National College of Ireland

Computing Project

Data Science

2022/2023

Jordan O'Donovan

x19372016

x19372016@student.ncirl.ie

Mimicking the Actions of a Support Employee Using Machine Learning

Data Science Report

Contents

Executive Summary

The purpose of this report is to utilise supervised classification models to predict the next step needed to be taken for a telecommunications test which occurred at Spearline Labs. This report details each step taken from cleaning the data, analysing and pre-processing it, to model training, testing and hyperparameter tuning. The impact of each feature and hyperparameter on the models was detailed and visualised.

This report found that once feature selection and hyperparameter tuning had occurred, there was less overfitting during the training of the models, thus their performance when using the testing dataset increased. The metrics measured in this paper are accuracy, precision, recall and the f-score.

The conclusions from this report are that these supervised classification algorithms are very effective in creating accurate predictions, and that the feature selection and hyperparameter tuning techniques utilised not only drastically reduce the computational power required to create the models, but they also increase the performance of said models.

1.0 Introduction

1.1. Background

As an exciting and innovating company, I chose to undertake this project so that I could assist Spearline Labs in the speeding up of their day-to-day operations, though the use of machine learning algorithms and techniques. The idea for this project came about through collaboration with the Data Science team lead at Spearline.

At the moment, Spearline is using a combination of logic based and deep learning approaches to facilitate the auto-marking of tests, although it is only for specific tests and specific scenarios. The goal of my project is to make accurate predictions for every test type and every target variable type. This would simplify Spearline's current approaches as they have to run multiple Docker containers and Python scripts to perform this automation. My project will reduce this to a single container and script.

1.2. Aims

This project aims to automate the process of predicting why one of Spearline's phone line tests failed, inputting into their database both the reason why, and if a further step for said test is required, predicting what that step is to be. Now, one of Spearline's Support Team employees must listen to the recording of the failed test and interpret the data to understand why the test failed, a process which can take up to a few minutes to complete. The goal of my project is to be able to perform these actions in a fraction of a second, at a higher accuracy rate than the employees can perform these actions at.

1.3. Technology

To create this project, I received datasets at the beginning of each month from Spearline, containing every failed test during the previous month, their data and the actions (target variable) taken.

I will use Python, namely the Pandas module to read and clean said data, by checking for outliers, missing data, inconsistencies. NumPy, Matplotlib and Seaborn to analyse and visualise the data. Finally, I will use Scikit-learn for my supervised classification task, as it encompasses all of the functions needed for building, testing and training the models, performing feature importance and selection, cross validation and hyperparameter tuning.

1.4. Structure

This project report is broken down into ten sections. The first (this one) is to serve as the introduction to the paper, explaining what the objectives are and how they're to be undertaken.

Section 2 is State of the Art. In this section I was create a detailed literature review, in which I examine existing, similar papers in order to gain a better understanding of how this project should be undertaken, and where there's gaps in the current research related to this.

Section 3 is Data. Here I will perform analysis on the dataset, explaining and visualising the characteristics of the data, what cleaning had to be done and any pre-processing which occurred during this step.

Section 4 is the Methodology section. In this one I will detail the steps taken during the building of my machine learning models, and I will explain why each step in this process must take place.

Section 5 is Analysis. In this section I will using graphs and statistics to explain how my models perform, and how that performance changes as the steps detailed in Methodology occur.

Section 6 is Results. Here I will discuss how my final models perform, detailing each metric being measured for this project.

Section 7 is Conclusions, where I will discuss the key takeaways from both my models and the project, whether it was a success or a failure, where the strengths and weaknesses are, and what the advantages and disadvantages of this project were.

Section 8 is Further Development or Research, where I will discuss what further work can be done to the findings from this project. I will detail what steps I would take next if I had more time and resources to carry out this project.

Section 9 is References, where I will detail the sources cited throughout this project.

The final section is Appendices, where the initial project proposal, the ethics approval application and reflective journals will be stored.

2.0 State of the Art

In November 2019, Mohammad Kazim Hooshmand and Hosahalli Doreswamy created a machine learning approach for detecting network anomalies [1]. They utilised several machine learning algorithms such as random forest, decision tree, support vector machines and gradient boosting. They trained and tested them then gauged their performance using their accuracy, precision, recall and f-score. What they concluded was that these machine learning based approaches can be extremely effective at detecting anomalies on networks. Random forest performed the best with 98% accuracy, while decision tree followed with 97%.

This paper has taught me much about how to go about completing the task I'm basing my project on. It has shown me how effective random forests are, as well as decision trees, and has taught me the importance of not just using accuracy as your measurement for success when it comes to making predictions.

3.0 Data

At the beginning of each month, I receive a text file containing the console log outputs for the marking of each test during the previous month ran through Spearline’s system, in a JSON format. This dataset contains 110 variables which contain datetime values, continuous and discrete numerical data, as well as ordinal and nominal categorical data. Many of these columns contain the same data, but in both a numerical and categorical format e.g., “company_id” containing the Universally Unique Identifier (UUID) value for a company and “company_name” containing the name of said company as a string.

Opening these text files as a DataFrame proved to be the most tedious aspect of this project. Each line of the file had to be iterated through, using regular expression (regex) to remove the beginning and ending of each line, as well as temporarily altering specific characters, like colons in datetime values and commas which were found inside of string values. This was necessary as after the files had been prepared in this way, the lines had to be looped through again, using Python’s dictionary feature to split each line into key-value pairs, as this limited the appearance of each key to a single instance.

Once these steps had been completed, the data was able to be opened as a Pandas DataFrame. In total, there are 122,668 rows and 110 columns.

At the beginning of my analysis of this data, I investigated the correlation values between five columns I suspected would have a strong correlation between each other after having a cursory glance at the DataFrame.

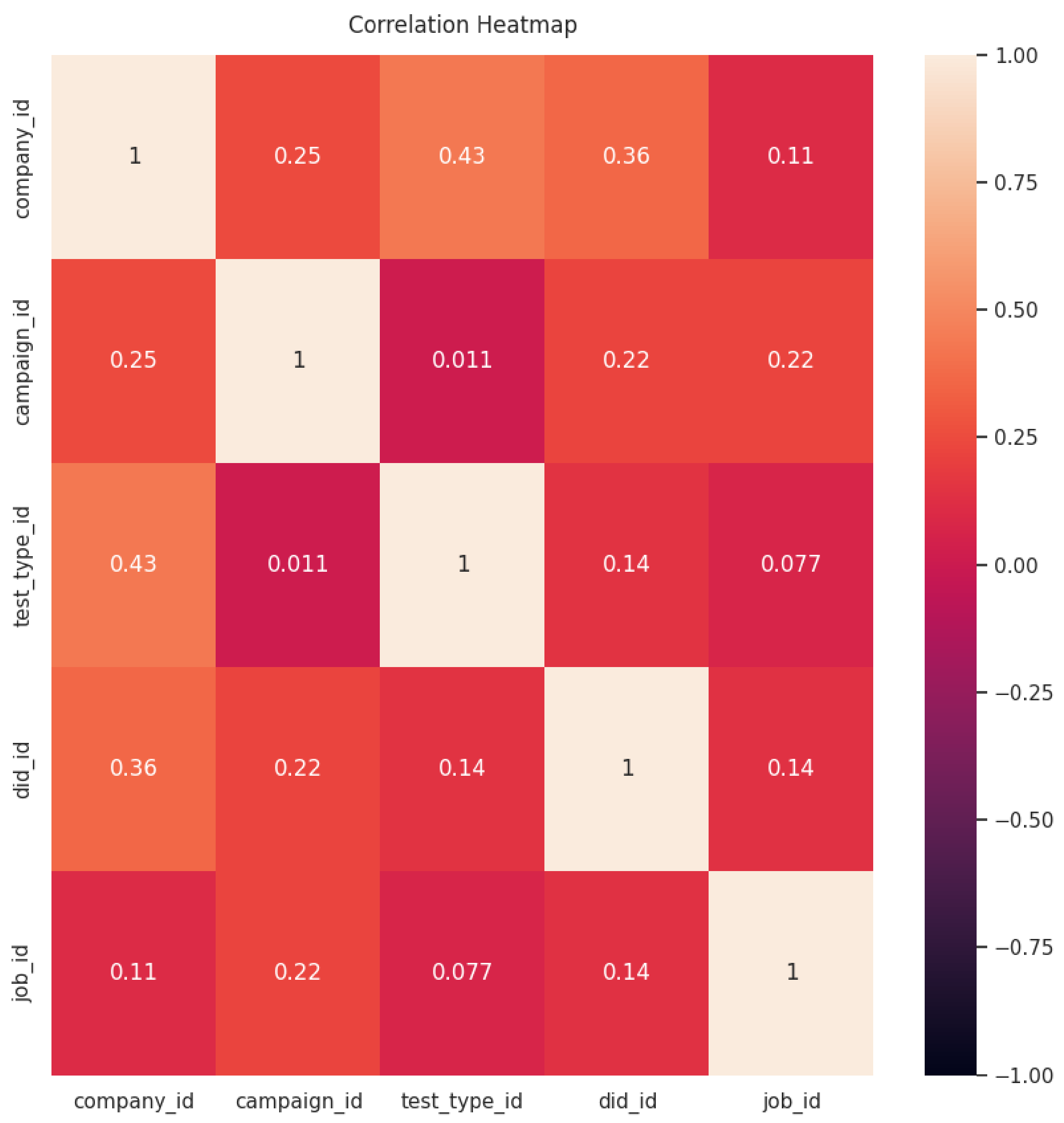


Figure 1. Heatmap showing correlation between five columns

The correlation between any of these variables was not as high as I was initially expecting, with the highest correlation being 0.43 between the company and the test type used.

When the correlation for every numeric column, it shows that there are much stronger correlations between some columns not seen in the previous heatmap. These four columns have very strong/perfect correlations as for 125 tests they all have call_length 2100, silence as 3, and the other two columns have values which range from one to four, but they also contain null values in some of the tests which the first two columns have data. All 125 of these tests were test type 11 – Conference Long Call, with number type 1 - Toll and company_id 400.

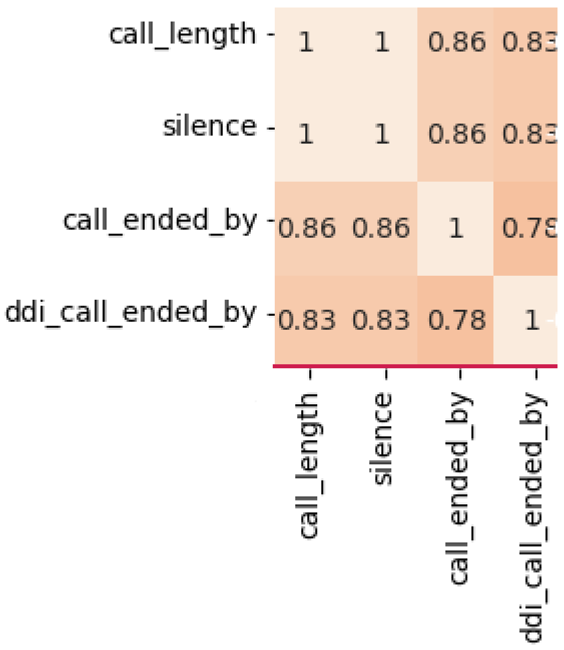


Figure 2. Heatmap showing strong correlation between four columns

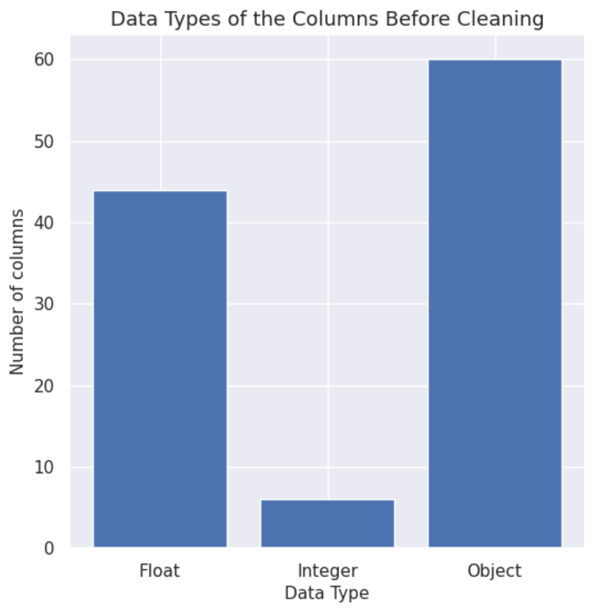


Figure 3. Bar chart of data types of columns in the DataFrame before cleaning

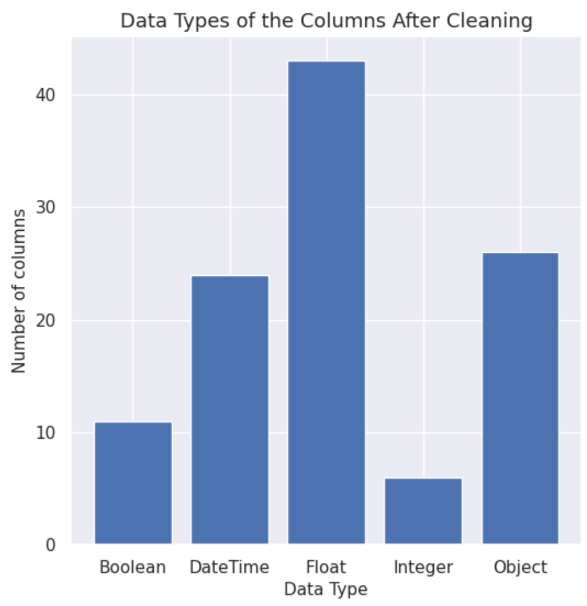


Figure 4. Bar chart of data types of columns in the DataFrame after cleaning

Before cleaning, Pandas automatically assigned 60 of the columns to be of datatype Object, 44 of them to be Float, with the remaining 6 being set to Integers. As this dataset contains many DateTime and Boolean values, many of the columns had to be manually assigned another datatype. After making these changes, the dataset now contained 43 Float columns, 26 Object, 24 DateTime, 11 Boolean and 6 Integer.

In this dataset, only 25 of the 110 columns contain data in each instance, the other 85 contain at least one NULL value. The column with the least amount of NULL values, "created_on", is missing a datetime value only once, while 9 of the columns lack any data at all.

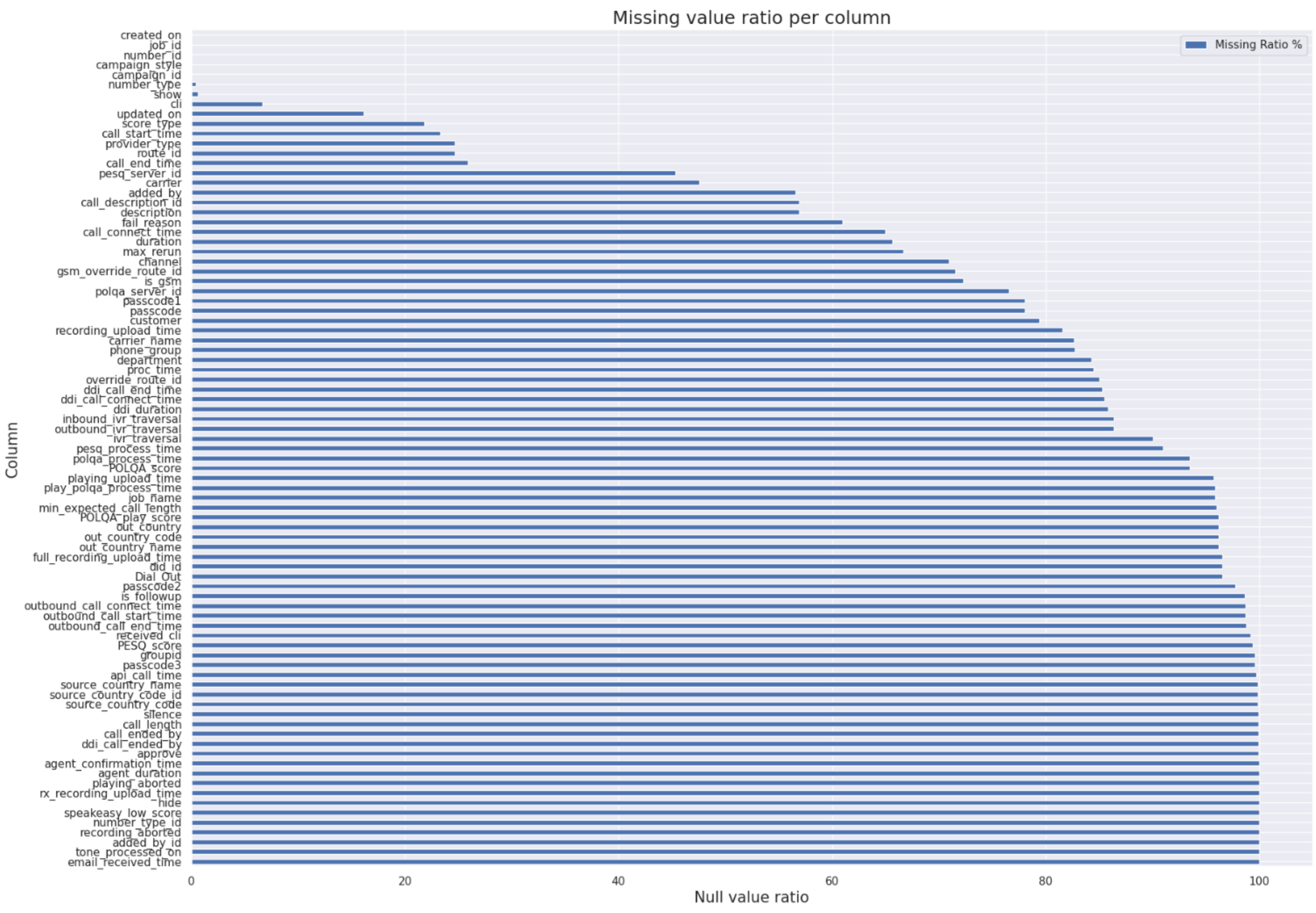


Figure 5. Horizontal bar chart showing percentage of missing values per column.

instances, this appears to be due to the type of test run through Spearline's system, along with the parameters chosen for said test.

