Comparative Study on Car Insurance Dataset Classification Models

Jia Hui Tang[1], Ka Li Lim[1] and Yu Jie Chong[1]

[1*]Department, Faculty of Information Science and Technology, Bangi, 43600, Selangor, Malaysia.

Contributing authors: a176297@siswa.ukm.edu.my; a176496@siswa.ukm.edu.my; a176730@siswa.ukm.edu.my;

Abstract

The establishment of effective and efficient approval of insurance claims by the insurer is of significance as it defines its annual financial loss and fulfilment of its obligations as a liable insurer. Nevertheless, the growth of incoming data regards to policyholders' personal information and submission for claims resulted in slow and human-prone insurance settlements despite the approvals are made by professionals. Thus, it justifies the need for automation in insurance claims approval via Machine Learning and Deep Learning in this domain to alleviate the risk of susceptibility to the aforementioned problems. This project aims to develop a machine learning and deep learning model in approving or disapproving car insurance claims with good accuracy, ensuring the insurer nor policyholders will suffer significant financial losses. Three models were developed to participate in this study – Multilayer Perceptron (MLP), Naïve Bayes and Decision Tree. Experimental results show that MLP is the best car insurance approval classifier among the three models developed, marking its binary classification accuracy, logarithm loss, AUC-ROC score at 85.6%, 0.332, 0.8275 and they are notably higher than those metric values in Naïve Bayes and Decision Tree which are 83.55%, 5.68, 0.7961 and 82.7%, 5.9753, 0.7899 respectively. This represents that MLP has a stronger capability in distinguishing scenario in which car insurance claims should be approved or otherwise and to concisely classify whether to insure a submitted car insurance claim to its respective policyholder. Besides, MLP has the highest F1 score for the classification of approved and disapproved car insurance claims cases which are 0.90 and 0.77 respectively and they are higher than those in Naïve Bayes and Decision Tree where their F1 scores are only 0.88, 0.71 and 0.88, 0.72 respectively. This means predictability of approving and disapproving a car insurance claim is the best for MLP as compared to the other models. Lastly, MLP has lower false positive (FP) and false negative (FN) which are 132, 156 as compared to Naïve Bayes and Decision Tree which have achieved FP and FN of 135, 194 and 152, 194 respectively. MLP which has a lower FP and FP will ensure insurer to suffer minimum financial loss whilst able to fulfil its obligation as a reliable insurer. Thus, MLP is the best classification model in classifying whether a submitted car insurance claim shall be approved or otherwise.

Keywords: Multilayer Perceptron, Decision Tree, Naïve Bayes, Binary Classification

1 Introduction

A policyholder makes a formal request to an insurance company for coverage or compensation for a covered loss or policy event, which is known as an insurance claim. One of the most important and effective ways to maintain market share and profitability in a highly competitive insurance market is differentiation through the effective automation of claims management practices with minimum human intervene. Insurers can transform claims processing in particular by integrating modern claims systems with robust business intelligence, document, and content management systems. This will improve the efficiency and effectiveness of claims processing and benefit insurers both operationally and strategically by allowing them to reduce claims costs, improve claims processing efficiency, and increase customer retention and acquisition. Thus, this justifies the need for machine learning to achieve such goals.

Machine learning is being used by insurers to improve operational efficiency from claim registration to claim settlement. Many carriers have already begun to automate their claims processes, improving the customer experience while also shortening claim settlement times. Insurers can benefit from machine learning and predictive models to gain a better understanding of claim costs. These insights can help a carrier save millions of dollars in claim costs through proactive management, fast settlement, targeted investigations and better case management. Insurers can also be more confident about how much funding they allocate to claim reserves.

In this chapter, we will use a machine learning approach to classify whether the insurance claim requested by the policyholder is approved or not based on other factors. These factors are the keys to determine the validity of the insurance claim made by the policyholder. Three machine learning models such as Decision Tree, Naïve Bayes, and Multilayer Perceptron are used in this chapter to classify whether the insurance claim made by the policyholder is claimed successfully or not. These models will be trained by using the new transformed insurance dataset that had been done in the previous chapter. The transformed datasets will split into training and testing dataset for the models to learn the efficient mapping of inputs to outputs and reduce data discrepancies.

The performance of these three models is evaluated based on few evaluation metrics such as overall classification accuracy, confusion matrix, logarithmic loss, area under ROC curve (AUC) and classification report. After that, a few visualization graphs of comparison among the three models mentioned above based on each of the different calculated evaluation metrics are made in order to visualize the performance of each model. Before selecting the best classification model to determine whether the insurer should approve or deny the car insurance claim requested by the policyholders, comparisons of the performance of these three models are made. Therefore, a comparative analysis between Multilayer Perceptron, Naïve Bayes and Decision Tree algorithms is made based on the different calculated evaluation metrics and figure out which model is the best among others.

In conclusion, the main objective of this project is to figure out which machine learning model from these three models such as Multilayer Perceptron, Naïve Bayes and Decision Tree is the best choice to determine whether the insurance claim requested by the policyholders should or should not be approved.

2 Related Work

As the demand for insurance grows, many companies are beginning to rely on the implementation of Artificial Intelligence, such as the machine learning model, to gain insights into issues such as risk management or fraudulent claims. In this section, various research that have been conducted throughout the years will be review and discussed to find out how predictive analytics should be implemented in order to achieve the best results. In 2015, Hanafy and Ruixing emphasized on how ML models can be applied onto the insurance big data, and also utilized six different ML methods known as Logistic Regression, XGBoost, Random Forest, Decision Trees, Naïve Bayes, and K-NN to forecast the occurrence of insurance claims.[1] Their findings show that Random Forest has the most promising result as it was able to correctly classify the input data with an accuracy of 86.77%, followed by Decision Tree (C50) - 70.67% and others with Naïve Bayes the worst at 60.56%. The result is justified by the fact that the dataset used is extremely large, and Random Forest outperforms all other algorithms under these conditions. Other than that, "A Proposed Model to Predict Auto Insurance Claims using Machine Learning Techniques" by Shady, Khaled, and Abdelsalam in 2020 also discussed about the implementation of ML models for the prediction of car insurance claims.[2] In their studies, they have compared the performance of Artificial Neural Network (ANN), Decision Tree, Naïve Bayes and XGBoost, and based on the result, it is shown that the XGBoost (92.53%) model have managed to prevail over the other models while Naïve Bayes have the lowest accuracy at 83.84%.

