Safe Driver Prediction Classification

Karthik Sonti (002784422)
Bharvi Patel (002786643)
Ankita Patel (002981420)
College of Engineering
Northeastern University
Toronto, ON

Abstract

The safe driver classification problem is a supervised machine learning problem that involves predicting whether a driver is safe or not based on various features or attributes. This problem is essential for insurance companies and other organizations that are interested in identifying safe drivers in order to minimize the risk of accidents and other incidents. By accurately identifying safe drivers, these organizations can offer lower insurance premiums and other incentives, which can help encourage safer driving behavior and reduce the overall risk of accidents. The goal is to build a model that can accurately predict whether a driver will file for insurance next year based on various features about the driver and the car. In this report, we implement and test multiple machine learning models, KNN, Decision Tree, Random Forest, and Gaussian Naïve Bayes. It turns out that KNN and Random Forest give us similar accuracy, prediction, recall, and F1 scores.

Keywords- KNN, Decision Tree, Random Forest, Gaussian Naïve Bayes, One Hot Encoding.

1 Introduction

Driving safety has always been a focus of concern. For the government, it will be great if we can reduce the traffic accidents, and one good way is to educate those drivers with a higher probability of causing traffic problems. On the other hand, insurance is a necessity for each driver to reduce the loss as much as possible in case traffic accidents happen. Statistically speaking, better drivers have less chance of being involved in traffic accidents thus resulting in less payment from the insurance companies on average. However, nowadays, drivers pay almost the same amount of money for insurance, which seems unreasonable due to their varying driving behaviors. It does not seem fair that a good driver being cautious on the road for years has to pay so much and the inaccuracies in car insurance companies' claim predictions are one of the most important reasons that raise the cost of insurance for good drivers and reduce the price for bad ones. It motivates us to build a predictive model that can accurately predict the probability of a driver filing an insurance claim next year based on the available features. Such a model could be used by insurance companies to identify high-risk drivers and adjust their premiums accordingly, thereby reducing their risk exposure. The Safe Driver Prediction project is a great example of how machine learning can be used to solve real-world problems in the insurance industry. By leveraging machine learning algorithms and techniques, the project aims to improve the accuracy of insurance risk assessment, reduce the risk of fraud, and ultimately provide better insurance coverage to customers. In this project, a Python-based application is utilized to analyze the dataset.

2 Dataset

2.1. Why is this dataset interesting?

This dataset is interesting for several reasons:

1. Real-world applicability: The dataset contains anonymized information about more than half a million car insurance policies, which makes it a valuable resource for insurance companies looking to improve their risk assessment and fraud detection capabilities.

2. High dimensionality: The dataset contains a large number of features (17 in total) with 30240 instances making it a challenging problem for machine learning algorithms. The high dimensionality of the dataset requires careful feature selection and preprocessing to improve the accuracy of the predictive models.

3. Feature engineering potential: The dataset contains a mix of categorical and numerical features, which provides opportunities for feature engineering. Feature engineering involves creating new features from the existing ones to improve the accuracy of predictive models.

2.2. Dataset Features

These are the basic features of the dataset.

Dataset	Characteristics	Attribute Characteristics	Associated Tasks	Number of Instances	Number of Attributes
Driver Dataset	Multivariate	Real	Classification	30240	17

3 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is the process of analyzing and summarizing a dataset to understand its main characteristics, relationships, and patterns. EDA is typically done before modeling to gain insights into the data and to help identify potential issues that may impact the performance of the model. EDA is an important step in the data science process as it helps to identify potential issues with the data, to gain a deeper understanding of the relationships between variables, and to identify potential predictors of the target variable. By conducting EDA, data scientists can make informed decisions about which modeling techniques to use and how to preprocess the data to achieve the best results. EDA involves several steps, including data cleaning, data visualization, and statistical analysis. The goal of EDA is to answer questions about the data and to generate hypotheses that can be tested using statistical models. The common tasks involved in EDA are listed below.

