ST1508: Practical for ai

CA2 Assignment Report

DAAA 2B02

Group 2

Jumana Haseen (2021627)

Tushita Govindaraj (2012155)

Chong Jun Hui (2021094)

Contents

[1. Background Information 1](#_Toc95513451)

[2. Data Description 2](#_Toc95513452)

[3. Data Wrangling using Advanced Techniques 3](#_Toc95513453)

[3.1. PCA Visualization Analysis 3](#_Toc95513454)

[3.1.1. Extracting PCs 3](#_Toc95513455)

[3.1.2. Visualisation **and** Exploration of Data 3](#_Toc95513456)

[3.2. Outlier Detection Algorithm 4](#_Toc95513461)

[3.3. Combatting Imbalance Classes Issue 4](#_Toc95513466)

[3.3.1 Normal Classification with Imbalanced Class 4](#_Toc95513471)

[4. Machine Learning and Experimentation 7](#_Toc95513472)

[4.1. Use of MLFlow 7](#_Toc95513473)

[4.2. Building of Baseline Models 8](#_Toc95513474)

[4.2.1. KNN 8](#_Toc95513475)

[4.2.2. Decision Tree 8](#_Toc95513476)

[4.2.3. Logistic Regression 9](#_Toc95513477)

[4.2.4. Gaussian Naïve Bayes 9](#_Toc95513478)

[4.2.5. SGD 9](#_Toc95513479)

[4.3. Optimizing the Better Models 10](#_Toc95513480)

[4.3.1. Tuning of KNN Model 10](#_Toc95513481)

[4.3.2. Tuning of Decision Tree Model 10](#_Toc95513482)

[4.4. Final Model Interpretation 10](#_Toc95513483)

[5. Deploying Prediction Models in GUI 11](#_Toc95513484)

# Background Information

Taxi has become a very important part of our lives. Without taxis it would be very hard to get to one place from another efficiently.

According to research on Evaluating Safety Issues for Taxi Transport Management, 52.5% of taxi drivers have dangerous driving behaviour and 46% of these behaviours are done repeatedly. This suggests that with increasing number of passengers these days, it is crucial for us to ensure the safety of both passengers as well as drivers.

Hence, through analysis based past data on vehicle movements and statistics, we can be successful enough to build a model which can aid us in predicting the safety of a trip through given parameters.

As part of the Data Science team of the Just Taxi company, we would be analysing taxi trip data to provide insights in the bigger picture of our company by addressing questions concerning driving safety.

# Data Description

Table

Description automatically generated

After going through phase 1 with data wrangling, feature engineering and visualization, this is the data we are left with. There are 12 columns and 7,194,164

Dataset contains information for each trip and statistical data such as accelerometer and gyroscope readings, GPS accuracy and bearing, speed as well as time of record in seconds. The labels column indicates whether a trip is considered safe or dangerous. The data description can be seen from the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Name** | **Description** | **Data Type** |
| 1 | featuresID | Identifies each record in column | Int |
| 2 | bookingID | Identifies each trip by taxi | Float |
| 3 | Accuracy | Accuracy inferred by GPS (in metres) | Float |
| 4 | Bearing | GPS bearing (in metres) | Float |
| 5 | acceleration\_x | Accelerometer reading at x axis (m/s2) | Float |
| 6 | acceleration\_y | Accelerometer reading at y axis (m/s2) | Float |
| 7 | acceleration\_z | Accelerometer reading at z axis (m/s2) | Float |
| 8 | gyro\_x | Gyroscope reading in x axis (rad/s) | Float |
| 9 | gryo\_y | Gyroscope reading in y axis (rad/s) | Float |
| 10 | gryo\_z | Gyroscope reading in z axis (rad/s) | Float |
| 11 | second | Time of the record by number of seconds | Float |
| 12 | Speed | Speed measured by GPS in m/s | Float |
| 13 | Label | 1: Dangerous trip, 0: Safe trips | Int |

# Data Wrangling using Advanced Techniques

## PCA Visualization Analysis

## Extracting PCs

For the first part of the assignment, we performed PCA to visualise the data. The image below shows a summary of our PCA results.