Furthermore, Weerasinghe and Wijegunasekara completed a study in 2016 to investigate the data mining technique used in the development of a predictive model for auto insurance claim prediction.[3] During the research, the prediction model was developed using Artificial Neural Network (ANN), Decision Tree and Multinomial Logistic Regression (MLR) and among these 3 models, the results showed that ANN (61.71%) is the best predictor while MLR (52.39%) is the worst. In addition, a study was conducted in 2022 by Sebastian Baran on the prediction of motor insurance claims occurrence as an imbalanced machine learning problem.[4] For this study, he have implemented techniques such as Logistic Regression, Decision Tree, Random Forest, XGBoost and feed forward neural network and based on the results obtained, it is demonstrated that Random Forest has the highest accuracy at 84.98%, followed by XGBoost (84.22%), Decision Tree (81.24%), NN (78.28%) and Logistic Regression at 52.18%. In 2018, Jing, Zhao, Karthink and Feng have compared two different model, Naïve Bayes and Bayesian Network for their research on the prediction of car insurance claim and as a result, both models have reached an accuracy of 93.16%.[5] However, since Bayesian Network managed to

obtain the same accuracy with Naïve Bayes with fewer variables, so the former is still more effective than the later. Apart from that, there has also been an attempt made by Rama, Ram, Ramesh and Srinivasa to use ML algorithms such as Naïve Bayes, Naïve Bayes Updatable, Multi-Layer Perceptron, Decision Tree, Random Tree, LMT and Random Forest to analysis insurance claim.[6] As a result of their comparison, it seems that LMT (100%) and Random Forest (100%) have shown more promising result when compared to the rest as proven by the precision and recall value.

Overall, it appears that while most of the studies above uses dataset that is larger in size, there are few ML model such as Random Forest, Decision Tree, and XGBoost that managed to stand out above the rest, others such as Naïve Bayes, may not be as efficient. Therefore, in this study, we will be focusing on developing ML models, specifically Decision Tree, Naïve Bayes and Neural Network and then carry out a comparative analysis with the help of evaluation metrics such as classification accuracy, AUC, confusion matrix and others to determine which method will be the most suitable for predicting the approval of insurance claims with the car insurance dataset.

3 Classification Methods

Three classification models are proposed to classify the approval of car insurance, namely Multilayer Perceptron (MLP), Decision Tree, and Naïve Bayes. The general characteristic of these models is they are a supervised algorithm and suitable to be used for binary classification problem, such as the approval of car insurance dataset problem.

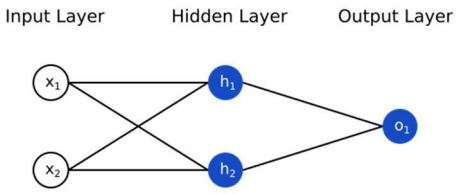


Figure 1: Network Topology of MLP

MLP is a supervised, feed forward neural network with the capability in dealing with datasets which are linearly inseparable and its novelty is based on how the human biological brain and nervous system works on information processing, decision making and solving highly complex tasks. Its building block are artificial neurons that have weighted input signals that produces an output signal where its strength of influence is determined by an activation or transfer function. These neurons will then be aligned into layers to form the complete architecture of the MLP which has an Input Layer, Hidden Layer(s) and an Output Layer (Refer Figure 1). The number of neurons in the input layer represents the total number of attributes / independent variables that are a dataset whilst the number of neurons in the output

layer depends on the classification problem, whether it is a binary or multiclass classification problem. All neurons are responsible in learning the data through the dot product computation of its weights and inputs. Upon reaching the output layer, the classification decision is made and loss function is used to compute the error made by the model before backpropagation algorithm is employed, whereby weights and internal states of the neurons are updated to enhance the performance of its succeeding classification.

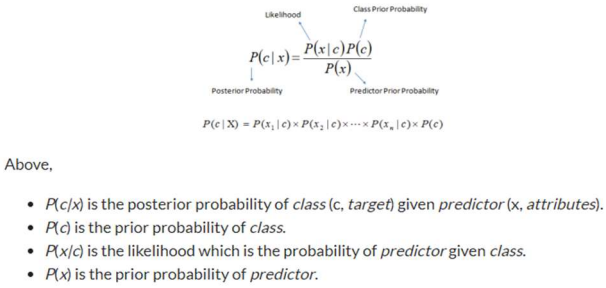


Figure 2: Equation of Bayes Theorem

The Naïve Bayes algorithm is a classification method that uses the Bayes Theorem (Refer Figure 2) and the assumption of predictor independence. A Naive Bayes classifier, to put it simply, believes that the presence of one feature in a class has nothing to do with the presence of any other features. Naive Bayes classifier learns each feature's conditional probability on the class label C from training data. The Bayes rule is used to determine the classification's probability of C for the feature instance xi, ..., xn. The class with the highest probability is the one that will be predicted. Given the value of class C, the classifier operates under the presumption that all attributes xi are conditionally independent. Because all attributes are processed independently, the performance can be surprising. This is because it ignores potentially significant correlations between features.

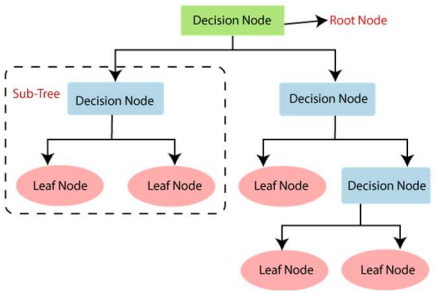


Figure 3: Basic Structure of Decision Tree

Decision Tree is a supervised learning algorithms that can be used for both classification and regression problems. It involves utilizing a set of rules by constructing a tree-structured classifier which contain several nodes and multiple

branches to represent the decision-making rules. In particular, the two primary nodes, Decision and Leaf node plays a crucial role where the former is employed in the decision-making process and consist of various branches while the latter represents the outcome of the decision that is made from the Decision node. Having stated that, the primary objective of adopting a Decision Tree is to develop a training model that can be used to forecast the class or value of the target variable by learning simple decision rules that is derived from the training data inputted.

4 Results and Discussion

Experimental results, specifically the classification accuracy, Logarithmic Loss, and AUC-ROC score will be presented individually according to each of the three models. Besides, the ROC curve, Precision-Recall curve, confusion matrix and classification report are visualized for each model to have an intelligible understanding of the results obtained.

4.1 Experimental Result Presentation

4.1.1 Multilayer Perceptron

Table 1 reports on the classification accuracy, logarithmic loss and ROC-AUC score obtained from the binary classification of MLP algorithm. Accuracy represents the total number of correct predictions made by MLP; Logarithmic loss indicative of how close the prediction probability is to the corresponding actual value; ROC-AUC score is equivalent calculating the rank correlation between predictions and actual value.

Table 1: Classification Accuracy, Log Loss and ROC-AUC score of MLP

Classification Accuracy (%)	85.60
Logarithmic Loss	0.3320
ROC-AUC Score	0.8275

Table 2 depicts the confusion matrix which consists True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN) on MLP's prediction.

Table 2: Confusion Matrix of MLP

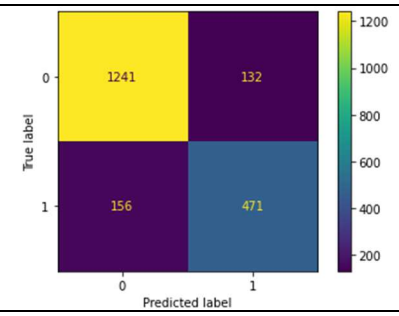
Confusion Matrix	

Table 3 depicts the classification report that provides information on the precision, recall and F1 score on the classification made by MLP according to the class labels – approved and disapproved.