The target variable for this project is to predict the "action" column. This is the step taken by a member of Spearline's Support Team, once they've analysed the test.

For this column, there are 79 unique values my model will have to choose from when making its prediction, but these are quite skewed. 40.48% of the tests contain "check call history" for this variable, with the second most common being "rerun alternative route" with 12.33%. The only two other values which make up a double-digit share of the total are "marked temporarily unable to test" with 11.28% and "marked busy" with 10.4%.

I replaced all missing data with "0" in the other numerical columns. I at first considered using imputation but as shown above, many of these columns are missing data for many of their rows thus I was concerned about imputation leading to overfitting the model. In almost all these columns, the minimal value they have is 1, thus replacing the missing values with 0 should not negatively affect the performance of the models, although a close eye must be kept to ensure this.

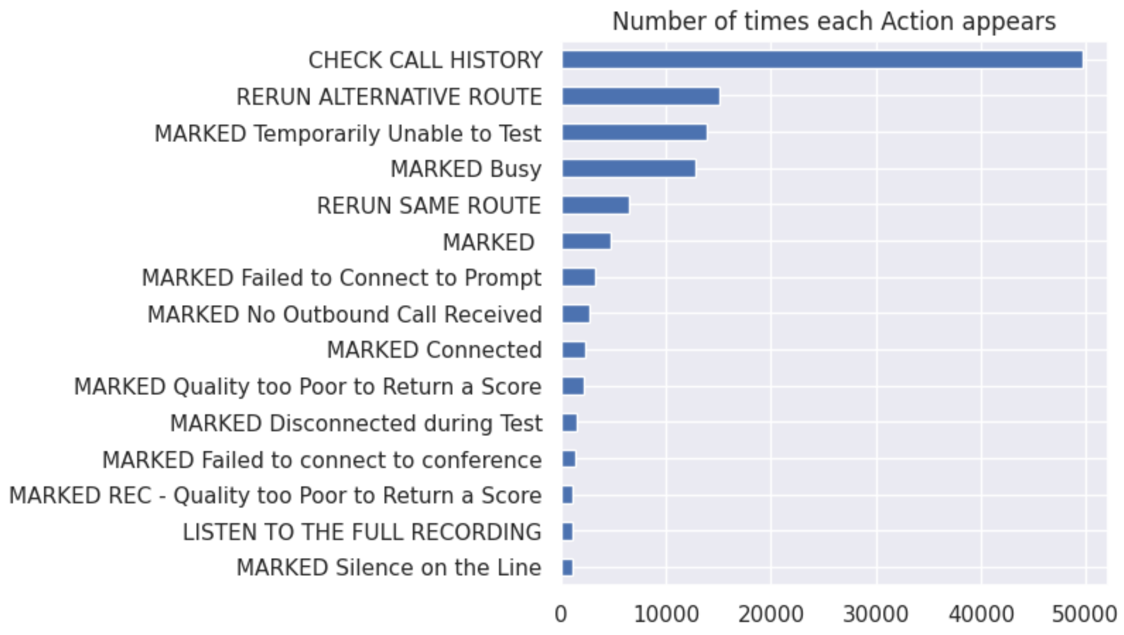


Figure 6. Horizontal bar chart showing number of times each target variable value appears.

I suspected that the tests with no call_connect_time value would present a much different story for their target variable, but surprisingly, the share of each value was very similar. Here, “check call history” made up 40.45% instead of 40.48% and “rerun alternative route” made up 18.36% instead of 12.33%. The two “marked” values were also third and fourth most frequently seen here, except this time their order was reversed, and they each made up a slightly higher percentage of the total.

When we look at the test types seen most frequently, “Conference” takes the top spot with 18.7%. There are 42 different test types present in this dataset. The correlation between the target variable Action and the test type column is only 0.008, thus predicting this variable may prove to be much more difficult than initially anticipated.

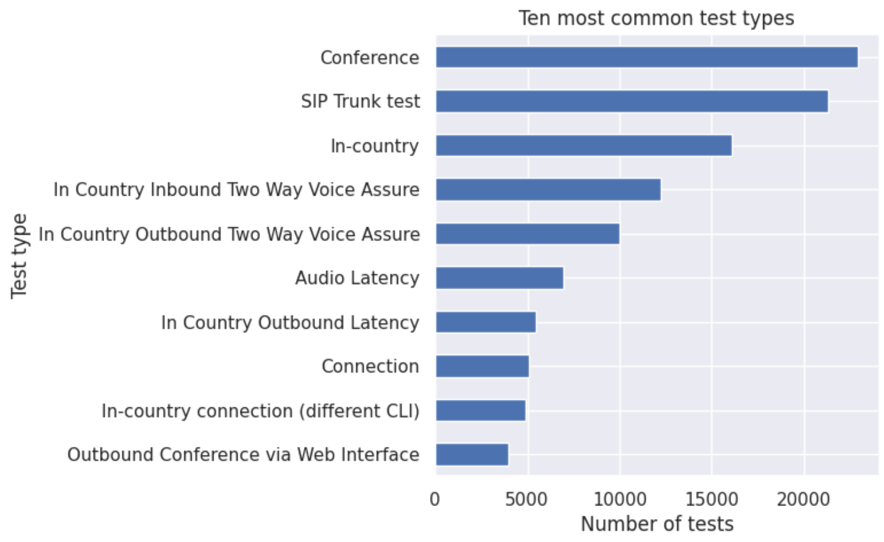


Figure 7. Horizontal bar chart showing the ten most frequent test types.

One company, ID 396 was the company responsible for 28.87% of the tests in this dataset. They ran almost twice as many tests as the next most frequently seen company ID, 100, who ran 15.51% of the tests. They also ran almost twice as many tests, with the third most common company ID, 147, making up only 8.21%.

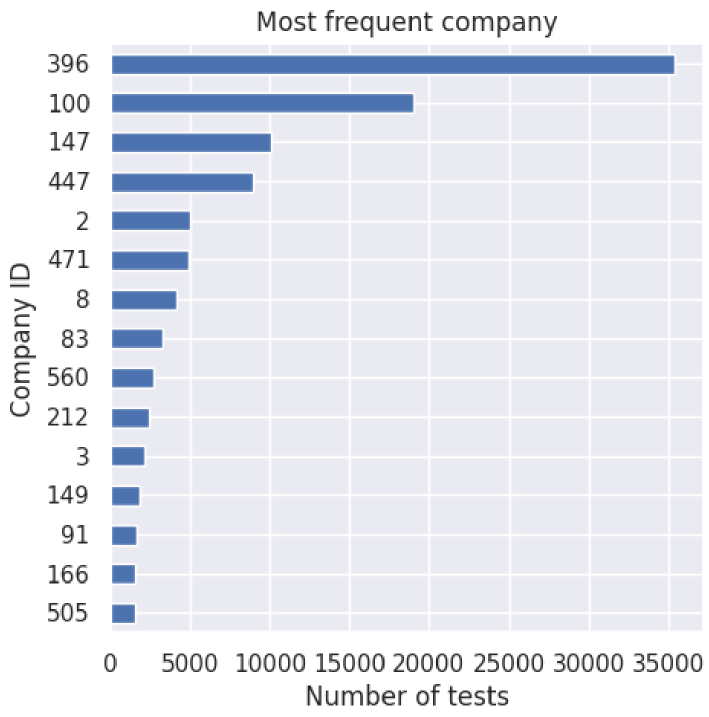


Figure 8. Horizontal bar chart showing the most frequent company IDs.

The share of countries these tests were ran in is much more distributed than the companies. The most popular country to run tests in, The United States, is only used for 7.04% of the total, with Germany making up 6.98% and Colombia with 5.87%.

I theorised that there would be a moderate correlation between the company and country columns, but their correlation is only 0.033. As shown in the first heatmap, the correlation between test types and company is a moderate 0.43 which begs the question what the correlation between the test type and the country is, but as shown below, it's a negligible -0.053.

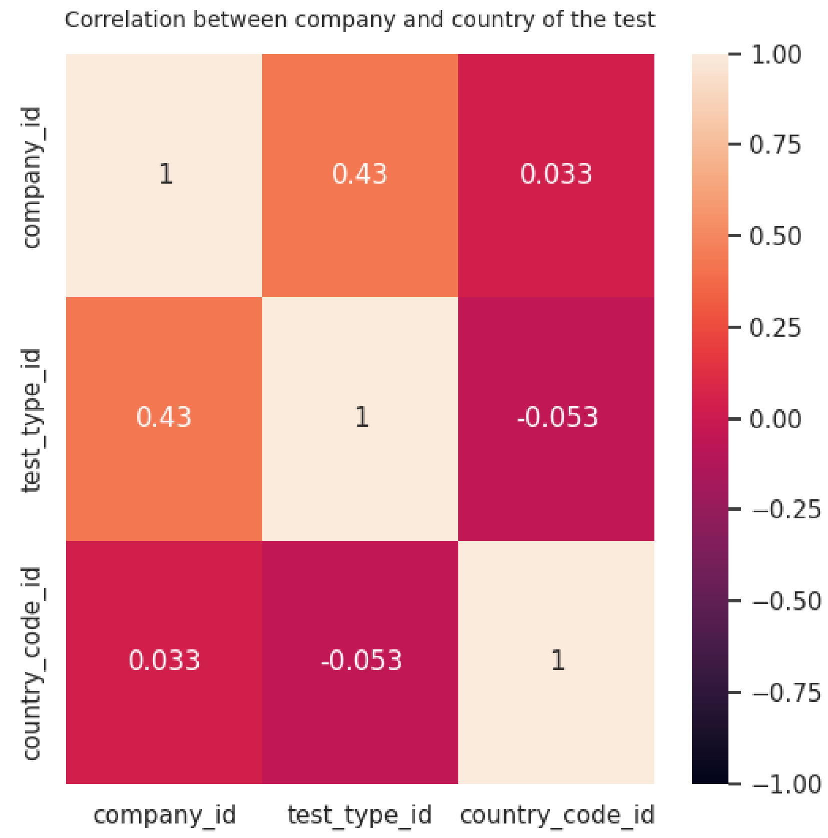


Figure 9. Heatmap showing correlation between three columns.

The call_description column of the dataset contains the reason why a test failed; thus this will likely be a very important column when predicting the target variable. However, in this column, 56.94% of the rows are missing data. Of the rows which contain a value for this variable, 30.29% of them contain call description ID 3, with ID 9 making up 29.47% and 53 making up 6.78%.

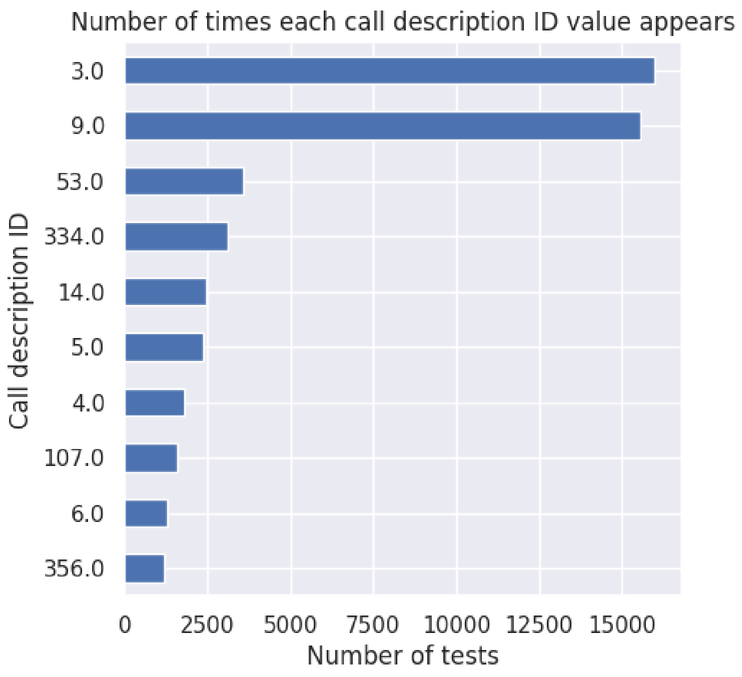


Figure 10. Horizontal bar chart showing how many times the most frequent call description ID values occur

I encoded the target variable Action as a new column named "Label" so that it could be utilised much more easily when calculating correlations and training and testing the models. The correlation between the target variable and this column is only 0.23, which is significantly lower than initially expected.

When we look at the tests with the most common call description value, 3, we can see that almost every test had one of two actions taken. 79.72% of them were marked busy, and 20.25% had "check call history". The remaining four actions were only taken once each.

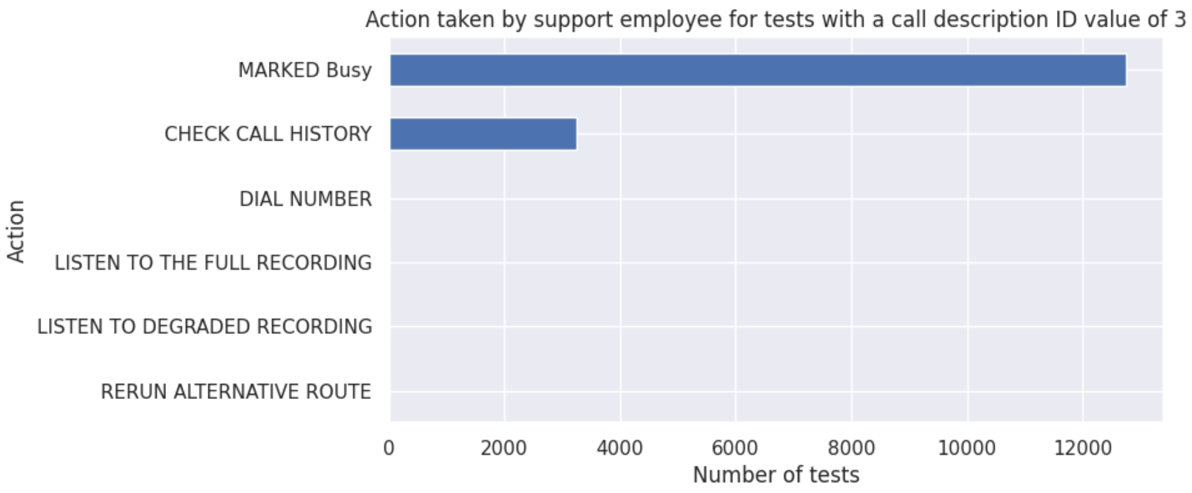


Figure 11. Horizontal bar chart showing the number of times each target variable occurs when call description ID is 3.

For the countries these tests occurred in, they're disproportionate to the total dataset. Here, Indonesia is the most seen country, making up 13.86% of these tests, with Australia making up 10.61% and Spain with 5.3%.

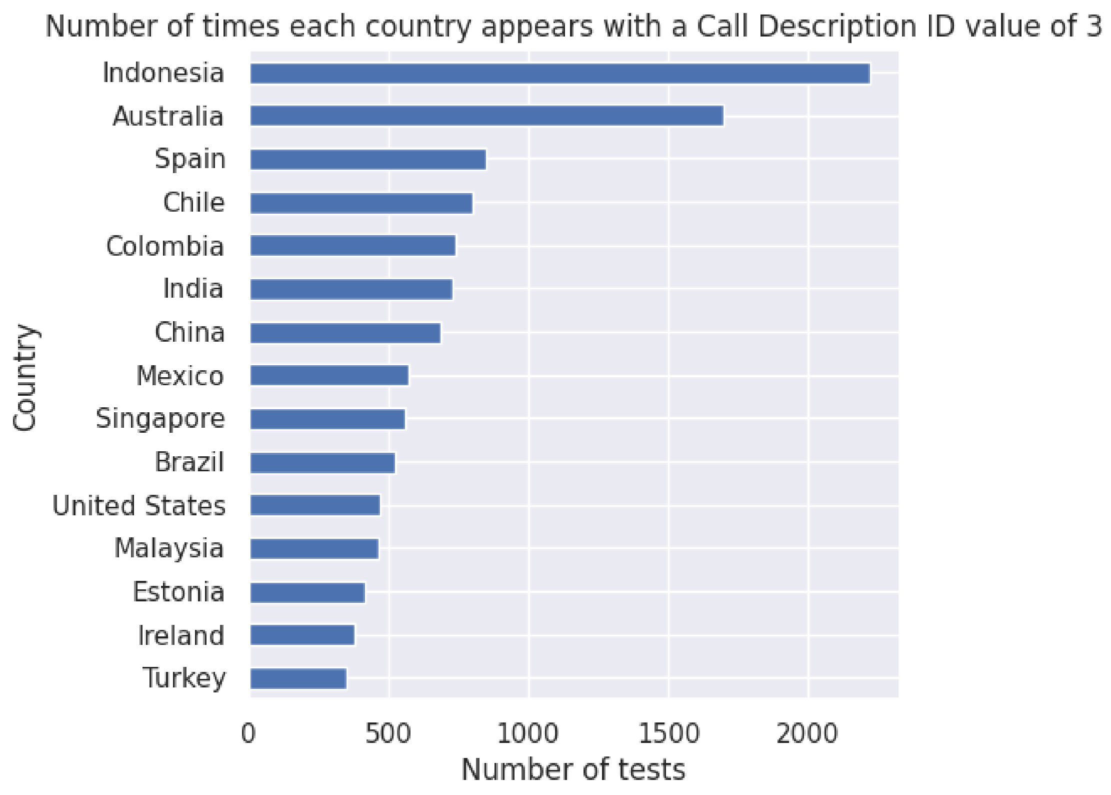


Figure 12. Horizontal bar chart showing how many times each country appears when call description ID is 3.