Text

Description automatically generated

From the summary above we can see that the total variance explained by first 5 PCs 0.5903 and the total variance explained by first 8 PCs 0.8649. According to Kaiser's rule, only the first 5 PCs should be extracted since their eigenvalues are more than 1. However, the first 5 PCs explain approximately only 59% of the total variance. Hence in this case, at least the first 8 PCs should be extracted to be able to explain approximately 86% of the total variance. The scree plot was then plotted, and a clear elbow was not to be found. Hence, we came to a decision that the highest possible number of PCs would be the most appropriate way to reduce dimension of dataset. Thus, the first 8 PCs will be extracted.

## Visualisation **and** Exploration of Data

Chart, scatter chart

Description automatically generatedWe then plotted a scatter plot for the first 2 PCs to visualise how the spread of data is. Below is an image of the scatter plot. From the scatter plot we can see that the above chart shows that both classes, safe and dangerous trips are overlapping and clustered over each other. This makes it difficult for them to be distinctively classified into two separate groups.

One reason for this could be because there are too many records with only a few features. However, if the dimension of the dataset is reduced, it may not be very accurate since the we have only 10 dimensions in this dataset. Hence, all the features are taken as of equal importance to predict whether a trip is safe or dangerous.



## Outlier Detection Algorithm

For Outlier Detection, we decided to use the isolation forest algorithm to remove outliers. The dataset was first split into training and test sets. Outliers are being removed in this data because the variability increases with high outliers which in turn causes statistical power to reduce. Outliers usually tend to be impossible values hence it is important to remove them. Therefore the isolation forest model was used to remove outliers. Logistic Regression model was used to create a baseline model. The Baseline model gave a ROC\_AUC score of 0.5429.



After obtaining the ROC\_AUC score for the baseline model, isolation forest was used to remove the outliers from the Dataset. The isolation forest model resulted in 500k data being removed as outliers and the final ROC\_AUC score was 0.6467. This value is higher than the baseline model’s AUC score.





## Combatting Imbalance Classes Issue

Chart, pie chart

Description automatically generatedBefore implementing baseline models, we need to look at the number of records for majority class and minority class. As seen from the pie chart, the 79.6% of the records are safe trips and 20.4% of the records are dangerous trips. Although the ratio of safe trips to dangerous trips is almost 4:1 which means we have a slight imbalance classification problem, looking at the total numbers of records for each class, the difference is actually very big. The actual number of records for safe and dangerous class is 5342606 and 1369549 respectively. If we train the models on the dataset without balancing the classes, the model would be biased towards the majority class and will always predict safe trips. This will result in the accuracy of the model being very high - 80% because it is always predicting the safe trips correctly but predicting the dangerous trips as safe trips. Thus, to combat imbalanced classes, we need to balance out the number of records in each class. We will be trying Random Under-Sampling, Random Over-Sampling and SMOTE and we will evaluate the performance of each approach by classifying the new dataset using logistic regression and we will use the method that gives us the best score.



### Normal Classification with Imbalanced Class

Firstly, we will do a classify the dataset without implementing any approach so that we can compare the scores before and after implementing the methods. The results show that the logistic model was out of all the predictions that were made, at least 80% of them were correct. However, looking at the recall score, the model could identify all the safe trips but the model was only able to identify 9% of the dangerous trips. This means that the model performs poorly in terms of extracting the minority samples out of the abundant samples.

Table

Description automatically generated

Thus, we will be implementing the 3 methods and evaluate the performance.

* + 1. Implementing Random Under Sampling

After implementing Random Under-Sampling, the precision scores for safe and dangerous class have decreased but the recall score for dangerous class have improved. This means that the model is not predicting all the test data as safe trips. Out of all the actual dangerous trips, the model was able to correctly label 52% of it which is a huge improvement as compared to the base model. But the model not doing well in predicting the safe trips correctly. Using Random Under-Sampling has decreased the total number of records to 2739098. We have lost 60% of the dataset. This is not good as we have lost a lot of useful information. Overall, using Random Under-Sampling is not a good method.

Table

Description automatically generated

* + 1. Implementing Random Over Sampling

After implementing Random Over-Sampling, the result of the classification is the same as Random Over-Sampling. Our dataset has increased to 10685212. Although increasing the data is better than decreasing the data, but this method randomly chooses a record belonging to the minority class and duplicates the record to balance the class distribution. this method will lead to the model overfitting.

