1 : 15-30 cycles**

**period 2 : 30+ cycles**

and the classification methods were able to predict the period in which an engine can fail.

# Conclusion

Thus, by analysing the data given to us I performed various predictions using training dataset and testing dataset and was able to conclude with the best performing regression method – Polynomial Regression and Classification Method – Logistic regression.

By employing various other Regression methods and Classification methods we could get a more refined and accurate model. To enhance the performance metrics of the regression models we can try to normalize the data or tune the model parameters.

With this assignment we addressed two main problem statements

1. When will an engine fail?
2. Which engine will fail during which time-period?

Predictive Maintenance is a field that requires refinement on a daily basis but with this assignment I could understand the process that is to be followed to perform Regression and Classification techniques.

# References

* *Multiple linear regression (MLR) definition*. (2008, March 16). Investopedia. <https://www.investopedia.com/terms/m/mlr.asp>
* Zach. (2021, September 13). *When should you use polynomial regression?* Statology. <https://www.statology.org/when-to-use-polynomial-regression/>
* *The ultimate guide to random forest regression*. (2022, November 15). Keboola - Connect any data source in less than 20

minutes. <https://www.keboola.com/blog/random-forest-regression>

* Gupta, A. (2021, December 15). *Mean squared error : Overview, examples, concepts and more*. Simplilearn.com. <https://www.simplilearn.com/tutorials/statistics-tutorial/mean-squared-error>
* *Root mean Square error (RMSE)*. (2021, September 28). C3 AI. <https://c3.ai/glossary/data-science/root-mean-square-error-rmse>
* *What is mean squared error, mean absolute error, root mean squared error and R squared?* (2019, July 13). Studytonight - Best place to Learn Coding Online. <https://www.studytonight.com/post/what-is-mean-squared-error-mean-absolute-error-root-mean-squared-error-and-r-squared>
* *IBM developer*. (n.d.). IBM Developer. <https://developer.ibm.com/tutorials/learn-classification-algorithms-using-python-and-scikit-learn/>
* Kumawat, D. (n.d.). *Introduction to logistic regression - Sigmoid function, code explanation*. Analytics Steps - A leading source of Technical & Financial content. <https://www.analyticssteps.com/blogs/introduction-logistic-regression-sigmoid-function-code-explanation>
* *Sklearn.metrics.recall\_score() - scikit-learn - W3cubDocs*. (n.d.). W3cubDocs API Documentation. <https://docs.w3cub.com/scikit_learn/modules/generated/sklearn.metrics.recall_score>
* Spiegel, M. R., John J. Schiller, J., & Srinivasan, R. A. (2013). *Schaum's outline of probability and statistics: 897 solved problems + 20 videos* (3rd ed.). McGraw Hill Professional.
* Murphy, K. P. (2012). *Machine learning: A probabilistic perspective*. MIT Press.
* *Classification: ROC curve and AUC*. (n.d.). Google Developers. <https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc>
* Towards AI Team. (2020, November 17). *Why choose random forest and not decision trees*. Towards AI. <https://towardsai.net/p/machine-learning/why-choose-random-forest-and-not-decision-trees>
* *Samimust/predictive-maintenance*. (n.d.). GitHub. <https://github.com/Samimust/predictive-maintenance>
* (n.d.). Simplilearn | Online Courses - Bootcamp & Certification Platform. <https://www.simplilearn.com/tutorials/machine-learning-tutorial/naive-bayes-classifier>
* Chatterjee, M. (2022, August 29). *A quick introduction to KNN algorithm*. Great Learning Blog: Free Resources what Matters to shape your Career!. <https://www.mygreatlearning.com/blog/knn-algorithm-introduction/>
* *Medium*. (n.d.). Medium. <https://levelup.gitconnected.com/random-forest-regression>
* Gupta, A. (2021, December 15). *Mean squared error : Overview, examples, concepts and more*. Simplilearn.com. <https://www.simplilearn.com/tutorials/statistics-tutorial/mean-squared-error>