For the other call description value making up almost a third of the total, these variables are much more skewed. There are six unique action values taken here, with "marked temporarily unable to test" making up 88.92%, "check call history" making up 10.91%.

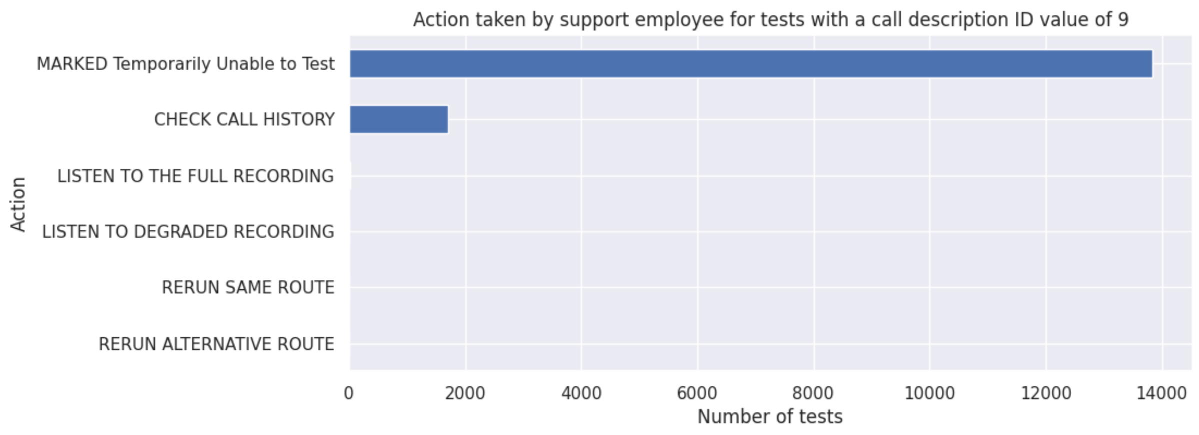


Figure 13. Horizontal bar chart showing the number of times each target variable occurs when call description ID is 9.

Germany is the most frequent country here, seen 30.43% of the time, with Brazil being the only other country making up a double-digit percentage of the total with its 12.14%.

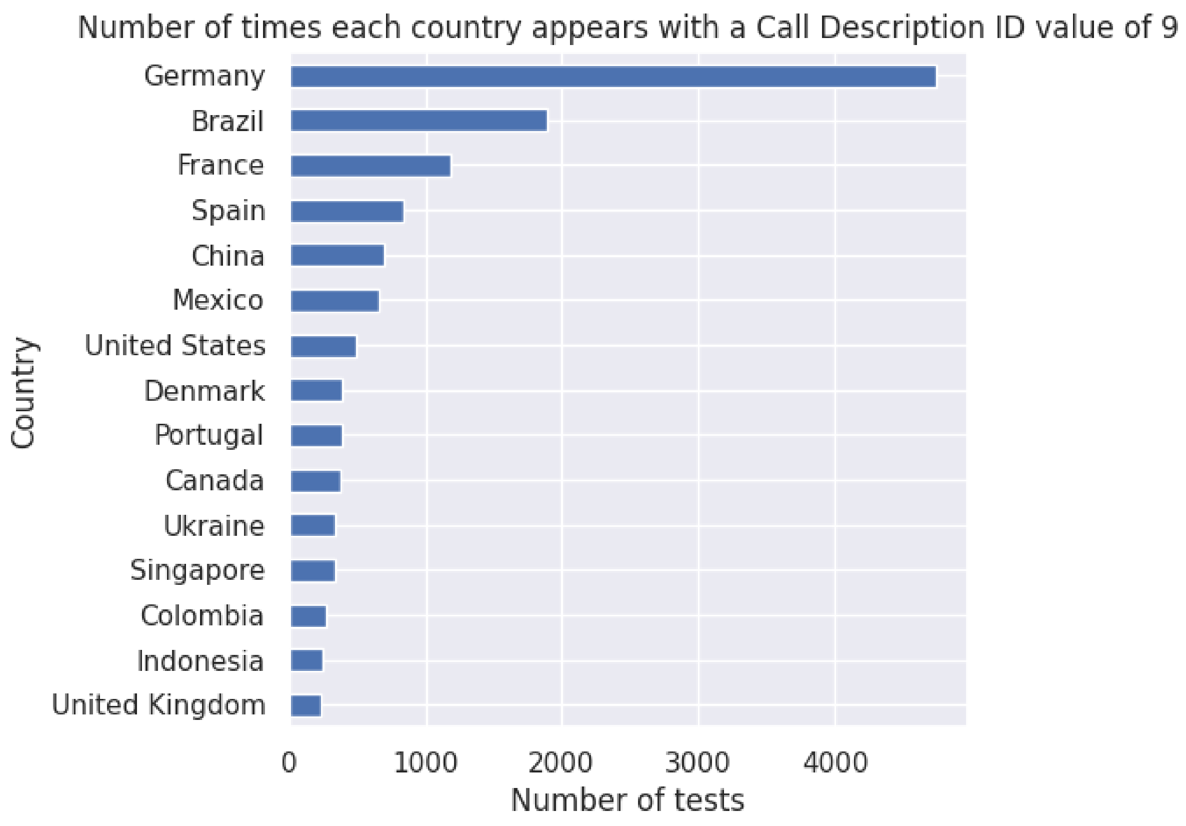


Figure 14. Horizontal bar chart showing how many times each country appears when call description ID is 9.

Something to note here is that the tests whose actions are "MARKED Busy" or "MARKED temporarily unable to test" make up 21.68% of the total. As their actions are one fifth and one tenth "CHECK CALL HISTORY", it could be very effective to either find out what circumstances are needed to get this action or create a machine learning model to work with these two call description IDs alone. This could ensure that over a fifth of the dataset would be predicted with an almost perfect accuracy.

4.0 Methodology

The target variable was encoded and was stored in a new column named “Labels”. This was done as the machine learning algorithms I was planning on using, such as random forests and gradient boosting require numerical values to make their predictions. Another column, fail_reason was also encoded as “fail_reason_ID”. This was done as this column is similar to the call description column i.e., it contains data pertaining to why the test failed thus may be instrumental for predicting the target variable.

The data was split into training and testing sets, with 80% of the data being in the training and the remaining 20% in the testing.

I also created a version of this split with standardisation applied to it by using Scikit-learn’s StandardScaler function. This reduces the mean of each feature to 0 with a standard deviation of 1. I will run this standardised version of the data through my model to investigate whether the model’s performance is improved once the variance of each feature is equal.

Once the data was pre-processed, it was now ready to be analysed. The data was run through several feature importance models, to calculate which features carried the most weight for the target variable. The data was run through Scikit-learn’s DecisionTreeClassifier, LogisticRegression, RandomForestClassifier, XGBClassifier and KneighborsClassifier, with the most important features chosen by each algorithm put into a table to compare their results.

Cross validation was used when training the models. This was essential to improve the model’s resistance to overfitting and to acquire a better understanding of how the model will perform when it’s fed new, previously unseen data. Scikit-learn’s RepeatedStratifiedKFold function was used to perform this cross validation, with ten folds and three repeats. These two parameter values were chosen as they represented an effective balance between the robustness of the cross validation and the time taken to perform the cross validation.

Scikit-learn’s RandomForestClassifier, DecisionTreeClassifier and GradientBoostingClassifier were the three algorithms used to train models. Their performance was first examined using every numerical column found in the dataset to use as a benchmark to compare my future models against. For these models their training and testing accuracy is calculated, along with their precision (the percentage of true positives out of all positive predictions), recall (the percentage of true positives out of all actual positives) and f-score (the harmonic mean of precision and recall).

Once these models had finished running, feature selection was the next step. Here, many different combinations of the features highlighted during the feature importance stage were run through the models. Any features which had no effect on the accuracy of the training and testing sets were eliminated, as all they were contributing was longer computation times.

The goal of the feature selection was to find the subsection of features which could be chosen to return better performance than was achieved by running every feature through the models.

After the optimal group of features had been found, the final step was to perform hyperparameter tuning. The objective of this set is the calculate which hyperparameters return the greatest performance (namely accuracy) from the models. Scikit-learn's GridSearchCV function was used to perform this tuning.

The performance of every hyperparameter working together was found using this GridSearchCV function, and afterwards the impact of the individual hyperparameters on the performance of the models was investigated and visualised.

For random forest, six hyperparameters were chosen to perform tuning with:

- **N_estimators** specifies the number of trees to be built in the forest. Increasing the number of trees may improve the performance of the model, but it will also increase the computation time needed for training and testing.
- **Max_depth** specifies how deep a tree in the forest can go. Increasing the depth of a tree may improve performance but it can also lead to overfitting.
- **Max_features** states the maximum number of features that can be considered during the building of the trees. This hyperparameter can be utilised to reduce overfitting.
- **Min_samples_split** states how many samples are needed when splitting nodes in the trees. This value is increased when attempting to reduce overfitting.
- **Min_samples_leaf** is like the previous hyperparameter, but instead of being the minimum number of samples needed to split a node, it's the minimum needed to be at a leaf node in a tree. This is also increased when attempting to reduce overfitting.
- **Bootstrap** is a binary hyperparameter. When set to true, bootstrap samples will be used in the trees in the hopes of reducing overfitting. Setting this to false makes the entire dataset be used to build each tree.

Seven hyperparameters were chosen to tune the decision trees with. They are:

- **Criterion** is the function which calculates the quality of a split separating the classes in the target variable in a decision tree. Lower values here indicate a better performing split. If the data is balanced, "gini" should perform better than "entropy".
- **Splitter** states the algorithm used to split the nodes in a decision tree. "Best" will select the best split while "random" chooses a random split.
- **Max_depth** is the maximum depth a decision tree can have. Increasing this may improve performance but may also introduce overfitting.
- **Min_samples_split** is the minimum number of samples needed to split a node. Having this too low can lead to overfitting while having it too high can lead to underfitting.

- **Min_samples_leaf** is the minimum number of samples needed to be at a leaf node. This is similar to the previous parameter, in the sense that having this be too low can lead to overfitting and too high can lead to underfitting.
- **Max_features** is the maximum number of features considered when splitting the node of a decision tree. A larger value may lead to overfitting while a value too small may lead to underfitting.
- **Class_weight** assigns the weight associated to each class in the target variable.

5.0 Analysis

The first feature importance algorithm used was logistic regression. The four features given the highest importance here (highest positive values) are the test_type, added_by, is_gsm and fail reason. The four features with the most significant negative scores are provider_type, POLQA_score, application_id and test_counter.

The positive coefficient values from this algorithm mean that these features are regarded as being important when predicting the target variable. The negative coefficients mean that these features may diminish the performance of models if they're included.

Is this an alright template to use for the remaining algorithms?

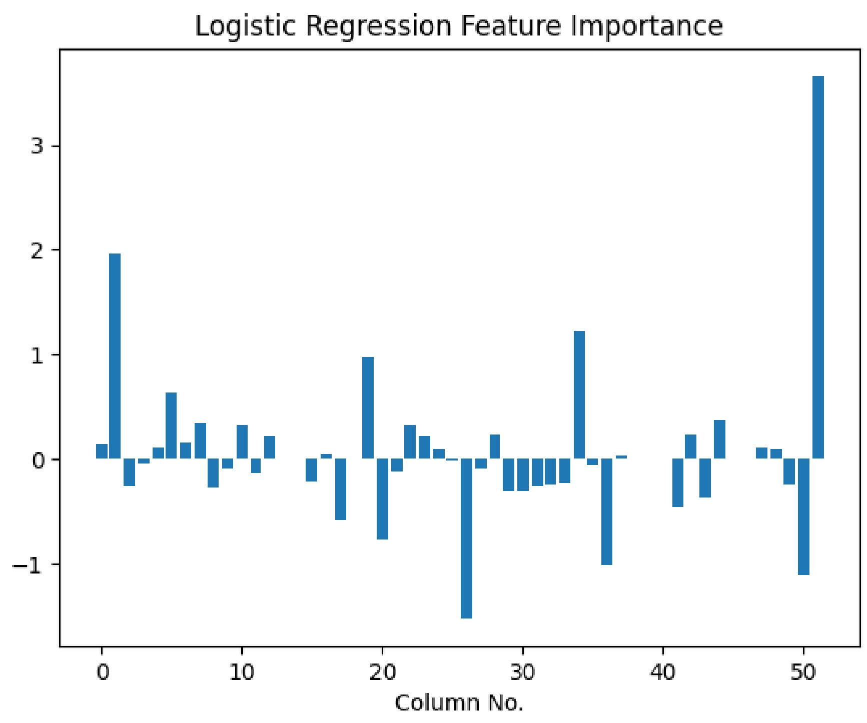


Figure 15. Feature importance using Logistic Regression.

When DecisionTreeClassifier was used, the feature is calculated as being the most important was by far call_description. It assigned this column a score of 0.47942, whereas the column with the second highest score, ID, only had 0.07370. The other two features given a score of above 0.07 were test_counter and failure_reason.

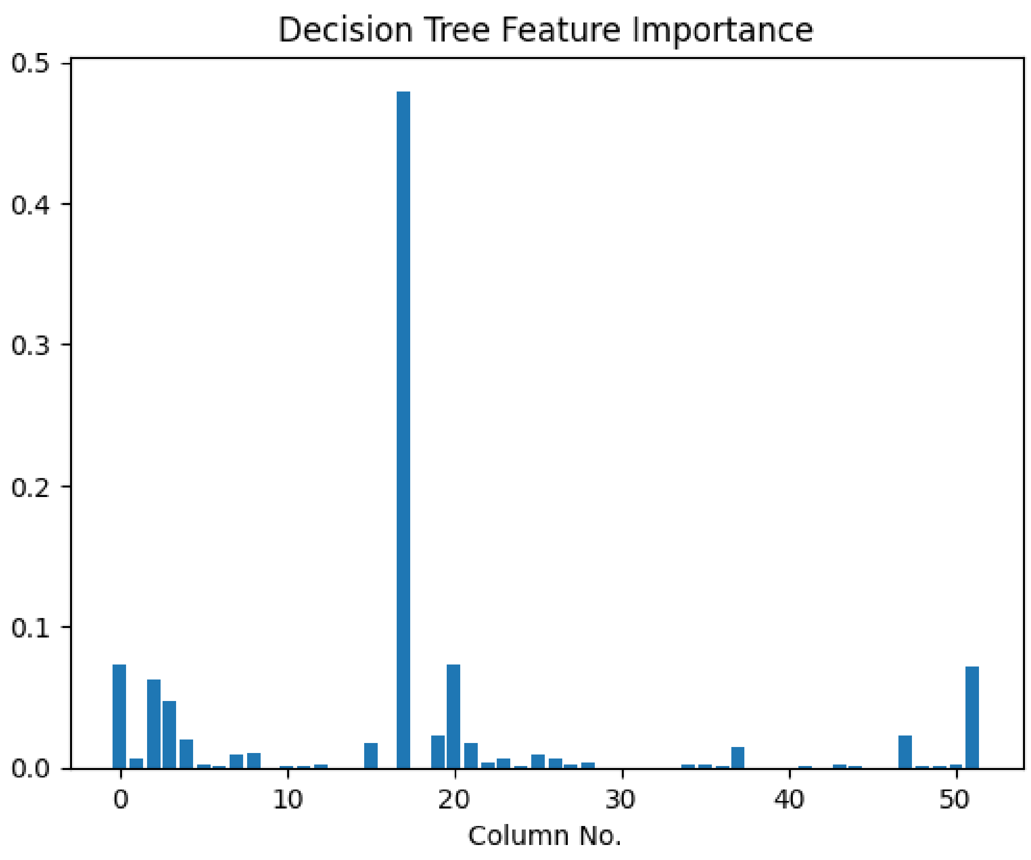


Figure 16. Feature importance using Decision Trees.

Random forest classification was the third algorithm used for feature importance. Again, call_description is by far the feature with the highest score here, although its not quite as ahead of the others as it was in the previous algorithm. The other three features with the highest scores are also highlighted by the previous two algorithms. Added_by was one of the top features chosen by decision tree and was the second most important here, while ID, job, call_description and failure_reason were also in this batch of highly rated features.