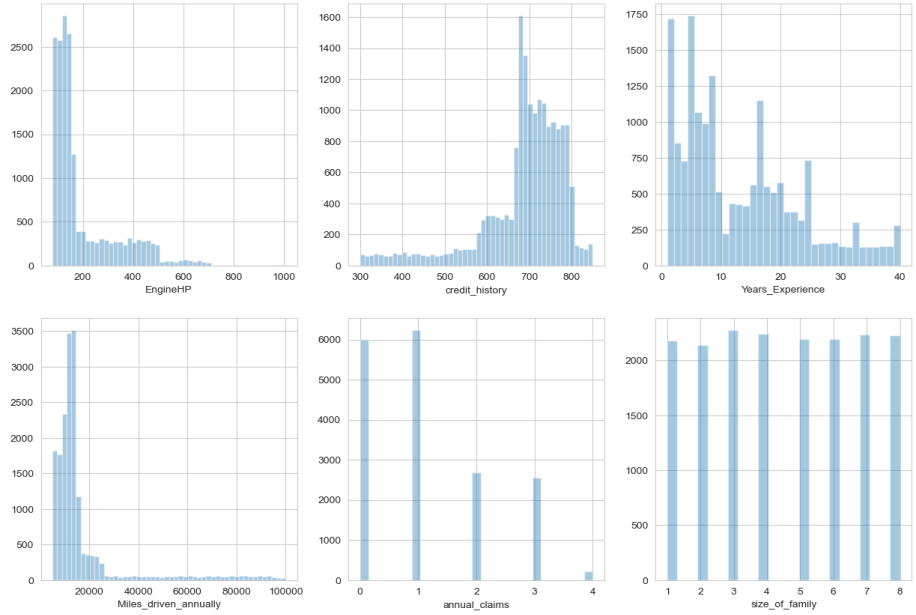

Fig 3.1: Distribution of data for different features.

3.1. Data Loading and Analysis

The first step in the project is to load the dataset using the pandas inbuilt modules. The dataset file that contains information about drivers and their cars, such as age, gender, car model, and so on. Once it's done, data analysis on the dataset is done. This includes visualizing the distribution of the target variable (i.e., whether a driver filed a claim or not). This also includes plotting various charts to understand the distribution of each feature in the dataset and how they are related to the target variable.

3.2. Data cleaning and preprocessing

The next step is to preprocess the data before training the machine learning models. This includes handling missing values, encoding categorical features, and scaling numerical features. The preprocessing steps include techniques such as one-hot encoding and standardization to transform the data into a format that can be used by the machine learning models.

3.2.1. What is Categorical Data?

Categorical data are variables that contain label values rather than numeric values. The number of possible values is often limited to a fixed set. Categorical variables are often called nominal. Some examples include:

- A “Vehical_type” variable with the values: “Car”, “Truck” and “Van”.
- A “State” variable with the values: “IL”, “NJ” and “CT”.
- A “credit_history_bucket” variable with values “Fair”, “Good” and “Very Good”

3.2.2. What is the Problem with Categorical Data?

Some algorithms can work with categorical data directly. For example, a decision tree can be learned directly from categorical data with no data transform required (this depends on the specific implementation). Many machine learning algorithms cannot operate on label data directly. They require all input variables and output variables to be numeric. In general, this is mostly a constraint of the efficient implementation of machine learning algorithms rather than hard limitations on the algorithms themselves. This means that categorical data must be converted to a numerical form. If the categorical variable is an output variable, you may also want to convert predictions by the model back into a categorical form in order to present them or use them in some application. There are various methods of categorical encoding such as One-Hot-Encoding, target encoding, label-encoding, and many more. We will be using One-Hot-Encoding for this experiment.

3.2.3. ONE HOT ENCODING

A categorical feature is removed and new binary features are added depending on the number of unique values in the categorical variable.

Type
Car
Truck
Van

Car	Truck	Van
1	0	0
0	1	0
1	0	1

Figure 3.2.3.1: Illustration of One Hot Encoding.

In the above diagram the categorical feature ‘vehical_type’ consists of 3 unique values, therefore 3 new binary features are required. A ‘1’ is placed in the corresponding row of its color and the rest of the values are made ‘0’. The binary variables are often called “dummy variables” in other fields, such as statistics.