Table 3: Classification Report of MLP

Classification Report		precision	recall	f1-score	support
	dissapproved	0.89	0.90	0.90	1373
	approved	0.78	0.75	0.77	627
	accuracy			0.86	2000
	macro avg	0.83	0.83	0.83	2000
	weighted avg	0.85	0.86	0.86	2000

4.1.2 Decision Tree

Table 4 reports on the classification accuracy, logarithmic loss and ROC-AUC score obtained from the binary classification of Decision Tree algorithm. Accuracy represents the total number of correct predictions made by Decision tree; Logarithmic loss indicative of how close the prediction probability is to the corresponding actual value; ROC-AUC score is equivalent calculating the rank correlation between predictions and actual value.

Table 4: Classification Accuracy, Log Loss and ROC-AUC score of Decision Tree

Classification Accuracy (%)	82.70
Logarithmic Loss	5.9753
ROC-AUC Score	0.7899

Table 5 depicts the confusion matrix which consists True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN) on Decision Tree's prediction.

Table 5: Confusion Matrix of Decision Tree

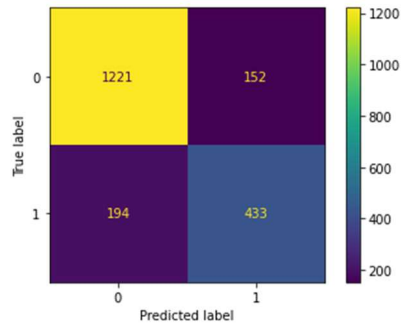
Confusion Matrix	

Table 6 depicts the classification report that provides information on the precision, recall and F1 score on the classification made by Decision Tree according to the class labels – approved and disapproved.

Table 6: Classification Report of Decision Tree

Classification Report		precision	recall	f1-score	support
	dissapproved	0.86	0.89	0.88	1373
	approved	0.74	0.69	0.71	627
	accuracy			0.83	2000
	macro avg	0.80	0.79	0.80	2000
	weighted avg	0.82	0.83	0.83	2000

4.1.3 Naïve Bayes

Table 7 reports on the classification accuracy, logarithmic loss and ROC-AUC score obtained from the binary classification of Naïve Bayes algorithm. Accuracy represents the total number of correct predictions made by Naïve Bayes; Logarithmic loss indicative of how close the prediction probability is to the corresponding actual value; ROC-AUC score is equivalent calculating the rank correlation between predictions and actual value.

Table 7: Classification Accuracy, Log Loss and ROC-AUC score of Naïve Bayes

Classification Accuracy (%)	83.55
Logarithmic Loss	5.6817
ROC-AUC Score	0.7961

Table 8 depicts the confusion matrix which consists True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN) on Naïve Bayes' prediction.

Table 8: Confusion Matrix of Naïve Bayes

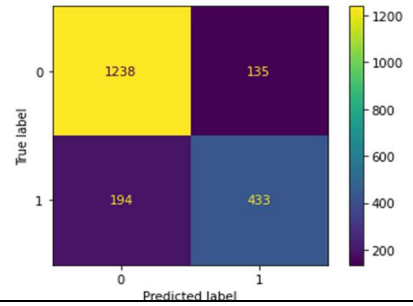
Confusion Matrix	

Table 9 depicts the classification report that provides information on the precision, recall and F1 score on the classification made by Naïve Bayes according to the class labels – approved and disapproved.

Table 9: Classification Report of Naïve Bayes

Classification Report		precision	recall	f1-score	support
	dissapproved	0.86	0.90	0.88	1373
	approved	0.76	0.69	0.72	627
	accuracy			0.84	2000
	macro avg	0.81	0.80	0.80	2000
	weighted avg	0.83	0.84	0.83	2000

4.2 Comparative Analysis Results

The classification accuracy, logarithm loss and ROC-AUC score are compared and reported as in Table 10.

Table 10: Comparative Analysis of Classification Methods Performance

Method	**Accuracy (%)**	**Logarithm Loss**	**ROC-AUC**
Multilayer Perceptron	85.60	0.3320	0.8275
Decision Tree	82.70	5.9753	0.7899
Naïve Bayes	83.55	5.6817	0.7961

4.3 Graph Visualization

Figure 4: Comparison of classification accuracy between Multilayer Perceptron, Decision Tree and Naïve Bayes

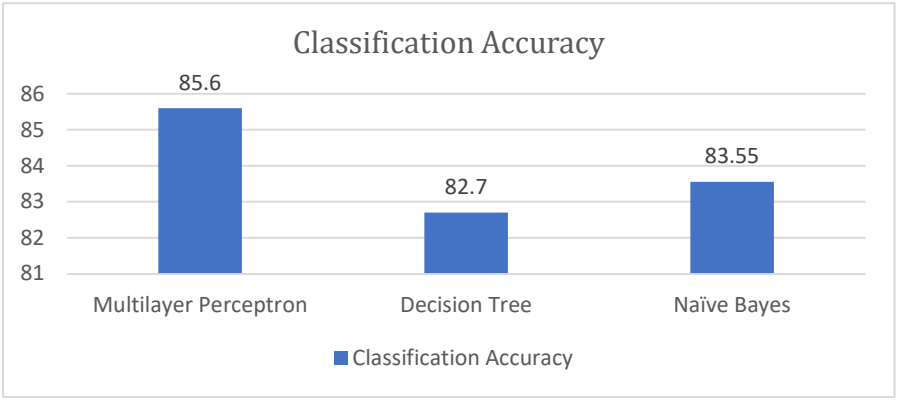


Figure 5: Comparison of logarithm loss between Multilayer Perceptron, Decision Tree and Naïve Bayes

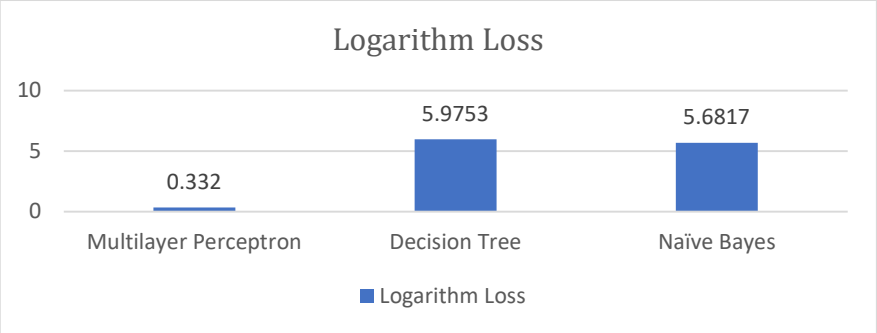


Figure 6: Comparison of ROC-AUC score between Multilayer Perceptron, Decision Tree and Naïve Bayes