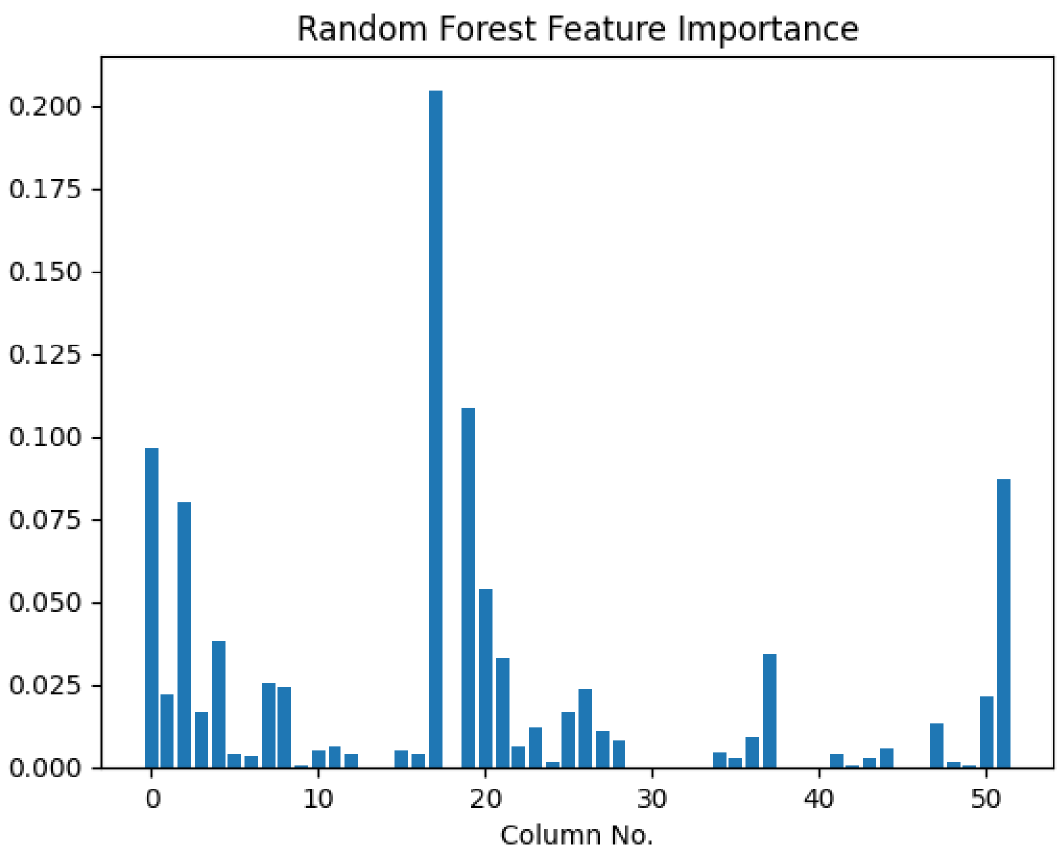


Figure 17. Feature importance using Random Forest.

XGBoost classifier had the most unique results of the algorithms seen so far. This model gave the significantly highest score to score_type, with the second most important feature being pesq_server_id. After these, it ranked some previously seen features as being moderately important, namely application, failure reason, ID, test_type and added_by. The unseen features which were also given moderate importance were route and campaign_style.

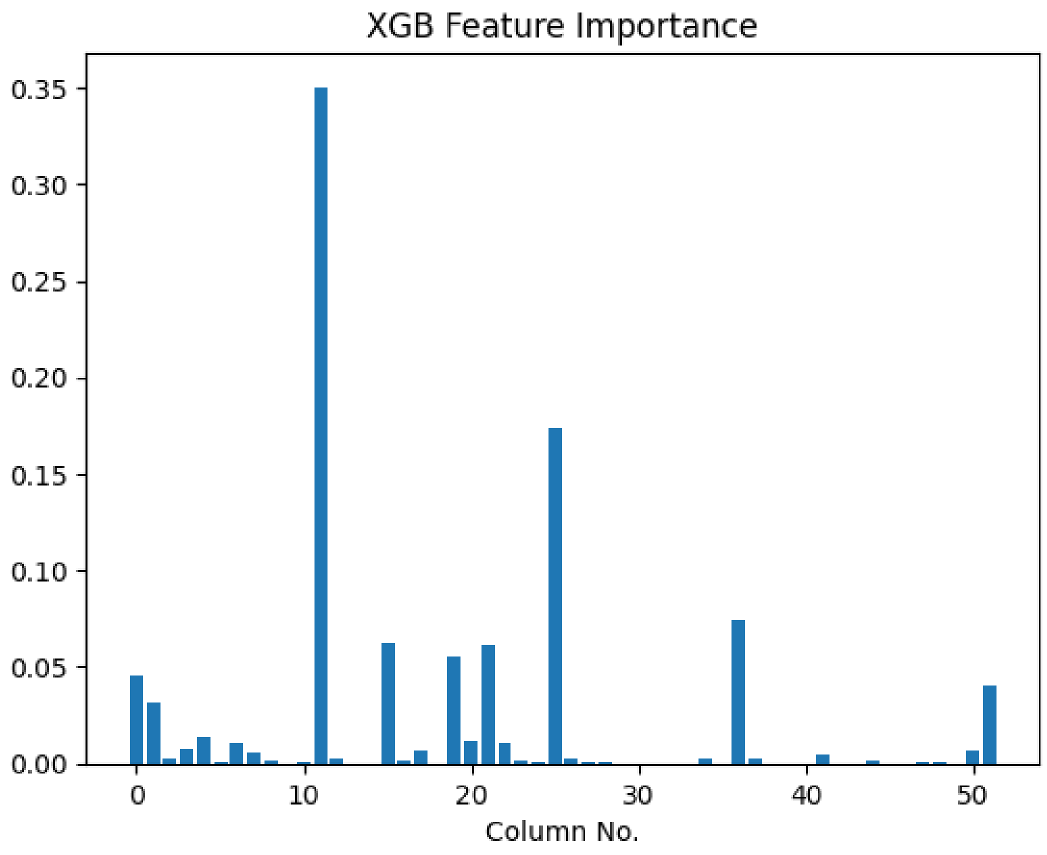


Figure 18. Feature importance using XGB.

The final feature importance algorithm used was Scikit-learn's KNeighborsClassifier. This algorithm was the only of the five to give the majority of the features a score of 0. The features calculated as being important here were ID, job, DID, added_by, campaign and passcode1.

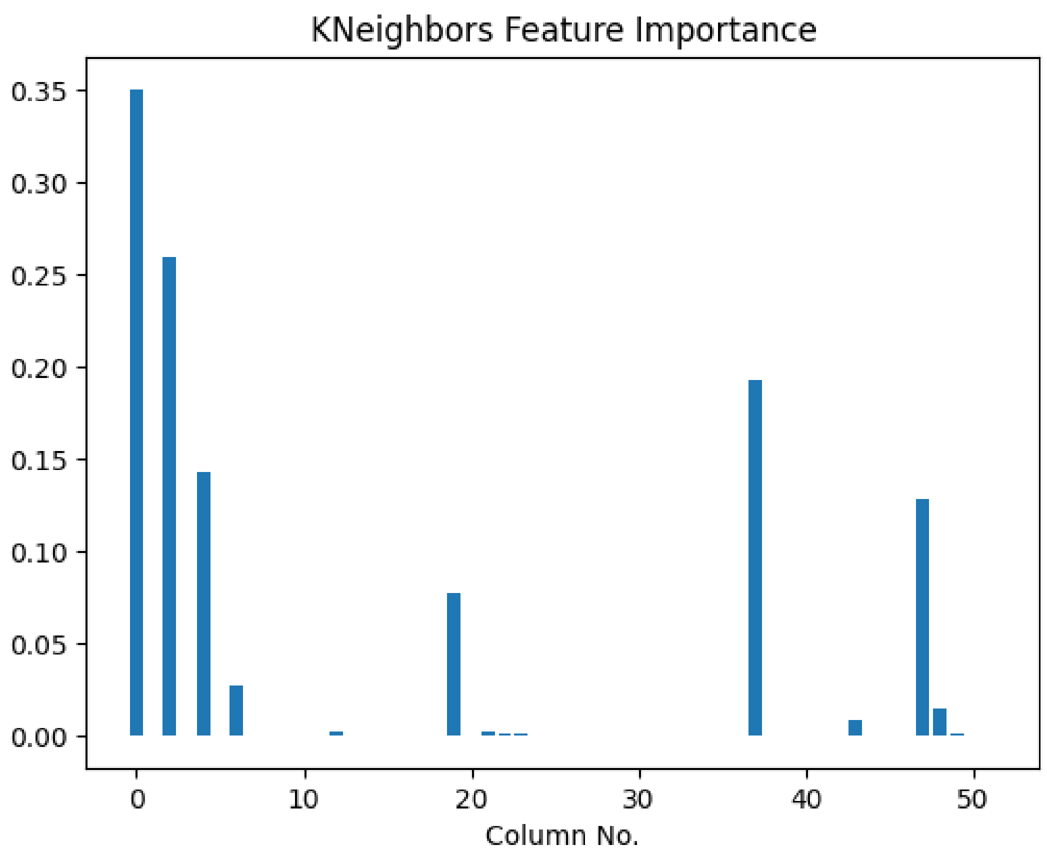


Figure 19. Feature importance using KNeighbors.

Below are the results from the five feature importance algorithms utilised. A “1” indicates that the algorithm said that the selected feature is important.

	LogisticRegression	**DecisionTree**	**RandomForest**	**XGB**	**KNeighbors**	**Total**
Columns						
Test_type	1			1		**2**
DID					1	**1**
Is_gsm	1					**1**
Call_description		1	1			**2**
ID		1	1	1	1	**4**
Test_counter		1	1			**2**
Job		1	1		1	**3**
Route			1			**1**
Added_by	1		1	1	1	**4**
Number		1	1			**2**
Campaign			1			**1**
Failure reason	1	1	1	1		**4**
Score_type				1		**1**
PESQ_server				1		**1**
Application				1		**1**
Passcode1					1	**1**

As this table tells us, the three features which were calculated as being important by four of the five algorithms were ID, added_by and failure_reason. Job was the only feature which was highlighted as being important, and test_type, call_description, number and test_counter each appeared twice.

All of the numerical columns were run through RandomForestClassifier, DecisionTreeClassifier and GradientBoostingClassifier. Their training and testing accuracy were noted.

With random forest, the training accuracy when every numerical column was used was 81.2% with a standard deviation of 0.3%. The accuracy of this model on the test set was 76.4%.

With decision tree, the training set accuracy was 81.9% with a standard deviation of 0.3%. The accuracy with the test set was 74.3%.

When these features and rows were run through the gradient boosting classifier algorithm, it did not perform nearly as well as the previous two models. Here, the cross-validated training accuracy was 45.9% with a standard deviation of 34.4%, and the testing accuracy was 51%. This model took several times longer than the others to run, with much worse performance, thus the idea of trying to get this algorithm to work well was abandoned.

These results are to be used as the benchmarks for these models. The objective of the future models is to achieve performance metrics higher than these. Accuracy, precision, recall and f-score are the metrics which will be gauged. When training the models, ten folds of cross validation were chosen using RepeatedStratifiedKFold with three repeats to reinforce the performance of the models.

Once feature importance analysis had been completed by comparing the outputs from each algorithm, feature selection was applied next. Here, several subsections of the features chosen as important were ran through classification models, namely random forest, to gauge how different combinations of features performed with the models before any hyperparameter tuning was applied.

The final set of features decided on were the **test type, call description, application** (slightly increases the performance of the models despite not being given a high importance by any of the feature importance models), **DID, the campaign and the failure reason.**

When these are run through random forest, the training accuracy increases to 82% with a standard deviation of 0.3% and the testing accuracy was 78.18%. The precision was 76.63%, recall was 78.18% and the f-score was 77.1%. The time needed to train and test the models was reduced drastically post feature selection. Instead of taking 15 minutes to perform these actions, it was now taking about 3 minutes.

The ID, failure_reason and added_by columns were the only ones chosen by four of the five features importance algorithms. When the failure_reason column is taken out of the final set of features, the training accuracy drops to 78.8% and the testing drops to 73.6%. When added_by is added to this final set, the performance of the model doesn't change, but when ID is added, the training accuracy drops to 79.2% with the testing being 72.9%.

With decision tree, these features achieved a training accuracy of 82% with a standard deviation of 0.3%, and a testing accuracy of 75.58%. Precision was 76.4%, recall was 77.98% and the f-score was 76.91%.

In the Data section, it was mentioned that when the call description ID was 3, the target variable was an almost 80:20 split between "marked busy" and "check call history". When this data through my models with the selected features, decision tree classifier achieved a training accuracy of 84.6% with a standard deviation of 2.3% and a testing accuracy of 77.7%. Random forest beat this, with its training accuracy being 86.1% (standard deviation 2%) and its testing being 79.2%.

When these steps were repeated with the call description ID 9 instances (88.92% "marked temporarily unable to test", 10.91% "check call history"), decision tree managed to predict 93.4% (standard deviation 1.1%) of the train accurately, with its testing performance being 85.9%. Random forest just about underperformed with the training data, achieving 93.3% with a standard deviation of 1.2% but it beat the previous model with the testing data, having an accuracy of 87.5%.

Once the models with all of the call description IDs included had successfully run using the selected features, the final step was to perform hyperparameter tuning. Here, Scikit-learn’s GridSearchCV function was used to try out each possible combination of hyperparameters chosen. The best and worst performing sets of parameters were returned, along with the accuracies of each. The accuracy of the average combinations of hyperparameters was also recorded, along with precision, recall and f-score.

5.1 Random Forest Hyperparameters

With the n_estimators hyperparameter for random forest, the best result seen was when the number of estimators was set to 100, although these five mean scores were all within a percentage point of each other. The accuracy achieved with the best value (100 estimators) was 77.98%.

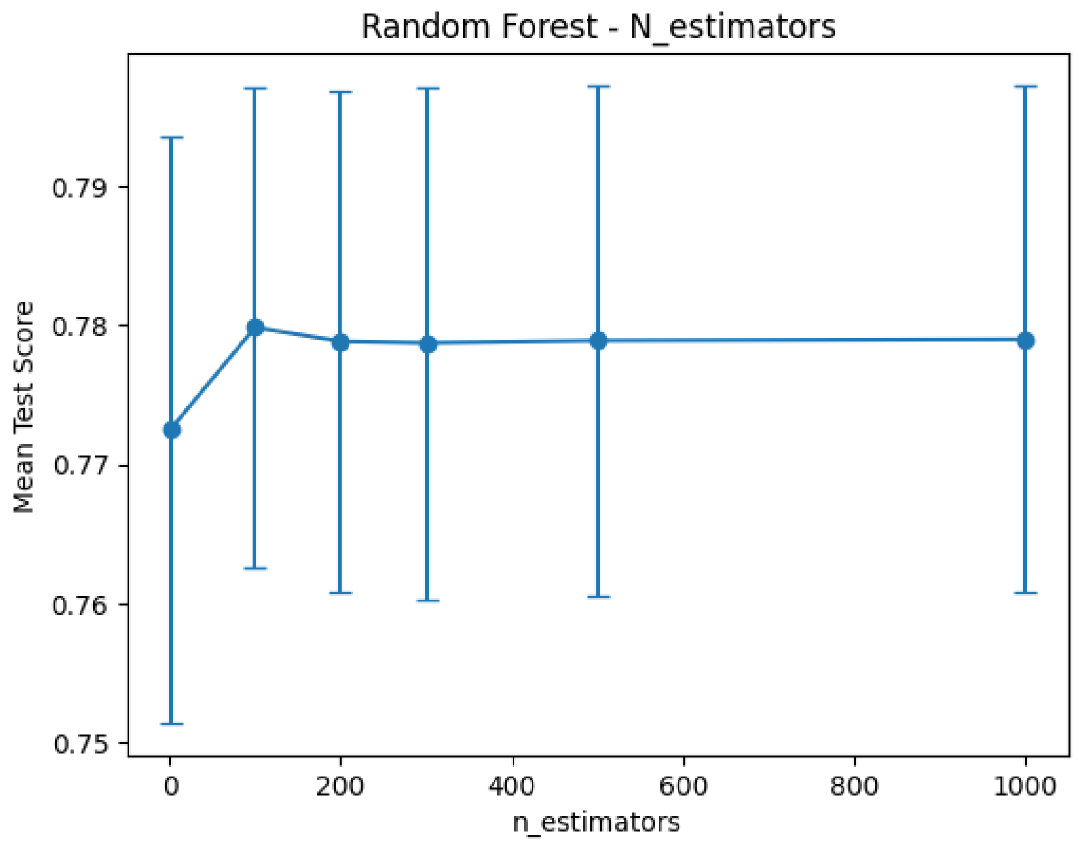


Figure 20. Accuracy of model when tuning n_estimators.

The next hyperparameter analysed was max depth. With this, the model's performance plateaued when the value was set to ten. The mean accuracy achieved with a maximum depth of ten was 78.6%.