3.4. Data division

It is common to divide the dataset into training and testing sets. The training set is a set of data to train the considered model. Now the testing set is used to give some insights into model performance. The split of training and testing can occur in many ways. However, the training set gets most of the data. In this project, I have used 70% of the data for training and 30% for testing purposes

4 Model Building

Generally speaking, this problem is a classification problem, which aims at predicting whether a driver will perform auto insurance claim based on the previous data. In this project, we have used four different classification models

which are KNN, Decision Tree, Random Forest, and Gaussian Naïve Bayes. KNN makes use of the nonlinear boundary (either in the original feature space or transformed feature space) of different classes. For the data that is hard to separate in the feature space clearly, KNN depends more on the density of the samples distribution, which doesn't have a meaningful correlation between the labels. Decision Trees are a hierarchical classification model where each internal node represents a feature, each branch represents a decision rule based on that feature, and each leaf node represents a class label. Decision Trees split the data recursively based on the feature that provides the most information gain at each split. Besides Decision Tree, Random Forest is an ensemble algorithm that builds multiple Decision Trees and aggregates their predictions through a majority voting approach. Random Forest uses a random subset of features and data points for each Decision Tree to prevent overfitting and increase the accuracy of the final prediction. On other hand Gaussian Naive Bayes is a probabilistic algorithm that models the joint distribution of features and class labels using the Bayes theorem. Naive Bayes assumes that the features are independent of each other, which simplifies the computation of the likelihood probabilities. Gaussian Naive Bayes assumes that the continuous features follow a Gaussian distribution. After running all four classifiers on dataset and considering all the features, we are getting classification score as shown in the below figure.

Classifier	**Score**
KNN	0.514
Decision Trees	0.511
Guassian Navie Bayes	0.503
Random Forest	0.524

Table 4.1: Classification scores for different models.

4.1. Feature Selection

Feature selection is the process of selecting a subset of relevant features from a larger set of features to use in a model. It is an important step in machine learning, as it can help to improve model performance by reducing overfitting, increasing interpretability, and decreasing computation time. Here are some reasons why feature selection is important in machine learning:

- **Improved Model Performance**: By selecting only the most relevant features, feature selection can help to improve the accuracy and generalization of a model, as it reduces the noise and irrelevant features that can negatively impact performance.
- **Reduced Overfitting:** Feature selection can help to prevent overfitting, which occurs when a model is trained to fit the noise in the data, rather than the underlying patterns. By selecting only the most important features, the model is less likely to learn noise in the data, and more likely to generalize well to new data.
- **Increased Interpretability:** By reducing the number of features used in a model, feature selection can help to increase its interpretability. This can be important in many domains, such as healthcare and finance, where decisions made by models need to be easily understandable and explainable.

F Score and Mutual Information are two commonly used techniques for feature selection in machine learning.

F Score: F Score is a statistical measure that is used to evaluate the differences between two groups of data. In feature selection, the F Score is used to evaluate the discriminative power of each feature. The F Score measures the ratio of the variance between the groups (i.e., the safe and unsafe driver groups) to the variance within the groups. Features with high F Scores are considered more important for prediction.

Mutual Information: Mutual Information is a measure of the mutual dependence between two variables. In feature selection, Mutual Information is used to evaluate the amount of information that a feature provides about the target variable (i.e., safe or unsafe driver). Features with high Mutual Information scores are considered more important for prediction.

Both F Score and Mutual Information are used as filter methods in feature selection, meaning they evaluate the features independently of the model. These techniques are simple and computationally efficient, making them useful

for large datasets. However, they may not be the best choice if the relationship between the features and the target variable is complex or nonlinear. In such cases, wrapper or embedded methods may be more appropriate.