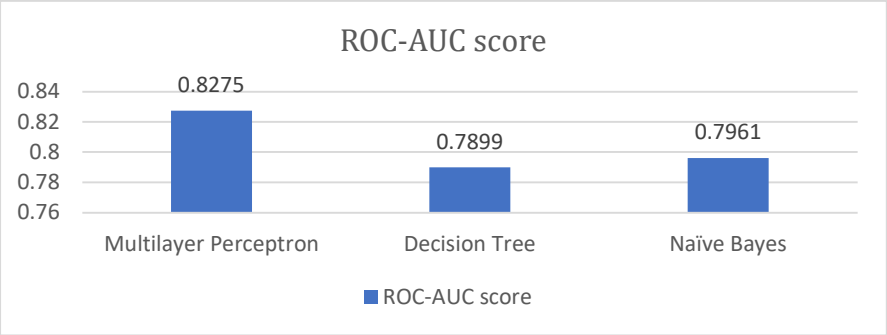
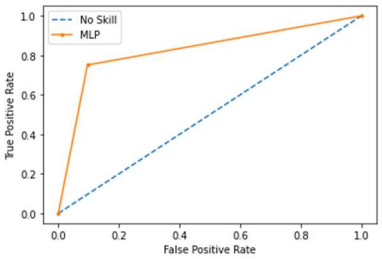
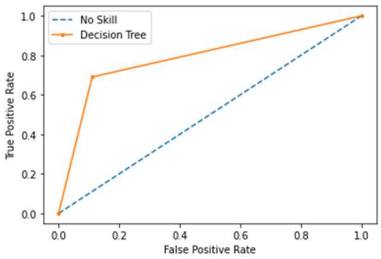


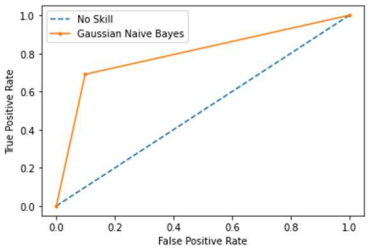
Figure 7: ROC Curve on classification made by (a)Multilayer Perceptron, (b) Decision Tree and (c) Naïve Bayes



(a)

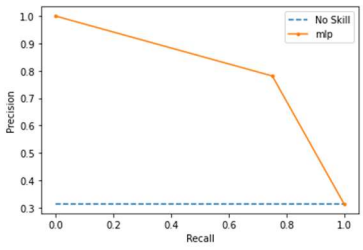


(b)

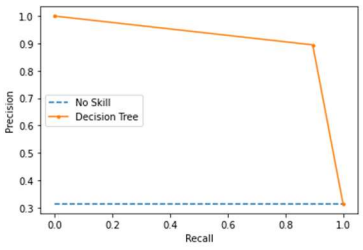


(c)

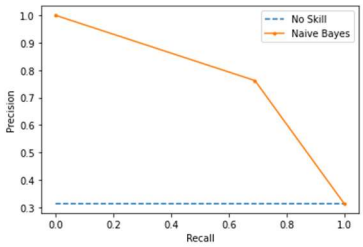
Figure 8: Precision-Recall Curve on classification made by (a)Multilayer Perceptron, (b) Decision Tree and (c) Naïve Bayes



(a)



(b)



(c)

4.4 Result Discussion

Comparisons on the performance are made between MLP, Decision Tree and Naïve Bayes models before ruling out the best classification model in determining should the insurer approve or disapproved the car insurance claim submitted by policyholders. Their overall performance is defined by the metrics obtained, which are classification accuracy, confusion matrix, F1 score, ROC-AUC value, and log loss.

Based on the aforementioned metrics obtained, MLP is the best classifier among the three classification models. The first reason is MLP achieved the highest classification accuracy, marking at 85.60 % which is 2.90% and 2.05% higher than the classification accuracy achieved by Decision Tree and Naïve Bayes respectively. This indicates MLP is able to correctly classify and assist the insurer in deciding if a submitted claim of car insurance should be approved or otherwise better than the other two models.

The second justification of which MLP is a better classification model in classifying car insurance claims is due to the smaller FP value as depicted in the confusion matrix, which is 132 as compared to 152 by Decision Tree and 135 by Naïve Bayes. FP in this case study represents the cases in which a car insurance claim submitted should be disapproved by the insurer but was wrongly classified as approved. A high FP value represents that the insurer is prone to financial losses as claims were approved to ineligible policyholders. Therefore, MLP which has the lowest FP value among the three models is the best candidate in classifying the approval of car insurance claims. The values of true positive (TP), false negative (FN)

and true negative (TN) in the confusion matrix are of secondary importance as it has lower influence on the insurer financial losses. However, FN value would be critical if the insurer priority and obligation is to ensure all eligible policyholders must be insured and that lower FN indicates that it is less likely a qualified policyholders will not be insured. This further emphasize that MLP is a better classification model again as it has the lowest FN (156) while Decision Tree and Naïve Bayes share the same FN value of 194. As for TP and TN that represents cases in which a submitted claim of car insurance was correctly classified into its correct respected categories, MLP has the highest values for both TP and TN, flagging at 1241 and 471 respectively while 1221 and 433 for Decision Tree and 1238 and 433 for Naïve Bayes.

F1 score as the hormonic mean of precision and recall is used as a metric in considering the overall performance of the classification models as it has higher robustness towards the problem of class imbalance as experienced in the Car Insurance Dataset used, whereby the number of disapproved cases is twice as much to the number of approved cases. Hence, its value considered the alleviation on the domination of one class in the dataset that mislead the performance of the classification model. As higher F1 score indicates a better predictive performance, this upholds that MLP is a better classification model in classifying the approval and disapproval of car insurance claims and that the F1 score for the mentioned cases are 0.90 and 0.77 respectively. In contrast, it is 0.88 and 0.71 for Decision Tree while 0.88 and 0.72 for Naïve Bayes.

ROC-AUC score is used to distinguish and to discriminate the ability to classify if a submitted claim of car insurance should be approved or disapproved. A model with high ROC-AUC score indicates high distinguishability and classification performance, which is also seen in MLP with score of 0.8275 while Decision Tree and Naïve Bayes have ROC-AUC score of 0.7899 and 0.7960 respectively. Therefore, this rationalizes that MLP is a better model than the rest of the two models.

Log loss or known as cross entropy loss indicates the difference between the predicted probability to the ground truth value and a lower log loss means a higher confidence in the classification or prediction made by the model upon penalizing wrongly classified cases. Thus, MLP would make a better classification model in classifying if a submitted car insurance should be approved or otherwise due to its lower log loss, which is 0.3320 as compared to 5.975 and 5.682, which are the log loss for Decision Tree and Naïve Bayes respectively. This means the prediction made by MLP has a higher confidence as compared to the remaining two models.