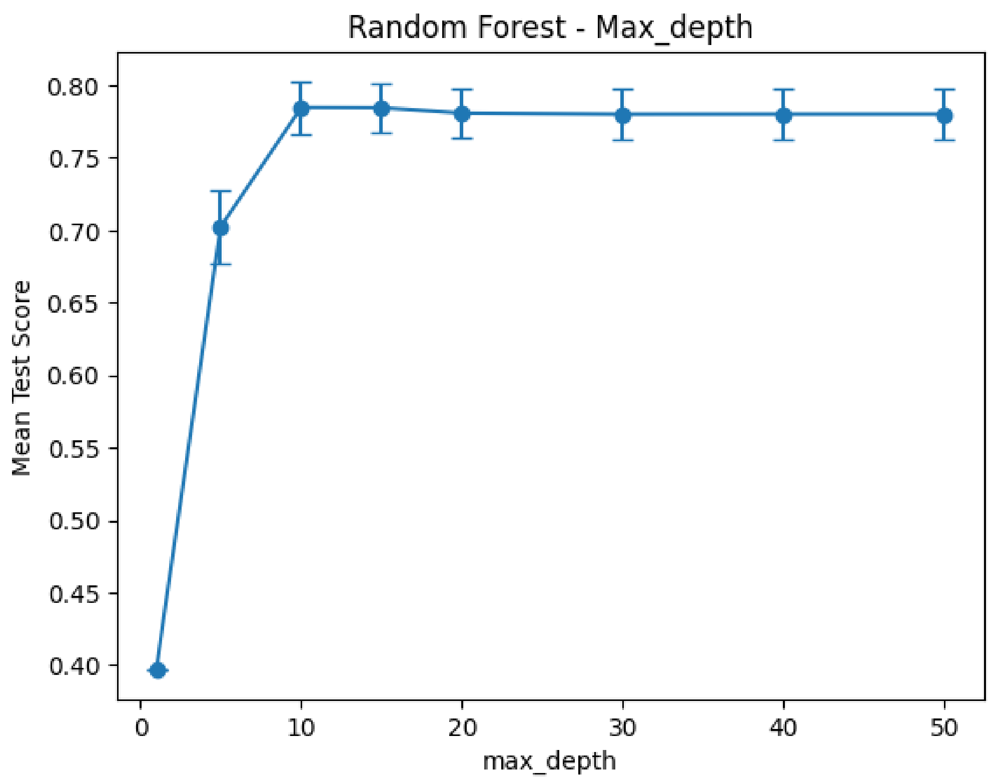


Figure 21. Accuracy of model when tuning max_depth.

The maximum features hyperparameter performed almost identically with both values, although square root came out on top. It achieved an accuracy of 77.98%

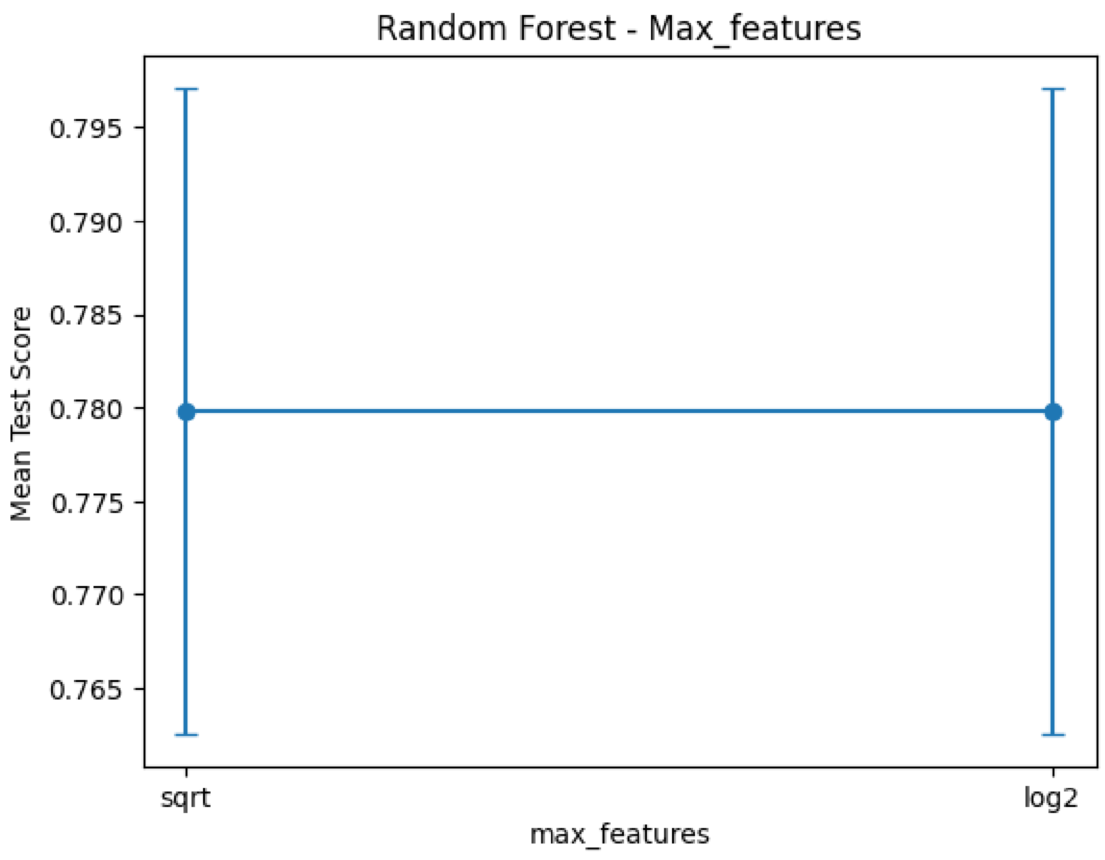


Figure 22. Accuracy of model when tuning max_features.

When minimum sample split is utilised with random forest, the best performance was seen when this value was set to fifty samples. The model achieved a test accuracy of 77.57% with this, although setting this hyperparameter to fifteen may be the safer option, as the performance was very close and the higher you set this parameter to be, the higher the chances of underfitting occur.

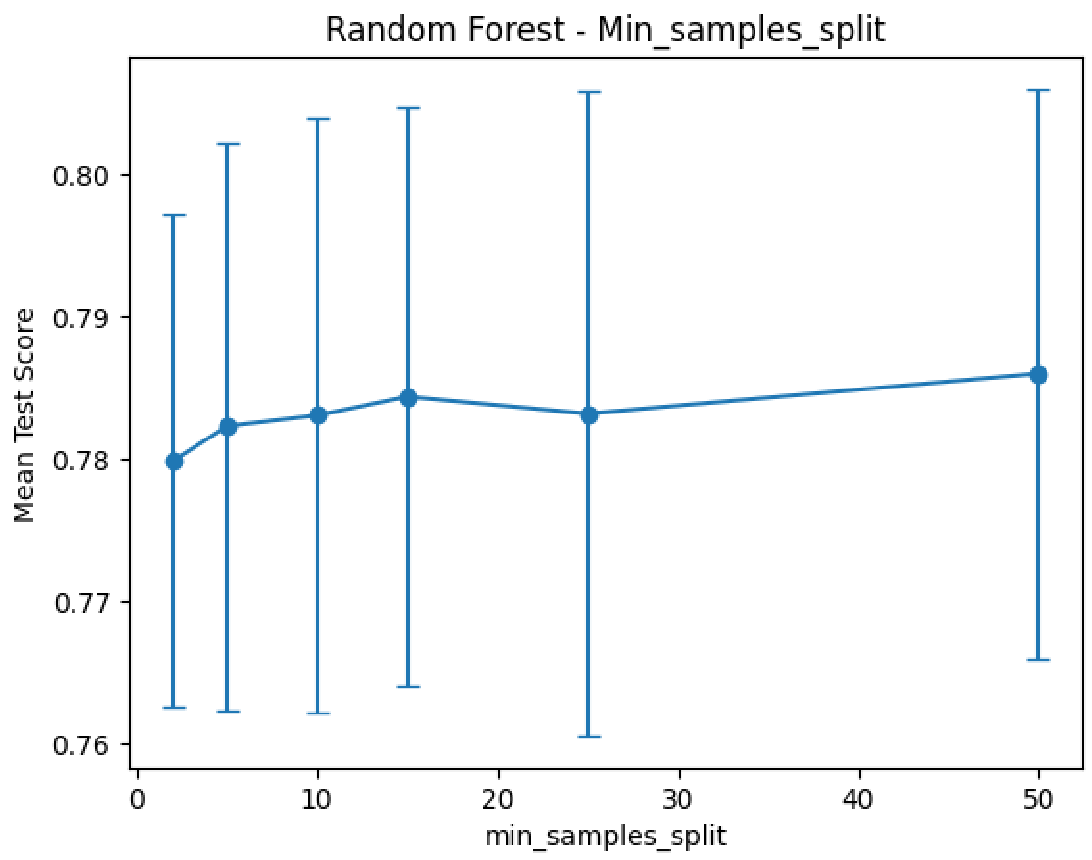


Figure 23. Accuracy of model when tuning min_samples_split.

With the minimum samples leaf hyperparameter, a similar trend was seen as in the above hyperparameter, except this time the value in the middle slightly performed better than the highest value. An accuracy of 77.99% was achieved when the minimum number of samples was set to four.

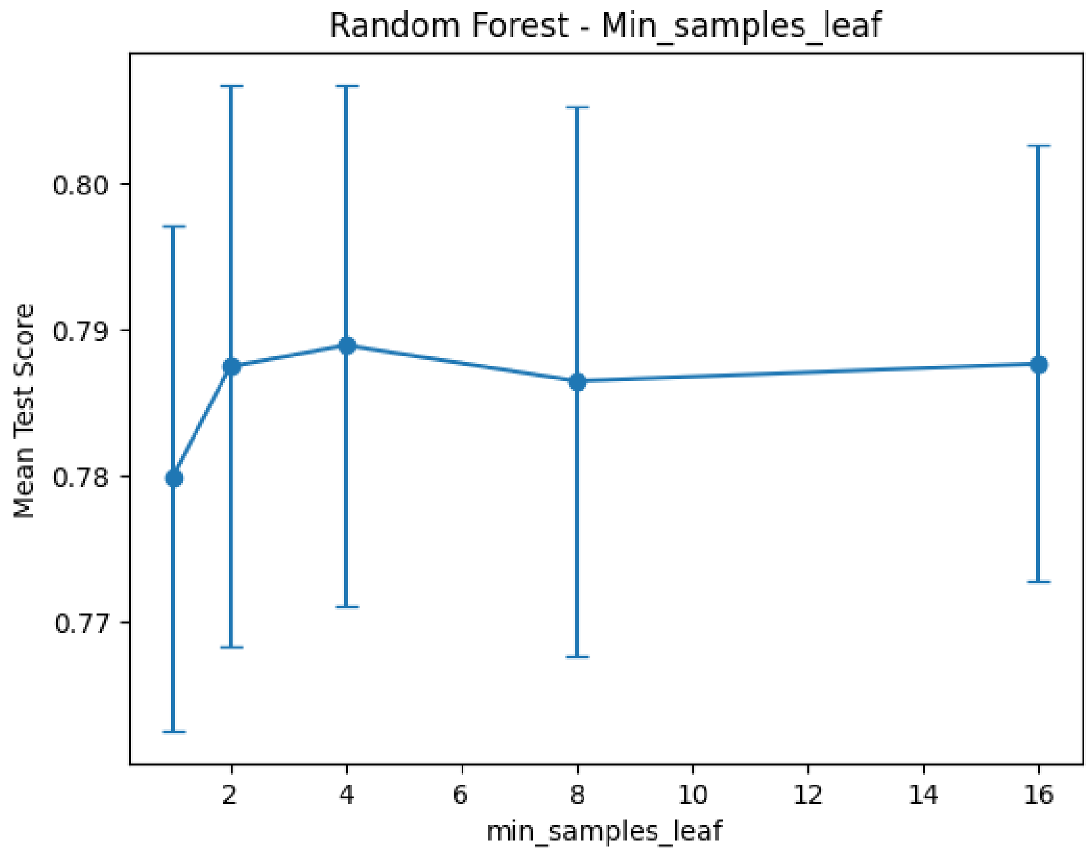


Figure 24. Accuracy of model when tuning min_samples_leaf.

The final hyperparameter analysed with my random forest model was bootstrap. “True”, represented on the x-axis as 1, performed slightly better than “false”, achieving an accuracy of 77.98%.

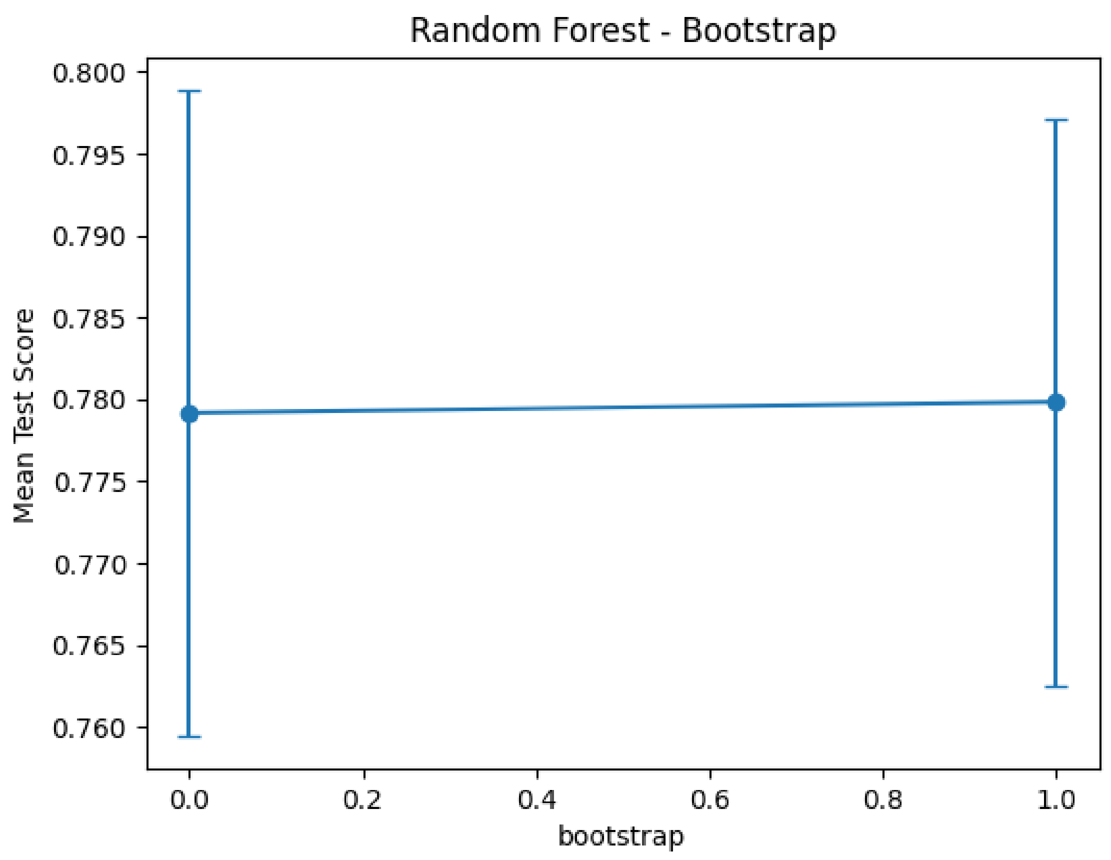


Figure 25. Accuracy of model when tuning bootstrap.

From these results, it is clear that max_depth has the greatest impact of the six on the performance of the model. It achieved the highest mean test accuracy with 78.6%. The hyperparameter with the least impact was min_samples_split, achieving a top accuracy of 77.57%.

The testing accuracy of the random forest model with the selected features was 78.18%, thus when these hyperparameters are only used one at a time, all but one of them had a negative impact on the performance of the model.

Hyperparameter	**Best value**	**Mean test accuracy**
N_estimators	100	77.98%
Max_depth	10	78.6%
Max_features	sqrt	77.98%
Min_samples_split	50	77.57%
Min_samples_leaf	4	77.99%
Bootstrap	True	77.98%

5.2 Decision Tree Hyperparameters

The first hyperparameter analysed with the decision tree model was criterion. The model performed extremely similarly with both values, but “entropy” ultimately performed better, with a mean accuracy of 76.55%.

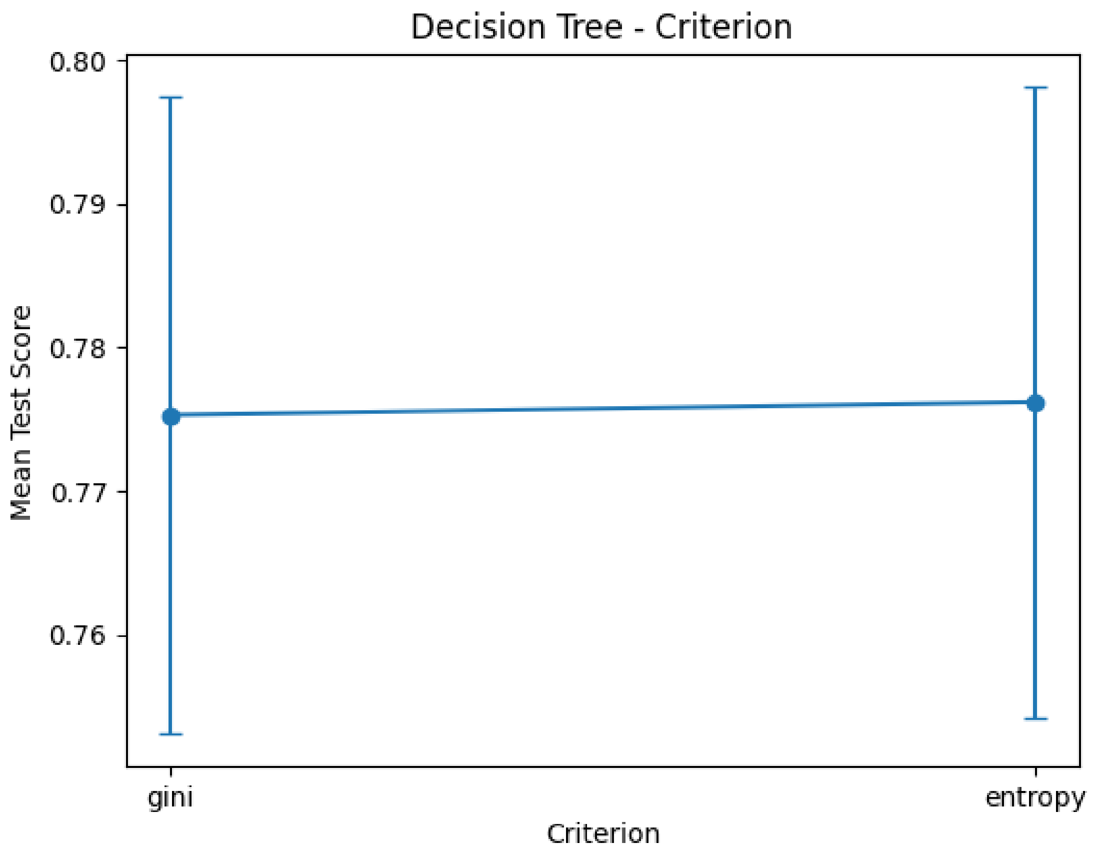


Figure 26. Accuracy of model when tuning citerion.