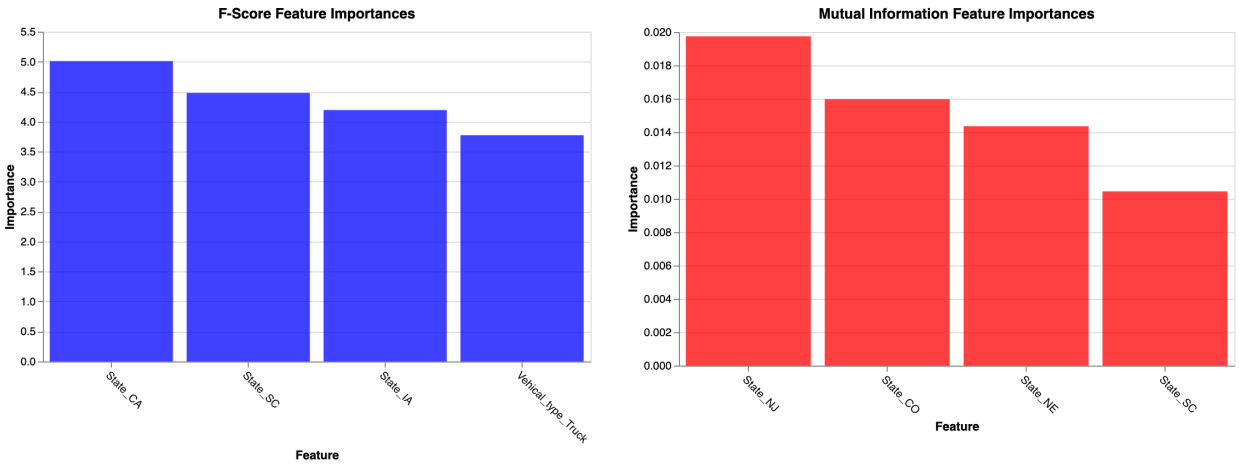

Figure 4.1.1: Important Features reported by F score and Mutual Information

.

4.3. Hypothesis Testing

Hypothesis Testing is a type of analysis in which you put your assumptions about a population parameter to the test. It is used to estimate the relationship between 2 statistical variables. Let's define the hypothesis for this problem.

Ho – Selecting all features.

H1 = Selecting only particular features (4) for better prediction.

After calculating P-value, a statistical measure that helps to determine the significance of an observed relationship, the alternative hypothesis (H1) is very weak. It's better to stick with all features rather than particular set.

4.4. Hyper-Parameter Tuning

Hyper parameter tuning is the process of selecting the optimal values for the hyper parameters of a machine learning model in order to improve its performance on a given task. Hyper parameters are the settings that cannot be learned from the data, unlike model parameters that are learned during training. Examples of hyper parameters include the learning rate, regularization strength, the number of hidden layers, the number of neurons in each layer, the activation function, and others. These hyper parameters play a crucial role in determining the performance of a machine learning model. Selecting optimal hyper parameters can be a challenging task, as it often requires extensive experimentation and computational resources.

For this project, we have ran KNN with different neighbours (neighbours=1, 5, 10, 15….) and different metrics (Manhattan, Euclidean), Decision Trees with different estimators (n_estimators=100, 200, 300…..).

4.5. Model Evaluation

The final step is to evaluate the performance of the model using various metrics such as accuracy, precision, recall, and F1 score. The accuracy is used to measure that how well the prepared model is able to automatically identify the data. It's the percentage of labels that have been correctly classified.

$$Accuracy = \frac{TP+TN}{TP+TN+FN+FP} \; x \; 100\%$$

The principal of confusion matrix formation is depicted in this table 4.5.1.

	Positive	**Negative**
Positive	TP	FP
Negative	TN	FN

Table 4.5.1: Confusion Matrix Principle.