Hence, as MLP has the highest binary classification accuracy, AUC-ROC score, F1 scores, and the lowest values for log loss and FP and FN, it makes MLP the best classification model in classifying car insurance approval prediction. The next best classification model after MLP is Naïve Bayes, then Decision Tree.

5 Conclusion

Machine learning plays an important role in the insurance industry as it can providing valuable insights that are crucial to the decision-making process regarding whether

an insurance claim submitted by the policyholder should be approved or not. By having the capability to correctly classify the outcome of an insurance claim accurately, the insurer or insurance company will be able to provide top-notch services to their client as they have a reliable decision-making tool. In short, machine learning algorithm have been utilized in this study to extract behaviour or trends that can be decisive and helpful in predicting the outcome for the status of insurance claim.

During this project, an attempt has been done to implement several different ML model namely, Multilayer Perceptron (MLP), Decision Tree and Naïve Bayes onto the cleaned dataset that is obtained after performing data pre-processing techniques on its raw data. Thus, after each model have been successfully developed, a comparative study between the models is done with the help of several evaluation metrics in order to find out which classifier is capable of producing the best performance with the given dataset. As a result, it is clear that MLP has managed to prevail over the other models as it was able to obtain an accuracy score of 85.6% which was the highest among the rest. Aside from that, this result is also justified by other evaluation metrics such as the logarithm loss (0.3320) and ROC-AUC score (0.8275) where it has been shown to outperform the other two models.

Throughout the whole project, some of the limitation that is faced is the lack of insurance claim dataset when compared to other related studies. Since most of the studies uses dataset that consists of triple to hundreds of times the amount of the cases than the ones we used, so in comparison, this might mean that our model may not be performing at its top condition since it should be receiving more data. Although more data is generally accompanied with the increase in accuracy, however, it is also important to know that the main factor here is not just more data but rather more quality data because if the data itself contain too much noise, then it will not be helpful regardless of the data volume since the model might overfit and thus produce a not so accurate prediction. Therefore, the suggestion that can be done for future improvements includes using a bigger dataset and considering combining several different base models in order to produce one optimal predictive model such as XGBoost since it is proven in some related research that ensemble learning is able to produce better result.

6 Acknowledgement

The authors would like to thank Dr. Azuraliza Abu Bakar for her advice, patience and guidance throughout the work of this research. The code implemented can be found in

Neural Network: https://colab.research.google.com/drive/1x7C0wu_iYHZ1K0-3tiORMHLrTJcEUtPA?usp=sharing

Naïve Bayes:
https://colab.research.google.com/drive/1UuQQ4mguPOthZMPgMETKlezHUi_dxv1l?usp=sharing

Decision Tree:
https://colab.research.google.com/drive/1DwgXoE8klKjsJQMGWzzqrH6T0YkdkVTu?usp=sharing

Tang Jia Hui is a computer science undergraduate from the National University of Malaysia (UKM), specializes in data science studies. Her contribution to the research focuses on abstract, all aspects related to neural network of this study of car insurance approval prediction and the result discussion section.
Contact: a176297@siswa.ukm.edu.my

Lim Ka Li is a computer science undergraduate from the National University of Malaysia (UKM), specializes in data science studies. Her contribution to the research focuses on the literature review, all aspects related to decision tree of this study of car insurance approval prediction and conclusion.
Contact: a176496@siswa.ukm.edu.my

Chong Yu Jie is a computer science undergraduate from the National University of Malaysia (UKM), specializes in data science studies. His contribution to the research focuses on the introduction and all aspects related to naïve bayes of this study of car insurance approval prediction.
Contact: a176730@siswa.ukm.edu.my

7 References

[1] Hanafy, Mohamed, and Ruixing Ming. 2021. Machine Learning Approaches for Auto Insurance Big Data. Risks 9: 42. https://doi.org/10.3390/risks9020042

[2] Abdelhadi, S., ElBahnasy, K.A., & Abdelsalam, M.M. (2020). A PROPOSED MODEL TO PREDICT AUTO INSURANCE CLAIMS USING MACHINE LEARNING TECHNIQUES.

[3] Weerasinghe, K.M., & Wijegunasekara, M.C. (2016). A Comparative Study of Data Mining Algorithms in the Prediction of Auto Insurance Claims.

[4] Baran, Sebastian & Rola, Przemysław. (2022). Prediction of motor insurance claims occurrence as an imbalanced machine learning problem.

[5] Jing, Longhao & Zhao, Wenjing & Sharma, Karthik & Feng, Runhua. (2018). Research on Probability-based Learning Application on Car Insurance Data. 10.2991/macmc-17.2018.14.

[6] Rama Devi Burri, Ram Burri, Ramesh Reddy Bojja, Srinivasa Rao Buruga. (2019). Insurance Claim Analysis Using Machine Learning Algorithms. International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN-2278-3075, 8 (6S4).

Appendix

Appendix A: Python code and Output of Multilayer Perceptron (MLP) classification model

Appendix A1: Install library using pip

```
!pip install tensorflow-model-analysis
!pip install tensorflow-addons
```

Appendix A2: Import library

```
import tensorflow as tf
import tensorflow_model_analysis as tfma
import tensorflow_addons as tfa
from tensorflow.keras import layers
from sklearn.metrics import classification_report
```

Appendix A3: Load cleaned dataset

```
import pandas as pd
df = pd.read_csv('/content/cleaned_car_insurance_dataset.csv', index_col='Unnamed: 0')
df.head()
```

	AGE	GENDER	RACE	DRIVING_EXPERIENCE	INCOME	VEHICLE_YEAR	POSTAL_CODE	VEHICLE_OWNERSHIP	MARRIED	CHILDREN	CREDIT_SCORE	ANNUAL_MILEAGE	PAST_ACCIDENTS	OUTCOME
0	0.953343	-0.622033	-0.354414	-0.898878	-0.384870	1.594129	0.184547	0.223370	0.500337	-0.701112	0.532798	0.078642	-0.136456	0.0
1	-2.537095	-0.802031	0.146777	-0.147050	0.303880	-0.340219	0.361956	-0.381019	0.240873	0.248644	0.022340	-0.273917	-0.108320	1.0
2	-1.385940	-0.290501	-1.482303	-1.232866	-0.422788	-0.479020	-0.578699	0.117883	-0.391011	-0.245258	0.527691	-0.291033	-0.089770	0.0
3	-2.382989	1.825815	-1.578011	0.208343	0.042549	-0.239941	0.408163	0.232866	-0.369905	-0.603366	-0.337437	0.258680	-0.116753	0.0
4	-0.886031	1.549266	-1.416426	0.222215	0.687438	-0.327309	0.337817	0.076373	-0.445377	-0.195532	0.264539	-0.556849	-0.091635	1.0