The next hyperparameter investigated was splitter. This also was given two options, with “random” beating out “best”. This value achieved a mean accuracy of 76.86%.

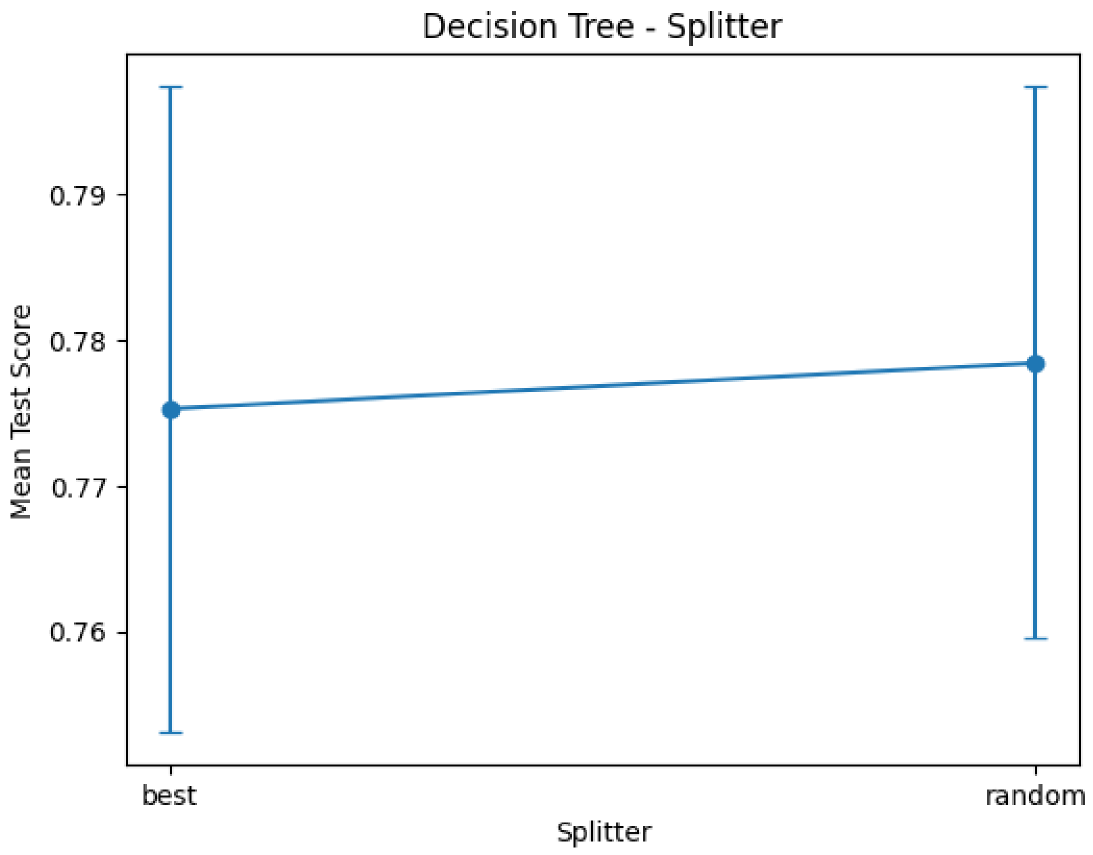


Figure 27. Accuracy of model when tuning splitter.

The third hyperparameter analysed was the maximum depth of a decision tree. The performance here peaked when the depth was set to fifteen, achieving an accuracy of 76.09%.

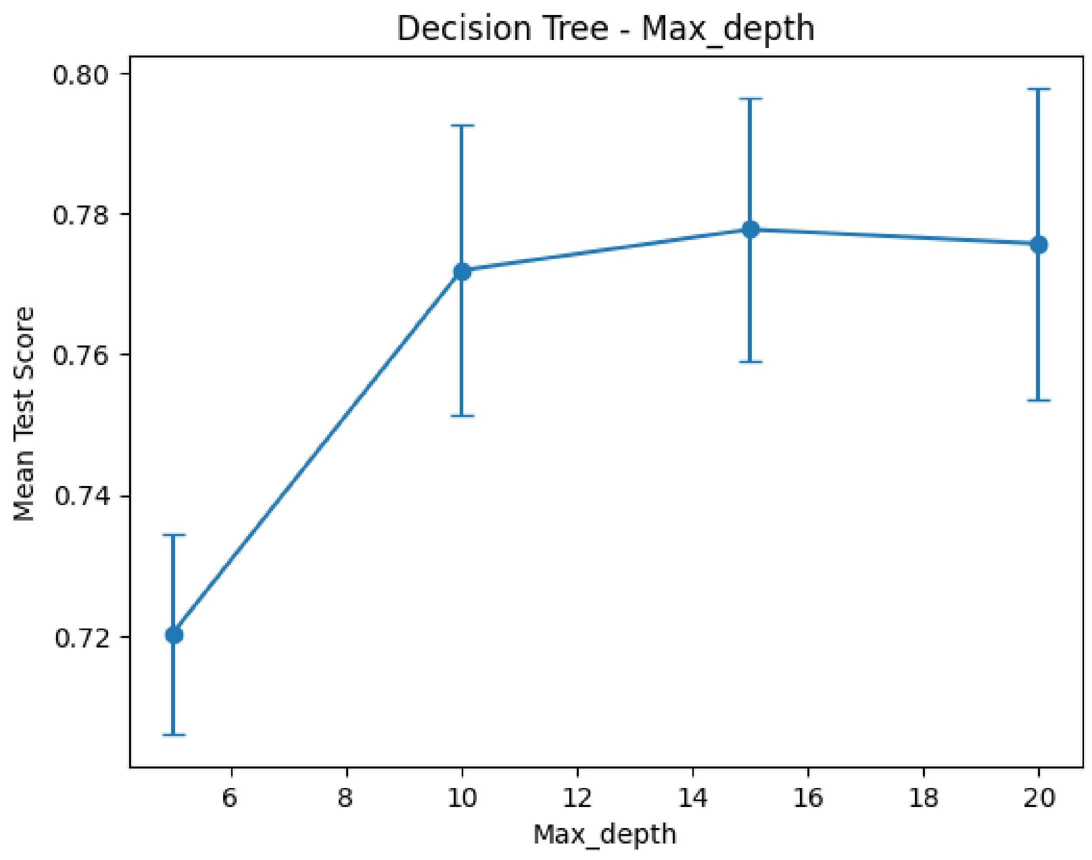


Figure 28. Accuracy of model when tuning max_depth.

The following hyperparameter analysed was the minimum number of samples for a split. The accuracy here peaked when the minimum number of samples was fifty. An accuracy of 74.4% was achieved with his value. After this, the performance of the model decreased, possibility due to underfitting from having the hyperparameter set too high.

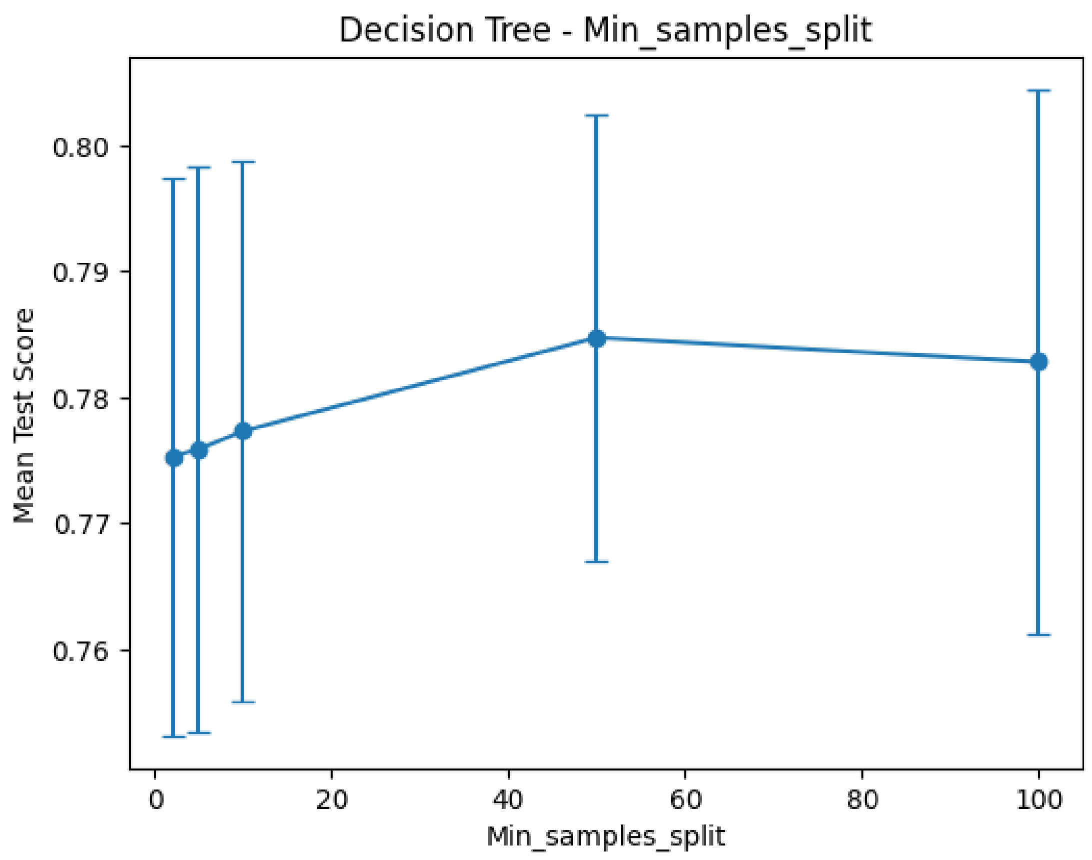


Figure 29. Accuracy of model when tuning min_samples_split.

The next decision tree hyperparameter analysed was the minimum number of samples per leaf. A similar trend is seen in the graph as with the previous hyperparameter. The best accuracy seen here was with eight samples, achieving an accuracy of 75.88%. After this the performance degraded, possibly due to underfitting.

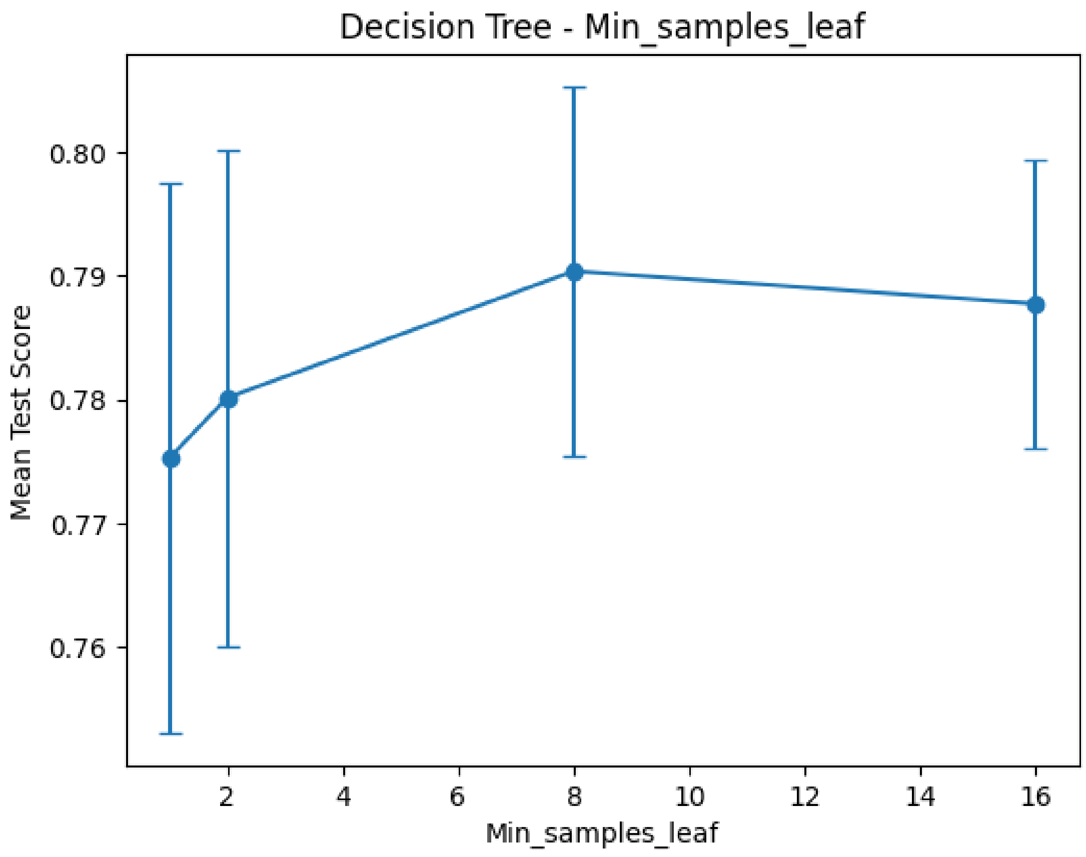


Figure 30. Accuracy of model when tuning min_samples_leaf.

The second last hyperparameter analysed was max_features. With this one, the best performing value was “None”, beating the other two, with a mean accuracy of 75.9%.

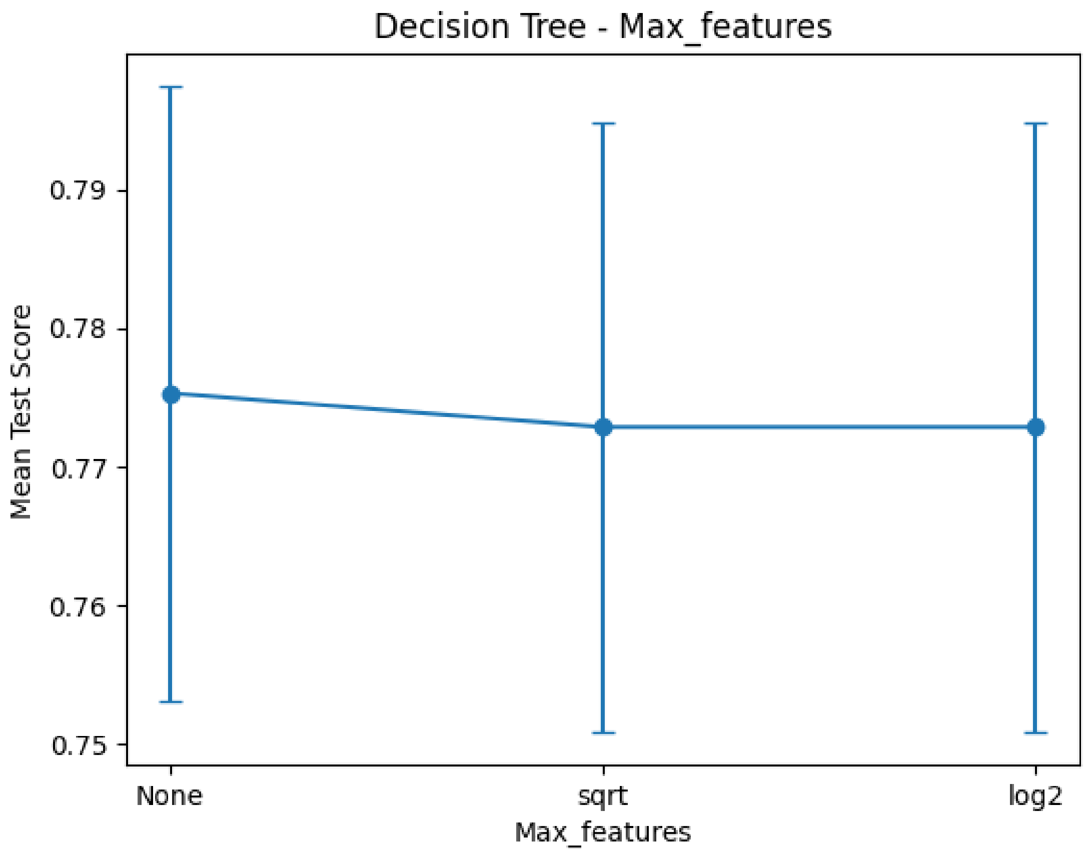


Figure 31. Accuracy of model when tuning max_features.

The final hyperparameter investigated was class_weight. The model performed much better when this value was set to "None" over "Balanced". With this better value it was able to predict 75.9% accurately.