Table

Description automatically generated

* + 1. Implementing SMOTE

After implementing SMOTE, the result of the classification is the same as Random Over-Sampling and Random Over-Sampling. Our dataset has increased to 10685212. However, instead of replicating records in the minority class, SMOTE generates synthetic data by finding the K-nearest neighbours of each minority sample, randomly selects one of them and calculate the linear interpolations. This method avoids over-fitting unlike Random Over-Sampling.

A screenshot of a computer

Description automatically generated with low confidence

* + 1. Conclusion for Random Under Sampling, Random Over Sampling and SMOTE

Random Under-sampling reduced too much of the data and random over-sampling and SMOTE generates new data but the performance of all 3 models are the same. Thus, we decided to combine Random Under-Sampling with either Random Over-Sampling or SMOTE. By combining the methods together, we can get balanced distribution by removing lesser records in the majority class, generating lesser records in the minority class which reduces overfitting.

* + 1. Implementing Random Under Sampling and Random Over Sampling

After implementing Random Under-sampling and Random over-sampling, the performance of the model has changed. The precision score for both classes are above 60% which is an average score. The recall score for class 0 is 90%. This is good as the model managed to label 90% of the actual safe trips. The recall score for class 1 is not as good as the 3 methods that we implemented earlier but it is better than the first model we implemented which had a recall score of 9%. Using this method, we have a dataset size of 7304260 which is similar to the number of records in the original dataset.

Table

Description automatically generated

* + 1. Implementing Random Under Sampling and SMOTE

After implementing Random Under-sampling and SMOTE, the performance of the model has is the same as the one above. The only difference is the precision score for class 1. The previous model had a precision score for 0.63 and the current model has a precision score of 0.62. The size of the dataset is also 7304260.

Table

Description automatically generated

* + 1. Conclusion

**Chart, pie chart

Description automatically generated**After comparing the performance of the all the methods, using Random Under-sampling and Random over-sampling gives the best results when we classify the dataset with Logistic Regression. Thus, we will use this method. As we can see, the distribution of the classes is now better than what it was at first. The percentage of the safe records is 62.5% and the percentage of the dangerous records is 37.5% The ratio is now 3:5. The final dataset size is 7304260.

# Machine Learning and Experimentation

## Use of MLFlow

For building the baseline model and hyperparameter tuning, we made use of MLFlow. MLFlow allowed us to handle the big dataset easily. Using normal jupyter notebook would have resulted:

1. A very long time to build and train baseline model.
2. A hard time tracking the ML Cycles.

With MLFlow, we were able to tackle the problem of handling a very big dataset. After the process combatting imbalanced class, we had more than 7 million data. Building a baseline model and performing hyperparameter tuning would have taken more than 10 days in normal jupyter but with the use of MLFlow, the entire process lasted close to a day only.

We were also able to use the ‘MLFlow ui’ command in the command prompt and that helped us to view and Track MLFlow Cycles. The command will result in a link that shows us all of the models we ran with MLFlow in the past. Running many models in Jupyter notebook may get very confusing as there are many codes and it would be hard to find a specific model. However, MLFlow helps us to sort the models by time and the metrics of the models will be shown in the first page which is great as we can easily find the best models instead of searching through the notebook. We can also view further details by clicking on the models and we can see many details such as the version of Python and Sklearn. We can also add description to the models for further details. We can also reuse any of the models at any time if we want to predict on another dataset. This is results in efficiency as we do not have to build, train, tune the models every time we want to do prediction.

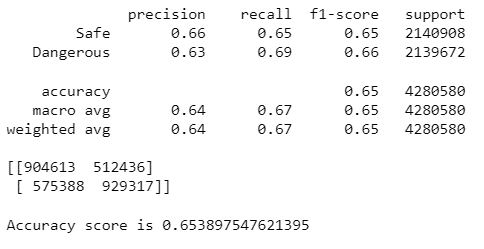
Table

Description automatically generated

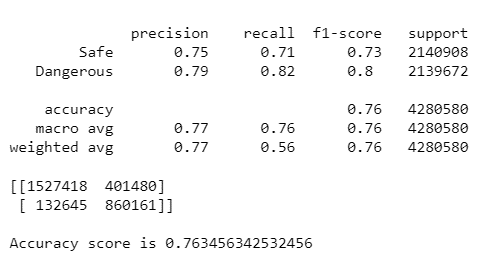
## Building of Baseline Models

For baseline models, we decided to experiment with 5 different classification models – KNN, Decision Tree, Gaussian Naïve Bayes, SGD, Logistic Regression. Hence, very basic models with default hyperparameters for each of the models were used to find the models with better accuracy scores. The better models will then be tuned further to improve its performance.