Appendix A4: Data Splitting using Stratified Shuffle Split with the split ratio of 80:20 where 80% of the dataset is the train set while 20% is the test set.

```
X = df.iloc[:,:13]
Y = df.iloc[:,-1]

from sklearn.model_selection import StratifiedShuffleSplit
sss = StratifiedShuffleSplit(n_splits=30, test_size=0.2, random_state=42)
sss.get_n_splits(X, Y)

for train_index, test_index in sss.split(X, Y):
  X_train, X_test = X.iloc[train_index], X.iloc[test_index]
  Y_train, Y_test = Y.iloc[train_index], Y.iloc[test_index]

print(X_train.shape, Y_train.shape)
print(X_test.shape, Y_test.shape)
```

```
(8000, 13) (8000,)
(2000, 13) (2000,)
```

Appendix A5: MLP Model Training

```
predictive_model = tf.keras.Sequential(layers=[
                                        layers.Dense(13, input_shape=(13,), activation='relu'),
                                        layers.Dense(8),
                                        layers.Dense(4),
                                        layers.Dense(1, name='classifier', activation='sigmoid', bias_initializer=None),
], name='CarInsurancePredictiveModel'
)
METRICS = [tf.keras.metrics.BinaryAccuracy(name='acc'),
           tf.keras.metrics.TruePositives(name='tp'),
           tf.keras.metrics.FalsePositives(name='fp'),
           tf.keras.metrics.TrueNegatives(name='tn'),
           tf.keras.metrics.FalseNegatives(name='fn'),
           tf.keras.metrics.Precision(name='precision'),
           tf.keras.metrics.Recall(name='recall'),
           tf.keras.metrics.AUC(name='auc'),
           tf.keras.metrics.AUC(name='prc', curve='PR')]

predictive_model.compile(optimizer=tfa.optimizers.AdamW(learning_rate=1e-3, weight_decay=1e-4),
                         loss=tf.keras.losses.BinaryCrossentropy(),
                         metrics=METRICS)

reducelr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5, verbose=0, mode='min')
earlystop = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=15, mode='min', restore_best_weights=True)
history = predictive_model.fit(X_train, Y_train, validation_split=0.20, shuffle=True, epochs=200, batch_size=64, callbacks=[reducelr, earlystop])
```

```
Epoch 1/200
100/100 [==============================] - 9s 34ms/step - loss: 0.4674 - acc: 0.7700 - tp: 974.0000 - fp: 458.0000 - tn: 3954.0000 - fn: 1014.0000 - precision: 0.6802 - recall: 0.4899 - auc: 0.8360 - prc: 0.6636 - val_loss: 0.4358
Epoch 2/200
100/100 [==============================] - 1s 7ms/step - loss: 0.4010 - acc: 0.8136 - tp: 1274.0000 - fp: 479.0000 - tn: 3933.0000 - fn: 714.0000 - precision: 0.7268 - recall: 0.6408 - auc: 0.8802 - prc: 0.7426 - val_loss: 0.4103
Epoch 3/200
100/100 [==============================] - 1s 7ms/step - loss: 0.3781 - acc: 0.8281 - tp: 1364.0000 - fp: 476.0000 - tn: 3936.0000 - fn: 624.0000 - precision: 0.7413 - recall: 0.6861 - auc: 0.8945 - prc: 0.7730 - val_loss: 0.3984
Epoch 4/200
100/100 [==============================] - 1s 8ms/step - loss: 0.3654 - acc: 0.8361 - tp: 1383.0000 - fp: 444.0000 - tn: 3968.0000 - fn: 605.0000 - precision: 0.7570 - recall: 0.6957 - auc: 0.9018 - prc: 0.7896 - val_loss: 0.3902
```

Appendix A6: MLP Model Evaluation

```
for index, val in enumerate(predictive_model.evaluate(X_test, Y_test)):
  if index==0:
    print("loss :"+str(val))
  else:
    print(METRICS[index-1].name +":"+str(val))
```

```
63/63 [==============================] - 1s 3ms/step - loss: 0.3320 - acc: 0.856
loss :0.3320310711860657
acc:0.8560000061988831
tp:471.0
fp:132.0
tn:1241.0
fn:156.0
precision:0.7810945510864258
recall:0.7511961460113525
auc:0.9191678762435913
prc:0.8422930836677551
```

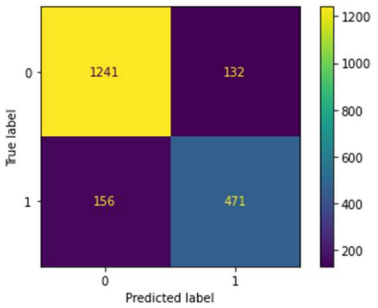
Appendix A7: MLP Classification Report

```
from sklearn.metrics import classification_report
import numpy as np

def prediction(X_test):
  a = []
  for predicted_val in predictive_model.predict(X_test):
    if predicted_val>0.5:
      a.append(1)
    else:
      a.append(0)
  return a

y_true = Y_test
y_pred = np.where(predictive_model.predict(X_test) > 0.5, 1, 0)
print(classification_report(y_true, y_pred, target_names=['dissapproved', 'approved']))
```

```
              precision    recall  f1-score   support

dissapproved       0.89      0.90      0.90      1373
    approved       0.78      0.75      0.77       627

    accuracy                           0.86      2000
   macro avg       0.83      0.83      0.83      2000
weighted avg       0.85      0.86      0.86      2000
```

Appendix A8: MLP Confusion Matrix

```
%matplotlib inline
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
ConfusionMatrixDisplay(confusion_matrix=confusion_matrix(y_true, y_pred)).plot()
plt.show()
```



Appendix A9: MLP ROC-AUC Score

```
from sklearn.metrics import roc_auc_score, roc_curve
roc_auc_score(Y_test, np.where(predictive_model.predict(X_test) > 0.5, 1, 0))
```

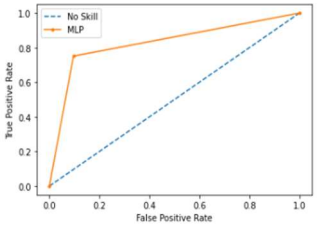
```
0.8275281662409351
```

Appendix A10: Plot ROC Curve

```
# ROC AUC GRAPH
no_skill_prediction_prob = [0 for _ in range(len(Y_test))]
predicted_prob = np.where(predictive_model.predict(X_test) > 0.5, 1, 0)
no_skill_prediction_auc = roc_auc_score(Y_test, no_skill_prediction_prob)
predicted_prob_auc = roc_auc_score(Y_test, np.where(predictive_model.predict(X_test) > 0.5, 1, 0))

# Summarize score
print('No Skill: ROC AUC=%.3f' % (no_skill_prediction_auc))
print('Logistic: ROC AUC=%.3f' % (predicted_prob_auc))
```

```
No Skill: ROC AUC=0.500
Logistic: ROC AUC=0.828
```