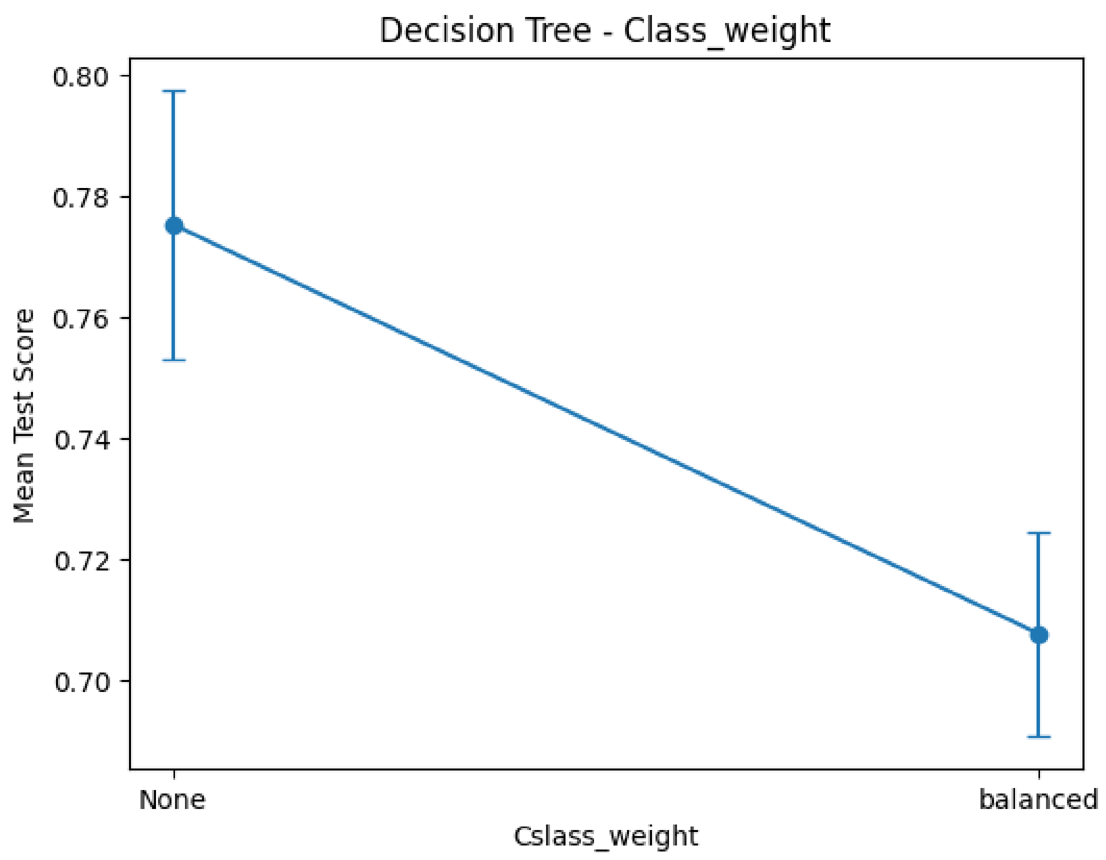


Figure 32. Accuracy of model when tuning class_weight.

From these results, the hyperparameter with the best impact on the performance of the model is splitter, achieving an accuracy of 76.86% when it's set to 'random'. The worst performing hyperparameter was min_samples_split, achieving a top mean accuracy of 74.4%.

Hyperparameter	**Best value**	**Mean test accuracy**
Criterion	entropy	76.55%
Splitter	random	76.86%
Max_depth	10	76.09%
Min_samples_split	50	74.4%
Min_samples_leaf	8	75.88%
Max_features	None	75.9%
Class_weight	None	75.9%

The testing accuracy of this model & features before hyperparameter tuning was 75.58%, thus, when on their own, six of the seven hyperparameters are able to increase the performance of the model.

6.0 Results

The final step with these models was to perform hyperparameter tuning using all the hyperparameters at once. The tuning was performed using five folds of cross validation.

With the **random forest** model, the best set of hyperparameters were:

{'bootstrap': True, 'max_depth': 15, 'max_features': 'log2', 'min_samples_leaf': 4, 'min_samples_split': 5, 'n_estimators': 200}

These achieved a training accuracy of 78.88% and a testing accuracy of 78.19%. The best testing accuracy seen before tuning was 78.18% and when tuning a single hyperparameter this was 78.6% with max_depth. The precision for this set of hyperparameters was 76.62%, recall was 78.19% and the f-score was 76.96%.

The average accuracy achieved during hyperparameter tuning was 75.4%, with the worst performing set achieving an accuracy of 68.4%. This set was:

{'bootstrap': True, 'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}

With the **decision tree** model, the best performing set of hyperparameters was:

{'class_weight': None, 'criterion': 'gini', 'max_depth': 20, 'max_features': None, 'min_samples_leaf': 8, 'min_samples_split': 2, 'splitter': 'best'}

This set achieved a training accuracy of 79.02%, with a testing accuracy of 75.73%. The highest testing accuracy achieved before tuning was 75.58%, and this figure reached as high as 76.86 when only the “splitter” hyperparameter was tuned.

With the best performing set of hyperparameters, the precision was 74.58%, recall was 75.73% and the f-score was 74.81%.

The worst performing set of hyperparameters with this model only predicted the right value 0.28% of the time, with the mean set of hyperparameters achieving an accuracy of 52.48%. The worst performing set was:

{'class_weight': None, 'criterion': 'gini', 'max_depth': 5, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}

Random Forest					
	Training Accuracy	**Testing Accuracy**	**Precision**	**Recall**	**F-score**
All features	81.2%	76.4%	76.14%	76.4%	76.2%
With selected features	82%	78.18%	76.63%	78.18%	77.1%
After hyperparameter tuning	78.89%	78.19%	76.62%	78.19%	76.96%

Decision Tree					
	Training Accuracy	**Testing Accuracy**	**Precision**	**Recall**	**F-score**
All features	81.9%	74.3%	75.6%	74.3%	74.7%
With selected features	78%	75.58%	76.41%	77.98%	76.92%
After hyperparameter tuning	79.02%	75.73%	74.58%	75.73%	74.81%

7.0 Conclusions

From the above tables, it is apparent that the while the training accuracy of the models decreased after feature selection and hyperparameter tuning, the accuracy of the models on the testing data increased.

This indicates that one of the strengths of this project is that the models are not overfitting the data during training. It also implies that the models are generalising better by becoming more accurate with new, unseen data. The models are also much more time efficient, requiring significantly less training and testing time than was needed before feature selection.

A weakness is that some of the metrics performed worse after hyperparameter tuning had occurred. This may be due to the fact that not enough values were given for some of the parameters, such as min_samples_split and min_samples_leaf for the decision tree model.

As this project uses real world data, an advantage is that Spearline Labs are able to use my models to speed up the process of their operations. While the models may not be accurate enough to let them input data to their database, they could still be used to speed up the process of employees understanding why a test failed. This would allow the employees to first check to see if the model's prediction is correct instead of looking at the data and considering every possible value for the target variable.

8.0 Further Development or Research

With additional time and resources, the first two steps I would take would be to perform more hyperparameter tuning and feature selection. From the results I achieved, they give the impression that the set of features wasn't quite perfect, and the models, especially decision tree, could've performed better if given more hyperparameter values to choose from. The problem with the latter step is that this tuning takes quite a long time, and this computational time increases exponentially as new values are added.

With additional time, I would also have more data to train and test the models on, improving their robustness. I could also spend more time training other models, such as gradient boosting classification.

I would also experiment with using deep learning/neural networks to make predictions with this data, which could then be compared to the performance of the machine learning models created here.

9.0 References

[1] Kazim*, M., Doreswamy , H. and Dr. Doreswamy (2019) 'Machine Learning Based Network Anomaly Detection', *International Journal of Recent Technology and Engineering (IJRTE)*, 8(4), pp. 542–548. doi:10.35940/ijrte.d7271.118419.

10.0 Appendices

10.1. Project Proposal

1.0 Objectives

The objective of this project is to create a reinforcement learning model(s) to mimic the actions taken by one of Spearline Lab's Support Team employees. At the moment, whenever one of Spearline's tests fail, a member of their support team is tasked with manually analysing the test to figure out why it failed, then they carry out any further steps necessary or mark into their database the failure reason. As this must be done manually, it is slow, costly and prone to human error. The goal of my model will be to receive all of data pertaining to a failed test through Spearline's API as soon as they're generated in their database. It will then use rule-based logic or potentially machine learning to predict if another step is to be taken or if the test can have its failure reason marked to finalise that test and move onto the next. Ideally, my model will take a fraction of a second to do a task which normally takes up to a few minutes. Hardware limitations or API connection times will most likely be the most influential factor in relation to the speed of my model. Once my model has decided on what to do with a test it will send the data back through the API and await updates.

2.0 Background

I chose to undertake this project as I did my work placement with Spearline thus I'm quite familiar with the data and am interested in pushing the boundaries of what advanced use cases can be utilised efficiently with said data. I also chose it as I want to test and expand my abilities using Python, MySQL, problem-solving and working with API's. The fact that I may have to introduce machine learning or even deep learning into my project is also enticing. I will meet my objectives by either receiving the dataset or connecting to the API and creating my own dataset, before analysing the data to understand all of the features and their metadata. I will then build a basic model to use as my foundation, over time increasing its complexity to enable it to carry out more tasks on the failed tests until I reach the limit of what I can do. Throughout this process I will tinker with hyperparameters in an attempt to improve the accuracy of my outputs. I will compare my results to the target data as it comes in and through analysis will be able to figure out any reasons why my model may be inaccurate. I will also be in constant contact with the data science team in Spearline to get feedback from them on my progress as I undertake this project.

3.0 State of the Art

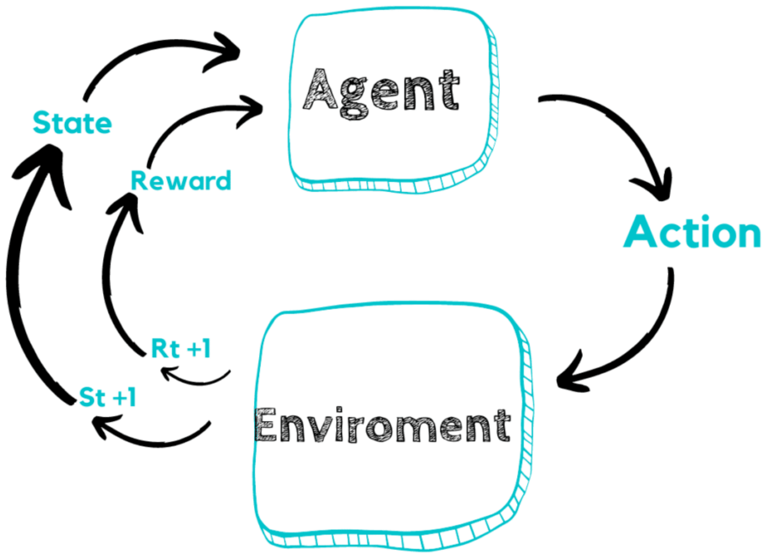
At the moment in Spearline, they have a deep learning model which predicts one of a few failure reason for a few specific test types. They also have a few other scripts created by me (hence why I'm already familiar with a lot of the data), which use vary basic rule-based logic to auto mark a few specific test types with specific failure reasons. The objective of my project would ideally be to create one script which will perform all these actions at a higher accuracy along with increasing the number of failure reasons for each of these test types already being auto marked and broadening the scope of test types to be automatically marked. As Spearline are constantly creating new test types and adding new possible failure reasons, ideally my model would be able to deal with these new features with very minimal changes needed to be made to the script.

4.0 Data

To accomplish the objectives of this project, I will need data containing many instances of each test type and each failure reason associated with a test type. The dataset must contain any and all data in relation to each test so that I can understand why a test was given a certain failure reason and why specific steps were taken before the test got marked. At the moment the plan is to work with a month's worth of data as there should be around 2.4 million tests to explore an to train my model with. The dataset will contain various data types, such as dates and times, categorical data for company and server names, and numerical data for ID values and metrics gathered about each test. These metrics may contain call connection times, or the quality of the audio being transmitted. Many of the tests will also have an MP3 recording of the audio heard by Spearline's system thus data can be extracted from said MP3's so that patterns or key insights can be found, whether it be silence on the line or by using a speech-to-text model to search for key words. If the phrase "Invalid passcode" was present in the audio file, then it would be incredibly simple to auto mark any test containing that phrase. As the data will contain ID values for many of the features, it will be easy to split the data by test type, failure reason, company running the test etc. This will be useful for visualisations and for working with smaller portions of the data, whether it be to figure out failure reasons for specific test types or if there's not enough memory on a system to hold the entire dataset. As Spearline's database is meticulously structured and organised, there will hopefully be not a great deal of cleaning to be done to the data, thus exploring the data and working on the model can begin quickly. As much as I would love to use Panda's "dropna()" feature, special attention will have to be given to any NULL values, as depending on the failure reason of a test, it may be correct for that test to lack a value for a feature. As of writing the plan is for the dataset to be emailed to me once it's created, although if this isn't done, I will be given access to the API to gather the data myself.

5.0 Methodology & Analysis

The first step in my methodology will be to understand what data will be required to fulfil my objectives. Once this is completed the dataset can be gathered. After this, I can begin cleaning the data. Each feature will must have the correct data type and any invalid tests containing incorrect data will have to be removed. Exploring the data is next. I will investigate the descriptive statistics of each numerical column, looking at the averages, standard deviations, min/max values etc. Visualisations will also be created to further my understanding of the data, by taking steps such as exploring the distribution of the data and looking at the outliers. To do this I will most likely use Matplotlib, Seaborn and Tableau. I will then have the option to split the dataset into subsets for each test type, failure reason, company etc, if it looks like by doing so it'll help with the creation of my model. As this will be a reinforcement learning model, the agent will utilise its rewards in order to make its actions as accurate as possible in the environment.



When working on my model I'll at first start off simple, using rule-based logic on a test type basis to try to get as many tests marked accurately as possible without having to create a complicated model which will produce incorrect outputs if not tuned correctly. As I cover more use cases for the model, it will inevitably become increasingly more complex. A deep reinforcement learning model will most likely be required for some of the failure reasons which can be understood by a human but not by any rule-based logic or machine learning algorithm.

6.0 Technical Details

For this project rule-basic logic will be utilised at first to mimic the actions of a Support Team employee, before the reinforcement learning model will be created. This model will most likely use a Q-learning algorithm and a Markov Decision Process to carry out its analysis and predictions. Hyperparameter tuning will be utilised to increase the accuracy of my model(s). Python will be used to create these. The Python module Pandas will likely be heavily utilised to load the data, to clean it and to explore it. Matplotlib, Seaborn (built on Matplotlib) and Tableau will be used to create visualisations. Tableau will be extremely useful for the creation of dashboards, so that various aspects of the data can be plotted next to each other, making it elementary for anyone to understand the data after at a brief glance.

7.0 Project Plan

I am currently awaiting to be sent the dataset, but as soon as I receive it, I will begin working on and with it. I will at first have to clean and gain an understanding of it, of each feature, each test type and each failure reason. Once this is completed, I will then be able to perform my data analysis. I will do this by creating visualisations using Matplotlib, Seaborn and Tableau. I will also have to perform non-visual analysis on the data, by utilising descriptive statistics. As I'm doing this, I will be able to find outliers in the data and get a clearer understanding of how the data is supposed to work in theory, and how it is represented in reality. Once this is completed, I can begin working on any rule-based logic applicable to auto mark these failed tests. Once this has been utilised completely, I shall introduce reinforcement learning. I will attempt to use this at first with the test type and failure reason combinations which theoretically shouldn't be on the more complex side of all these combinations. I will tune my model(s) to mimic the actions of a Support Team employee with an accuracy of > 99.1%, so that my model is proved to be more accurate than the average human. I will continue to increase the functionality of my model(s) as time goes on, expanding the number of test types and failure reasons it covers and I will utilise hyperparameter necessary to achieve the maximum accuracy I'm capable of. While I'm doing all of this, I will be in constant contact with my supervisor, looking for feedback and advice on how to improve my project and to keep him updated on my progress. The Data Science Team lead at Spearline will also be updated on my progress and will be contacted by me whenever I have a question regarding the data. I will also have to complete all my deliverables on time, those being the monthly project ethics form, the monthly journals, the mid-point implementation, documentation and video presentation, the final implementation, documentation and video presentation and the project showcase.

10.2. Ethics Approval Application (only if required)

National College of Ireland

DECLARATION OF ETHICS CONSIDERATION

School of Computing

Student Name:	………Jordan O'Donovan………………………………………………………………………………
Student ID:	……X19372016…………………………………………………………………………………..………………
Programme	Bachelor of Science (Hons) in Data Science **Year:** ……4…………………..
Module:	…………Computing Project…………………………………………………………………………………
Project Title:	………… Mimicking the Actions of a Support Employee Using Machine Learning……………………………….………

Please circle (or highlight) as appropriate

This project involves human participants	Yes / [✓ No]

Introduction

Secondary data refers to data that is collected by someone other than the current researcher. Common sources of secondary data for social science include censuses, information collected by government departments, organizational records and data originally collected for other research purposes. Primary data, by contrast, is collected by the investigator conducting the research.

A project that does not involve human participants requires ONLY completion of Declaration of Ethics Consideration Form and submission of the form on module's Moodle page

A project that involves human participants requires ethical clearance and an Ethics Application Form must be submitted through the module's Moodle page. Please refer to and ensure compliance with the ethical principles stated in NCI Ethics Form available on the Moodle page.

The following decision table will assist you in deciding if you have to complete the Declaration of Ethics Consideration Form or/and the Ethics Application Form.