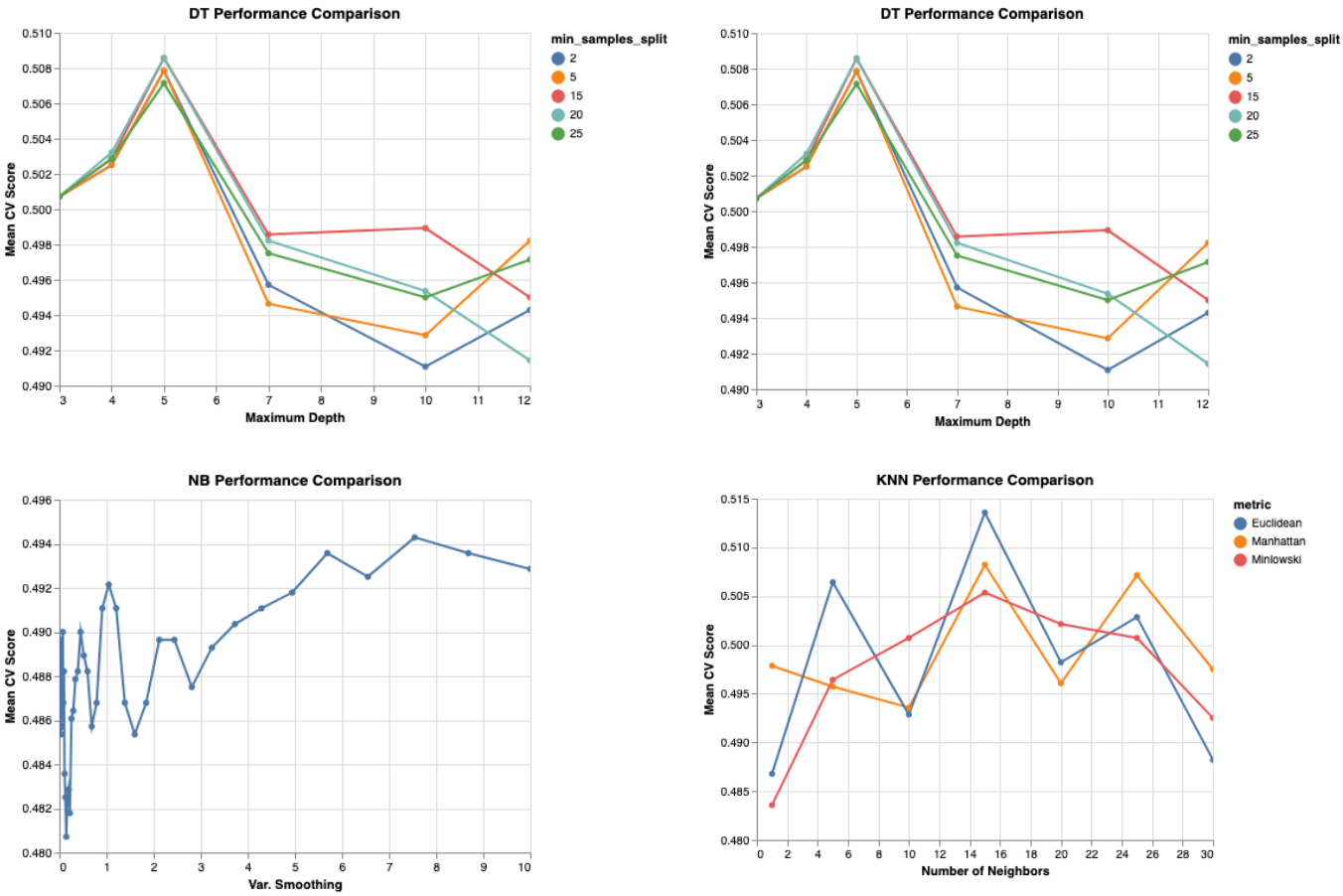

Figure 4.5.1: Illustration of different classifiers results.

4.6 Model Outcome

As seen above, the precision is 50% for both the target values. The recall for KNN, Random Forest model is more than 50% which indicates they would be a better model with respect to recall. Looking at F1 score, we can observe that all the classifiers except Gaussian Naive have similar performances

5 Conclusions

. Thus we can conclude that KNN and Random Forest give us similar accuracy, prediction, recall and F1 scores.

6 References

[1] Safe Driver Prediction Base on Different Machine Learning Models - Xianzhi Liu, Qingquan Song.
[2] One Hot Encoding - https://www.geeksforgeeks.org/ml-one-hot-encoding-of-datasets-in-python/