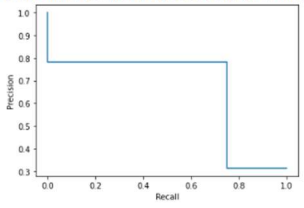
```
# calculate roc curves
ns_fpr, ns_tpr, _ = roc_curve(Y_test, no_skill_prediction_prob)
lr_fpr, lr_tpr, _ = roc_curve(Y_test, np.where(predictive_model.predict(X_test) > 0.5, 1, 0))
# plot the roc curve for the model
plt.plot(ns_fpr, ns_tpr, linestyle='--', label='No Skill')
plt.plot(lr_fpr, lr_tpr, marker='.', label='MLP')
# axis labels
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# show the legend
plt.legend()
# show the plot
plt.show()
```



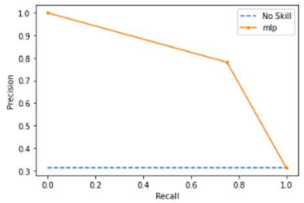
Appendix A11: Plot MLP Precision-Recall Curve

```
from sklearn.metrics import precision_recall_curve
precision, recall, threshold = precision_recall_curve(Y_test, np.where(predictive_model.predict(X_test) > 0.5, 1, 0))
PrecisionRecallDisplay(precision=precision, recall=recall).plot()
```

```
<sklearn.metrics._plot.precision_recall_curve.PrecisionRecallDisplay at 0x7f8e1b010450>
```




```
no_skill = len(Y_test[Y_test==1]) / len(Y_test)
plt.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill')
plt.plot(recall, precision, marker='.', label='mlp')
# axis labels
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend()
plt.show()
```



Appendix B: Python code and Output of Naïve Bayes classification model

Appendix B1: Import library

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedShuffleSplit

from sklearn.naive_bayes import GaussianNB

from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix, accuracy_score, roc_auc_score, roc_curve, auc, precision_recall_curve, average_precision_score
from sklearn.metrics import log_loss
```

Appendix B2: Load cleaned dataset

```
df = pd.read_csv('latest_dataset.csv')
df.head()
```

	AGE	GENDER	RACE	DRIVING_EXPERIENCE	INCOME	VEHICLE_YEAR	POSTAL_CODE	VEHICLE_OWNERSHIP	MARRIED	CHILDREN	CREDIT_SCORE	ANNUAL_MILEAGE	PAST_ACCIDENTS	OUTCOME
0	0.953343	-0.622033	-0.354414	-0.898978	-0.384870	1.594129	0.184547	0.223370	0.500337	-0.701112	0.532798	0.078642	-0.136456	0.0
1	-2.537095	-0.802031	0.146777	-0.147050	0.303880	-0.340219	0.361956	-0.381019	0.240873	0.248644	0.022340	-0.273917	-0.108320	1.0
2	-1.385940	-0.290501	-1.482303	-1.232866	-0.422788	-0.479020	-0.578699	0.117883	-0.391011	-0.245258	0.527691	-0.291033	-0.089770	0.0
3	-2.382989	1.825815	-1.578011	0.208343	0.042549	-0.239941	0.408163	0.232866	-0.369905	-0.603366	-0.337437	0.258680	-0.116753	0.0
4	-0.886031	1.549266	-1.416426	0.222215	0.687438	-0.327309	0.337817	0.076373	-0.445377	-0.195532	0.264539	-0.558849	-0.091635	1.0

Appendix B3: Data Splitting using Stratified Shuffle Split with the split ratio of 80:20 where 80% of the dataset is the train set while 20% is the test set

```
sss = StratifiedShuffleSplit(n_splits=30, test_size=0.2, random_state=42)
sss.get_n_splits(X, y)

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

Appendix B4: Naïve Bayes model training

```
nb = GaussianNB()
nb.fit(X_train,y_train)
nbpred = nb.predict(X_test)
```

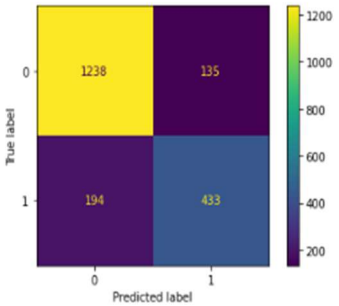
Appendix B5: Naïve Bayes Classification Report

```
print(classification_report(y_test,nbpred, target_names = ['dissapproved','approved']))
```

	precision	recall	f1-score	support
dissapproved	0.86	0.90	0.88	1373
approved	0.76	0.69	0.72	627
accuracy			0.84	2000
macro avg	0.81	0.80	0.80	2000
weighted avg	0.83	0.84	0.83	2000

Appendix B6: Naïve Bayes Confusion Matrix

```
from sklearn.metrics import ConfusionMatrixDisplay

ConfusionMatrixDisplay(confusion_matrix=confusion_matrix(y_test, nbpred)).plot()
plt.show()
```



Appendix B7: Naïve Bayes ROC-AUC score

```
nb_auc = roc_auc_score(y_test, nbpred)
nb_auc
```

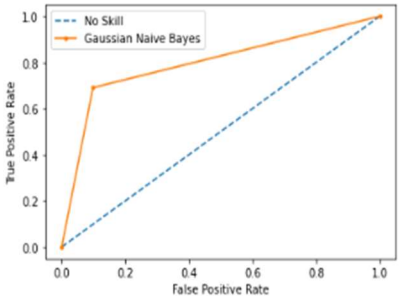
```
0.7961326377587351
```

Appendix B8: Plot Naïve Bayes ROC Curve

```
# ROC AUC GRAPH
no_skill_prediction_prob = [0 for _ in range(len(y_test))]
predicted_prob = np.where(nb.predict(X_test) > 0.5, 1, 0)
no_skill_prediction_auc = roc_auc_score(y_test, no_skill_prediction_prob)
predicted_prob_auc = roc_auc_score(y_test, np.where(nb.predict(X_test) > 0.5, 1, 0))

# Summarize score
print('No Skill: ROC AUC=%.3f' % (no_skill_prediction_auc))
print('Naive Bayes: ROC AUC=%.3f' % (predicted_prob_auc))
```

```
No Skill: ROC AUC=0.500
Naive Bayes: ROC AUC=0.796
```

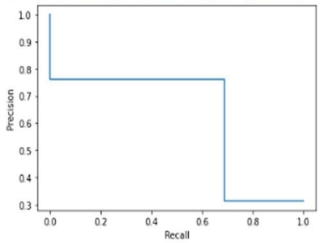
```
# calculate roc curves
ns_fpr, ns_tpr, _ = roc_curve(y_test, no_skill_prediction_prob)
lr_fpr, lr_tpr, _ = roc_curve(y_test, np.where(nb.predict(X_test) > 0.5, 1, 0))
# plot the roc curve for the model
plt.plot(ns_fpr, ns_tpr, linestyle='--', label='No Skill')
plt.plot(lr_fpr, lr_tpr, marker='.', label='Gaussian Naive Bayes')
# axis labels
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# show the legend
plt.legend()
# show the plot
plt.show()
```