Public Data	Y	Y	Y	Y	N	N	N	N
Private Data	Y	Y	N	N	Y	Y	N	N

Human Participants	Y	N	Y	N	Y	N	Y	N
Declaration of Ethics Consideration Form	x	X	x	X	X	X	x	
Ethics Application Form	X		X		X		X	

Please circle (or highlight) as appropriate

The project makes use of secondary dataset(s) created by the researcher	Yes / [✓ No]
The project makes use of public secondary dataset(s)	Yes / [✓ No]
The project makes use of non-public secondary dataset(s)	[✓ Yes] / No
Approval letter from non-public secondary dataset(s) owner received	[✓ Yes] / No

Sources of Data:

It is students' responsibility to ensure that they have the correct permissions/authorizations to use any data in a study. Projects that make use of data that does not have authorization to be used, will not be graded for that portion of the study that makes use of such data.

Public Data

A project that makes use of public secondary dataset(s) ***does not need ethics permission****, but* ***needs a letter/email from the copyright holder*** *regarding potential use.*

Some websites and data sources allow their data sets to be used under certain conditions. In these cases, a letter/email from the copyright holder is NOT necessary, but the researcher should cite the source of this permission and indicate under what conditions the data are allowed to be used. See Appendix I for examples of permissions granted by Fingal Open Data, and Eurostat website.

Where websites or data sources indicate that they do not grant permission for data to be used, you will still need a letter/email from the copyright holder. For example, see Appendix II for an example from the Journal of Statistics Education.

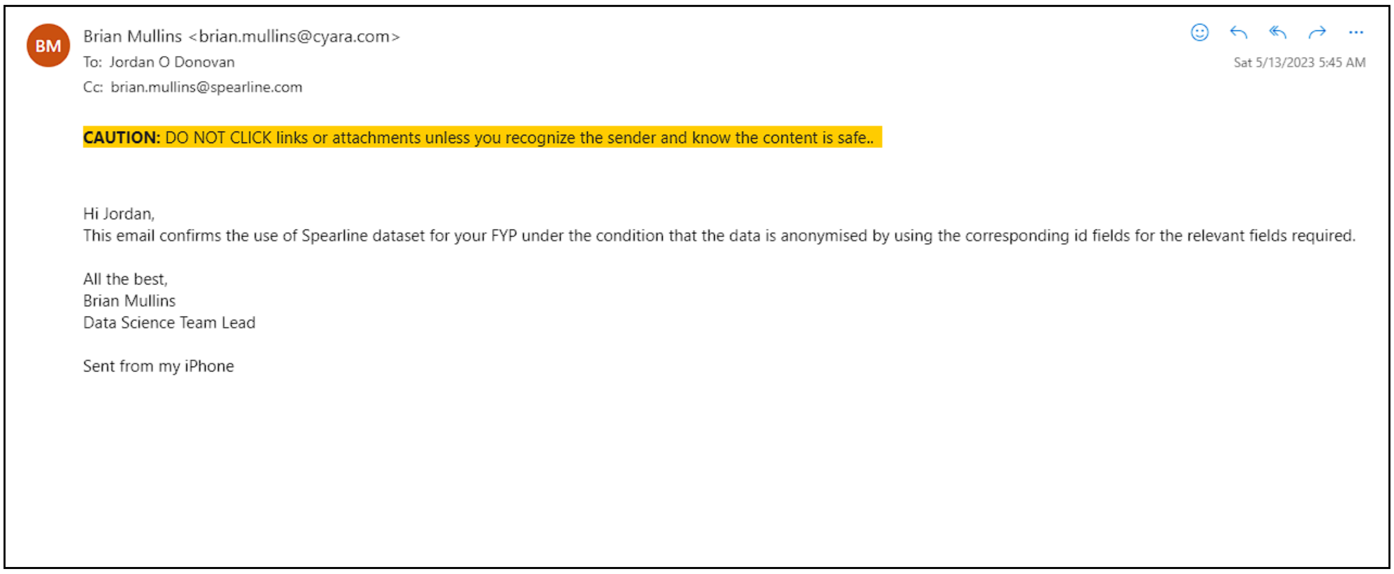
Private Data

A project that makes use of non-public (private) secondary dataset(s) must receive data usage permission from School of Computing.

An approval letter/email from the owner *(e.g. institution, company, etc.)* ***of the non-public secondary dataset must be attached to the Declaration of Ethics Consideration****. The letter/email must confirm that the dataset is anonymised and permission for data processing, analysis and public dissemination is granted.*

Evidence for use of secondary dataset(s)

Include dataset(s) owner letter/email or cite the source for usage permission


BM

Brian Mullins <brian.mullins@cyara.com>
To: Jordan O Donovan
Cc: brian.mullins@spearline.com

Sat 5/13/2023 5:45 AM

CAUTION: DO NOT CLICK links or attachments unless you recognize the sender and know the content is safe..

Hi Jordan,
This email confirms the use of Spearline dataset for your FYP under the condition that the data is anonymised by using the corresponding id fields for the relevant fields required.

All the best,
Brian Mullins
Data Science Team Lead

Sent from my iPhone

CHECKLIST

Non-public/private secondary dataset(s) -Owner letter/email is attached to this form ***OR*** Citation and link to the web site where permission is granted – provided in this form	[✓ Yes] / No Yes / [✓ No]

ETHICS CLEARANCE GUIDELINES WHEN HUMAN PARTICIPANTS ARE INVOLVED

The Ethics Application Form must be submitted on Moodle for approval prior to conducting the work.

<u>Considerations in data collection</u>

- Participants will not be identified, directly or through identifiers linked to the subjects in any reports produced by the study
- Responses will not place the participants at risk of professional liability or be damaging to the participants' financial standing, employability or reputation
- No confidential data will be used for personal advantage or that of a third party

<u>Informed consent</u>

- Consent to participate in the study has been given freely by the participants

- participants have the capacity to understand the project goals.
- Participants have been given information sheets that are understandable
- Likely benefits of the project itself have been explained to potential participants
- Risks and benefits of the project have been explained to potential participants
- Participants have been assured they will not suffer physical stress or discomfort or psychological or mental stress
- The participant has been assured s/he may withdraw at any time from the study without loss of benefit or penalty
- Special care has been taken where participants are unable to consent for themselves (e.g children under the age of 18, elders with age 85+, people with intellectual or learning disability, individuals or groups receiving help through the voluntary sector, those in a subordinate position to the researcher, groups who do not understand the consent and research process)
- Participants have been informed of potential conflict of interest issues
- The onus is on the researcher to inform participants if deception methods have to be used in a line of research

I have read, understood, and will adhere to the ethical principles described above in the conduct of the project work.

Signature: ……Jordan O’Donovan……………………………………………………………………

Date: ……30/10/2022……………………………………………………………………………

Appendix I

1) Fingal Open Data: http://data.fingal.ie/About

Licence

Citizens are free to access and use this data as they wish, free of charge, in accordance with the Creative Commons Attribution 4.0 International License (CC-BY).

Note: From November 2010 to July 2015, data on Fingal Open Data was published in accordance with the PSI general licence.

Use of any published data is subject to Data Protection legislation.

Licence Statement

Under the CC-BY Licence, users must acknowledge the source of the Information in their product or application by including or linking to this attribution statement: "Contains Fingal County Council Data licensed under a Creative Commons Attribution 4.0 International (CC BY 4.0) licence".

Multiple Attributions

If using data from several Information Providers and listing multiple attributions is not practical in a product or application, users may include a URI or hyperlink to a resource that contains the required attribution statements.

2) Eurostat: https://ec.europa.eu/eurostat/about/policies/copyright

COPYRIGHT NOTICE AND FREE RE-USE OF DATA

Eurostat has a policy of encouraging free re-use of its data, both for non-commercial and commercial purposes. All statistical data, metadata, content of web pages or other dissemination tools, official publications and other documents published on its website, with the exceptions listed below, can be reused without any payment or written licence provided that:

- the source is indicated as Eurostat

- when re-use involves modifications to the data or text, this must be stated clearly to the end user of the information

Appendix II

Journal of Statistics Education: http://jse.amstat.org/jse_users.htm

JSE Copyright and Usage Policy

Unlike other American Statistical Association journals, the Journal of Statistics Education (JSE) does not require authors to transfer copyright for the published material to JSE. Authors maintain copyright of published material. Because copyright is not transferred from the author, permission to use materials published by JSE remains with the author. Therefore, to use published material from a JSE article the requesting person must get approval from the author.

10.3. Reflective Journals

Student Name	Jordan O’Donovan
Student Number	X19372016
Course	Bachelor of Science (Hons) in Data Science
Supervisor	William Clifford

Month: October

What?
This month, I have been given a few project ideas by the company I did my third-year placement with, Spearline Labs LTD, a new telecommunications company based in Skibbereen, Cork. I decided on mimicking the actions taken by a member of their Support team using reinforcement learning, thus I spent the majority of the month researching into this machine learning concept; namely by reading articles and watching YouTube videos on the subject.
Towards the end of the month, I created my project proposal and filled out the ethics declaration form. While I’ve been given written communication by the Data Science Team lead in Spearline allowing me to use their data for my project, they have not sent me nor the college a document detailing this permission, nor have they provided an NDA to be signed thus far.

So What?
As they have not begun gathering the data for this project thus far, in terms of written progress for this project (specifically coding) the project has not progressed at all. The only progress made so far is expanding my understanding of reinforcement learning, as at the beginning of this month the only thing I knew about the technology went as far as its name. My successes were vastly improving my understanding of the technology and becoming somewhat more familiar with how one would write Python code for it.
The challenges which still remain are to further expand upon my knowledge of the topic, and on how to code it.

Now What?
What can you do to address outstanding challenges?
At the moment, all I can do is research further into the technology, by reading more articles, watching more videos, and by looking at reinforcement learning examples on websites like Kaggle and GitHub, until I am sent data. Once that happens, I can begin cleaning, exploring and creating models using the data.

Student Signature	Jordan O'Donovan

Month: November

What? This month, Spearline began compiling the dataset, but they have not sent me any data yet, thus I spent the month researching into the technologies I'll use and coming up with ideas on how and what could be done with the project. Just like the previous month, I spent a lot of time reading articles about the topic (namely on TowardsDataScience) and I looked through many GitHub repos which utilise reinforcement learning. I have what I believe is a decent understanding of the technology although almost all of the examples I've seen thus far of Python code utilising this technology use the Gym module instead of showcasing how to clean and pre-process data for this problem.	
So What? This has meant that my knowledge and confidence for this project has increased significantly. My successes were improving my knowledge on the topic and further clarifying with Spearline what is to be done for this project and what they're expecting to see. The main challenge that still remains is creating the code for this project. As they have not sent any data yet, I haven't been able to write any code for it but as I am quite familiar with their data from my time with them during my placement, I expect that the analysis and cleaning can be done very quickly once I receive the data, and work on the models will begin almost immediately. During my time with them I created a few machine learning models, so I expect to be able to transfer a lot of the knowledge gained from that to this project.	
Now What? In terms of coding, there is nothing I can do for the data they'll send me. I have been experimenting with other data to become more familiar with how the code should be written and I'm currently looking for a course on reinforcement learning. I usually use DataCamp as its provided by the college and in my opinion extremely useful, but unfortunately, they do not have any courses which utilise reinforcement learning as of writing. I expect to be sent the first dataset at the beginning of December, so I will begin my cleaning and analysis as soon as that arrives. I will also begin working on my midpoint presentation, which is due on the final week of the semester.	
Student Signature	Jordan O'Donovan

Month: December

What?
This month, I received the first two months of data, thus I was able to begin the real work for this project. I performed by cleaning, pre-processing and EDA, and I created several models when experimenting the effectiveness of them on the data, before any complex changes are made.
I also created my midpoint presentation for the project, that being the midpoint report, the PowerPoint slides, and the video presentation.

So What?
My successes this month were the cleaning and pre-processing of the data. This was tricker than I had initially expected as I wrongfully assumed I would be sent CSV files containing the data, but I was instead sent text files containing log outputs. I eventually managed to get the data into a DataFrame correctly, so when I'm sent more data in the coming months, I'll be able to run the files through my code, thus saving a lot of time trying to manually clean it.
I was able to get my machine learning and deep learning models working and creating predictions with the data, with varying degrees of accuracy.
Something concerning I've realised this month is that this data is not suited for reinforcement learning. Supervised classification machine learning models are what will likely be the best solution for this problem, however it was Spearline's idea for me to do this using reinforcement learning (which is possible but likely not as efficient) so I will need to speak to them about my findings and ask which direction they want me to proceed in.

Now What?
For my models created so far, I can perform more feature selection and hyperparameter tuning to increase their performance. I must also wait for more data to be sent to me so that the models have larger datasets to be trained using.
For the issue arisen with reinforcement learning, I will attempt to create a model which will produce predictions on the data, but as of writing this seems like an extremely tricky task.
I will also try to perform more EDA in the hopes of uncovering key information not yet found in the data.
For next month I plan on having more EDA done and I plan on having the performance of my models improved.

Student Signature	Jordan O'Donovan

Month: January

What? At the beginning of the month, I received December's data. I added it to the total dataset, cleaned it and updated all my statistics and visualisations to include this new data. I've been continuing to find new insights into the data and have been working on improving the performance of my models. I spoke to Spearline about my concerns involving the usage of reinforcement learning, and they agreed the idea should be abandoned in favour for supervised classification machine learning.	
So What? The successes this month were adding to the EDA of the data, finding a few new ways of cleaning it which improved the performance of the models slightly, and abandoning the reinforcement learning aspect of the project. I was dedicating most of my time into that section, with very little to show for it. This data is not suited for a reinforcement learning task, but if I dedicate more hours to this project, I will be able to make up for the time spent trying to force reinforcement learning to work.	
Now What? To increase the performance of my models, I will be spending more time on the feature importance and feature selection for them. This will ideally not only improve their performance, but also reduce the amount of computational power and thus time needed to train and test the models. As I receive more data and try to make my models more advanced, my hardware struggles to keep up.	
Student Signature	Jordan O'Donovan

Month: February

What?
This month I received January's data and have cleaned it, added it to my previous data and rerun all of the pre-processing/visualisations using my combined dataset. I have also continued to work on my machine learning and neural network models. I'm currently experimenting with hyperparameter tuning for both types of models to improve their performance, but this takes a lot of time, so I've begun looking into services like AWS or Google Colab for more computing power.
So What? My successes this month were mainly furthering the performance of my models and increasing the amount of data I have to work with. I now have over twenty-five thousand more instances to train my models with. Another success I had was furthering my knowledge on how to use hyperparameter tuning with deep learning models. The challenges which remain are to increase the performance of all of my models and to perform much more hyperparameter tuning on them. As mentioned previously, due to the hardware limitations on my current machine it takes quite a long time to perform this action, and as I receive more data, more time will be required to run this code. I also need to choose a cloud platform to run my code more quickly on.
Now What? To address this increasing time needed to perform the tuning, I will try to find the best parameters before the next batch of data is sent to me, with the assumption that the same parameters will perform well with the future data. My current plan is to do this, then perform more hyperparameter turning once I receive the final batch of data. I will also experiment with using the free versions of Google Colab and AWS to run my Jupyter notebook. From my experience thus far, setting up a Google Colab instance is much quicker and requires much less knowledge than on AWS, so my current assumption is that I'll stick to Google Colab if I choose to use a cloud platform. I will likely be able to run the code efficiently with the free tier.

Student Signature	Jordan O'Donovan

Month: March

What?
This month I transferred the running of my code from my local machine to a Google Colab instance. I did this as using their free tier, I'm able to access higher computational power and I'm able to run the code with a significantly smaller chance of it crashing before completion.
I have also been working on hyperparameter tuning for my machine learning models. Thus far, I have run several different hyperparameter tuning setups with many different parameters using GridSearchCV but I've only been able to achieve an increase of 1-2% in the accuracy of my models.
I received the February data and have cleaned and pre-processed it.

So What?
Using Colab has and will be great for my project progress. I'm now able to carry out more tasks while my code runs than I was able to when I was running it on my local machine. It also means that I'll be able to train and test new models much more quickly in the future.
The slight increase in accuracy for my models is good progress but I'm trying to squeeze as much improvement as I can from the tuning. At the moment, I'm getting 80% accuracy which I'm content with, but I want to have a much higher result from this project's final model.

Now What?
To address this tuning performance, I must carry out more experimentation with parameters and different algorithms. After speaking to my supervisor, we decided that I should create graphs displaying the different results obtained from the various parameters tested using GridSearchCV. I will compare each variable to see which one(s) have the greatest impact on my model's performance, whether it be good or bad. I must also research into why those variables in particular cause the greatest effect and write about it.

Student Signature	Jordan O'Donovan

Month: April

What? This month I finalised my code (other than when new ideas come into my head as I write about my findings) and have been writing up the project file. I received March’s data and have cleaned it and rerun my models using these new tests from Spearline. I’ve continued to experiment with hyperparameter tuning and have made some small advancements in the performance of my models.
So What? My success this month were improving the performance of my models and writing up a lot of my final report. I was also able to improve my feature selection, which bumped up the accuracy of my models by another roughly three percent. What still remains is that I have to finish writing my project, edit it and complete the rest of the deliverables.
Now What? To address these, I must create my poster for the showcase and create the video presentation to be submitted. I must also tidy up and comment my code to make it much more presentable and readable. I must also force myself to stop spending so much time writing new code, as new ideas keep coming into my head while I write the final report, making it take much longer than it should to complete.

Student Signature	Jordan O'Donovan