## KNN

The KNN model has an accuracy of 65%, which is only an average score for a baseline model. They are still a lot more room for improvement The precision of the model when classifying safe and dangerous trips correctly is also around the same as its accuracy, with scores 0.66 and 0.63 respectively.

## Decision Tree

The Decision tree model has an accuracy of 76% which is relatively a good score for a baseline model. The precision of the model in classifying safe and dangerous trips is also good with scores 0.75 and 0.79 respectively. With further tuning of this model, we might be able to build a model that predicts the safety of a taxi trip to high accuracy.

## Logistic Regression

Table

Description automatically generatedThe accuracy of Logistic Regression model is 59% which is not a very high score for a baseline model. The precision of the model in classifying safe and dangerous trips is also relatively low with scores 0.58 and 0.61.

## Gaussian Naïve Bayes

Table

Description automatically generatedThe accuracy of Gaussian Naïve Bayes model is 55% which is the lowest score so far for a baseline model. The precision of the model in classifying safe trips is low with a score of 0.53 but the precision of model in classifying dangerous trips is surprisingly high with a score of 0.81.

## SGD

Table

Description automatically generatedThe final model, SGD, has an accuracy of 59% which is also quite low for a baseline model. The precision of the model classifying safe trips and dangerous trips is also low with scores 0.57 and 0.65.

The best 2 models from baseline are KNN and decision tree with accuracy scores of 65% and 76% respectively. Hence, by further tuning their hyperparameters we can build the best model that will be used for predicting safe and dangerous trips to high precision and accuracy.

## Optimizing the Better Models

## Tuning of KNN Model

Graphical user interface, application, Word

Description automatically generated

GridSearchCV was used to tune the model and find the best hyperparameters for the KNN model. Then, the best parameters were used to predict using the test data. The final accuracy of the model after tuning is 78% which suggests the model’s performance improve by a lot compared to its baseline. The accuracy of this model is relatively good for prediction.

## Tuning of Decision Tree Model

Graphical user interface, application

Description automatically generated

GridSearchCV was also used here to tune the Decision tree model and find its best hyperparameters. Then, the best parameters were used to predict using the test data. The final accuracy of this model after tuning is 88% which is the highest score for a model so far. The accuracy score is also very good for predicting the safety of taxi trips to high precision and accuracy.

## Final Model Interpretation

Hence, overall, in comparison, both models perform relatively well with high accuracy after tuning. However, decision tree accuracy score is much higher than that of KNN. Furthermore, when comparing all the other metrics such as precision and f1-score, in all aspects, decision tree seems to be performing much better than KNN model. Therefore, the final and the best model chosen is decision tree with an accuracy score of 88%. The model is saved as finalized\_model.sav.

# Deploying Prediction Models in GUI

Graphical user interface, application

Description automatically generatedAfter finalising the model, we proceeded to deploy the model on Tkinter. This allows drivers to input their trip variables and they will get a prediction of either safe or dangerous trips. This comes to use when they are unsure if they are driving carefully, and the model is able to help him understand better.

To create this GUI, first, we create labels, textbox, buttons to allow the user to enter the parameters that will be used for predicting. Labels and textbox are generated for each feature and there will be 2 buttons. The first button is called “Delete” and this resets all the input in the textbox so that the user do not have to manually delete the inputs on by one. The second button is called Predict and pressing on the button will generate the parameters entered by the user and the prediction class. After the user has entered the parameters, the program will check if the parameters entered are valid. Only numbers are accepted and that includes negative positive values, floats, and integers. If the user has entered an empty input or a non-valid input, the user will be told that the parameters are invalid.

Graphical user interface, table

Description automatically generatedIf the parameters entered are valid, the parameters will be shown on the right side of the canvas. To generate the prediction, the model saved previously is loaded. The parameters will be stored in an array, and it will be reshaped into the correct shape for the model to predict. After the model is predicted, the predicted class will be shown at the right side of the canvas, below the parameters.

The layout of the canvas is created by using 2 frames, each is different columns. The labels, textbox, buttons are created in the first frame. After pressing on the predict button, the canvas will expand and the parameters and prediction will be shown in the second frame.

To exit the program, the user can click on the exit button or close the window.