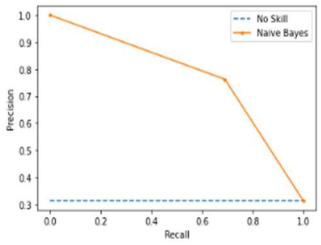
Appendix B9: Plot Naïve Bayes Precision-Recall Curve

```
precision, recall, threshold = precision_recall_curve(y_test, np.where(nb.predict(X_test) > 0.5, 1, 0))
PrecisionRecallDisplay(precision=precision, recall=recall).plot()
```

```
<sklearn.metrics._plot.precision_recall_curve.PrecisionRecallDisplay at 0x7f64c792b350>
```




```
no_skill = len(y_test[y_test==1]) / len(y_test)
plt.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill')
plt.plot(recall, precision, marker='.', label='Naive Bayes')
# axis labels
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend()
plt.show()
```



Appendix C: Python code and Output of Decision Tree classification model

Appendix C1: Import library

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
import seaborn as sns
import numpy as np
```

Appendix C2: Load cleaned dataset

```
df = pd.read_csv("cleaned_car_insurance_dataset.csv")
df = df.drop(['Unnamed: 0'],axis=1)
df.head()
```

	AGE	GENDER	RACE	DRIVING_EXPERIENCE	INCOME	VEHICLE_YEAR	POSTAL_CODE	VEHICLE_OWNERSHIP	MARRIED	CHILDREN	CREDIT_SCORE	ANNUAL_MILEAGE	PAST_ACCIDENTS	OUTCOME
0	0.953343	-0.622033	-0.354414	-0.898978	-0.384870	1.594129	0.184547	0.223370	0.500337	-0.701112	0.532798	0.078642	-0.136456	0.0
1	-2.537095	-0.802031	0.146777	-0.147050	0.303880	-0.340219	0.361956	-0.381019	0.240873	0.248644	0.022340	-0.273917	-0.108320	1.0
2	-1.385940	-0.290501	-1.482303	-1.232866	-0.422788	-0.479020	-0.578699	0.117883	-0.391011	-0.245258	0.527691	-0.291033	-0.089770	0.0
3	-2.382989	1.825815	-1.578011	0.208343	0.042549	-0.239941	0.408163	0.232866	-0.369905	-0.603366	-0.337437	0.258680	-0.116753	0.0
4	-0.886031	1.549266	-1.416426	0.222215	0.687438	-0.327309	0.337817	0.076373	-0.445377	-0.195532	0.264539	-0.558849	-0.091635	1.0

Appendix C3: Data Splitting using Stratified Shuffle Split with the split ratio of 80:20 where 80% of the dataset is the train set while 20% is the test set

```
x = df.iloc[ : , :-1]
y = df.iloc[ : , -1]

from sklearn.model_selection import StratifiedShuffleSplit
sss = StratifiedShuffleSplit(n_splits=30, test_size=0.2, random_state=42)
sss.get_n_splits(x, y)

for train_index, test_index in sss.split(x, y):
  X_train, X_test = x.iloc[train_index], x.iloc[test_index]
  y_train, y_test = y.iloc[train_index], y.iloc[test_index]

print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
(8000, 13) (8000,)
(2000, 13) (2000,)
```

Appendix C4: Decision Tree model training

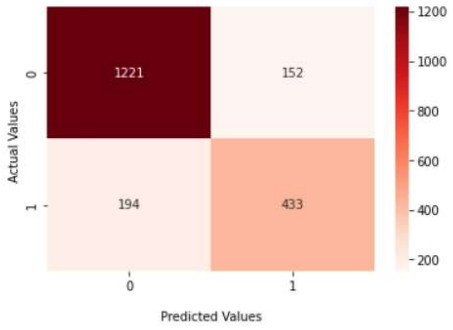
```
clf = DecisionTreeClassifier(criterion="gini", max_depth=8, random_state=40)
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
```

Appendix C5: Decision Tree Classification Report

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred, target_names=['disapproved', 'approved']))
```

```
              precision    recall  f1-score   support

 disapproved       0.86      0.89      0.88      1373
    approved       0.74      0.69      0.71       627

    accuracy                           0.83      2000
   macro avg       0.80      0.79      0.80      2000
weighted avg       0.82      0.83      0.83      2000
```

Appendix C6: Confusion Matrix



Appendix C7: Decision Tree ROC-AUC score

```
from sklearn.metrics import roc_auc_score, roc_curve
roc_auc_score(y_test, y_pred)
```

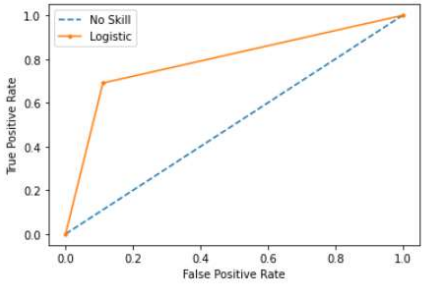
```
0.7899418147434402
```

Appendix C8: Plot Decision Tree ROC Curve

```
ns_probs = [0 for _ in range(len(y_test))]
lr_probs = clf.predict(X_test)

# calculate scores
ns_auc = roc_auc_score(y_test, ns_probs)
lr_auc = roc_auc_score(y_test, lr_probs)

# summarize scores
print('No Skill: ROC AUC=%.3f' % (ns_auc))
print('Logistic: ROC AUC=%.3f' % (lr_auc))
```

```
No Skill: ROC AUC=0.500
Logistic: ROC AUC=0.790
```

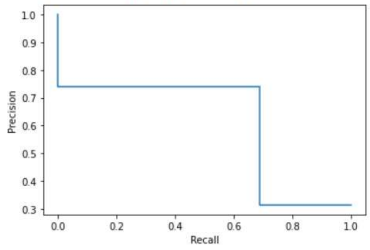
```
from matplotlib import pyplot

# calculate roc curves
ns_fpr, ns_tpr, _ = roc_curve(y_test, ns_probs)
lr_fpr, lr_tpr, _ = roc_curve(y_test, lr_probs)

# plot the roc curve for the model
pyplot.plot(ns_fpr, ns_tpr, linestyle='--', label='No Skill')
pyplot.plot(lr_fpr, lr_tpr, marker='.', label='Logistic')

# axis labels
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')

# show the legend
pyplot.legend()

# show the plot
pyplot.show()
```



Appendix C9: Plot Decision Tree Precision-Recall Curve

```
from sklearn.metrics import precision_recall_curve
precision, recall, threshold = precision_recall_curve(y_test, np.where(clf.predict(X_test) > 0.5, 1, 0))
PrecisionRecallDisplay(precision=precision, recall=recall).plot()
```

```
<sklearn.metrics._plot.precision_recall_curve.PrecisionRecallDisplay at 0x7fa72e203a90>
```




```
import matplotlib.pyplot as plt
no_skill = len(y_test[y_test==1]) / len(y_test)
plt.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill')
plt.plot(recall, precision, marker='.', label='Decision Tree')
# axis labels
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend()
plt.show()
```