1 Introduction

This project investigates using machine learning to classify aspects of driving behavior, road conditions, and traffic based on vehicle sensor data. Utilizing datasets from both Opel Corsa and Peugeot 207 sourced from Kaggle, the project aims to identify the most effective classification algorithms among Support Vector Machines (SVM), Logistic Regression, and k-Nearest Neighbors (kNN) [1]. The datasets are comprised of sensor readings from each respective vehicle, structured around 14 specific features. These features serve as the basis for classifying the data into labeled outputs across three main categories: **Driving Style** (Even Pace and Aggressive), **Road Surface Conditions** (Smooth Condition and Uneven Condition), and **Traffic** (Low Congestion, High Congestion, and Others). This project focuses on the **Driving Style** variable, given its significance for risk assessment and accident prevention through its classification.

2 Methodology

2.1 Data Pre-Processing

In the data pre-processing section, we define our file paths and load the datasets. We then combine the datasets into a singular combined dataset for simplicity. Upon review of our dataset's initial rows, we removed inconsistencies such as rows with missing values, unnamed rows, and the "roadSurface" and "traffic" variables. Columns which were labelled as objects, unsuitable to our models, were converted to a numerical (float64) format. NaN values were also handled.

2.2 Data Labelling

In the data labelling section, we encode the data, to fully ensure we have numeric variables. It's essential to split the dataset into training and testing sets, to ensure we can train our models on one portion of data, and compare it against the test set to evaluate performance. We verify the label mapping, and then perform a stratified split, to ensure the test and training sets have similar distributions.

2.3 Data Scaling

To scale our data, we use the scaler from sklearn [1], and compute the mean and standard deviation. Scaling is used to centre the feature value around 0 with a standard deviation of 1. This shows us that the scaler has standardized features, ensuring we have scaled our features to variance.

3 Model Training and Results

The performance of each model is evaluated by a set of metrics which assess the efficiency that the model demonstrates. These metrics are accuracy, which provides the proportion of true results, precision, which shows the models ability to classify only the relevant instances, and recall, which assess the models ability to classify all the relevant instances. The results of the model training are defined as follows [2]:

Table 1: Classification Metrics for EvenPaceStyle (Majority)

Classifier	**Precision**	**Recall**	**F1-Score**	**Accuracy**
k-Nearest Neighbors	0.94	0.96	0.95	0.92
Support Vector Classifier	0.90	0.99	0.94	0.89
Logistic Regression	0.89	0.99	0.94	0.88

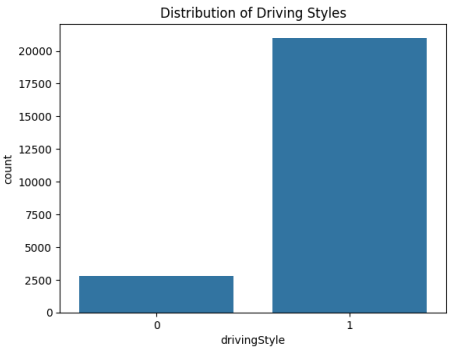
Table 2: Classification Metrics for AggressiveStyle (Minority)

Classifier	**Precision**	**Recall**	**F1-Score**	**Accuracy**
k-Nearest Neighbors	0.65	0.57	0.61	0.92
Support Vector Classifier	0.75	0.11	0.20	0.89
Logistic Regression	0.41	0.05	0.09	0.88

The metrics above show us that for the EvenPaceStyle (majority) class, all models score extremely well, with k-Nearest Neighbors excelling. The results across all models show that they have exceptional capability in identifying the majority class. The high precision and recall rates demonstrate that the models are adept at not only capturing true instances of the class, but also in minimizing false positives.

However, looking at the AggressiveStyle (Minority) class, we can see that the results are not as promising. The models have high accuracy, but have significantly lower recall, precision and F1-scores in comparison to the majority class. These low scores tell us that the models struggle to accurately detect the minority class. This discrepancy is apparent when we examine the dataset, and can see a severe data imbalance.

Here, we can see the significant data imbalance. The EvenPaceStyle class substantially outweighs the AggressiveStyle class, which explains why the majority class performs so well in comparison to the minority class. To combat this imbalance, we can implement the Synthetic Minority Over-sampling Technique (SMOTE) [3]. This technique can be used to balance the dataset, by generating more examples for the minority class.



3.1 Data Imbalance

Utilizing SMOTE, we re-train and test our models again, and get the following results:

Table 3: Classification Metrics after SMOTE (Majority Class)

Classifier	**Precision**	**Recall**	**F1-Score**	**Accuracy**
K-Nearest Neighbors	0.98	0.85	0.91	0.85
Support Vector Machine	0.97	0.80	0.88	0.81
Logistic Regression	0.96	0.68	0.80	0.69

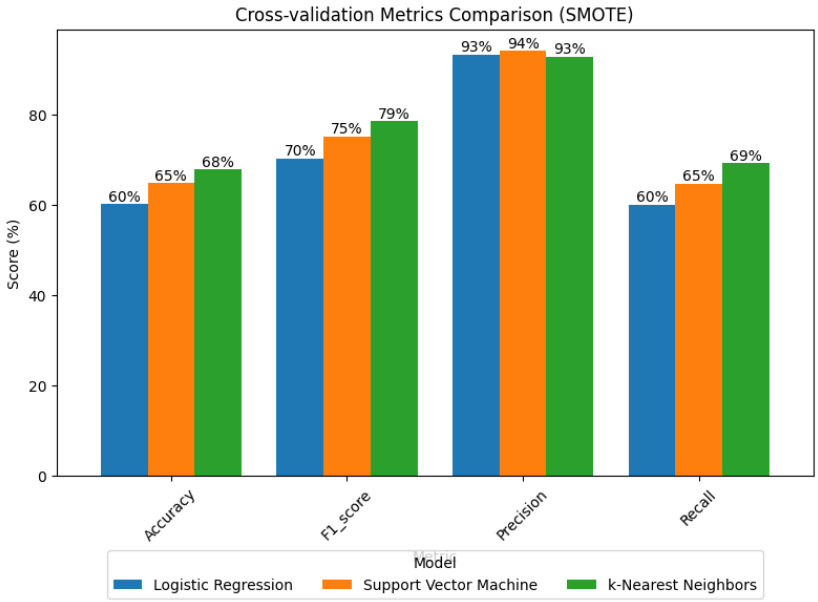
Table 4: Classification Metrics after SMOTE (Minority Class)

Classifier	**Precision**	**Recall**	**F1-Score**	**Accuracy**
K-Nearest Neighbors	0.43	0.87	0.58	0.85
Support Vector Machine	0.36	0.83	0.50	0.81
Logistic Regression	0.24	0.76	0.36	0.69

After applying SMOTE and retraining our models, the classification metrics change significantly. Notably, the accuracy of each model decreased to 85%, 81% and 69% respectively from their pre-SMOTE levels of 92%, 89%, and 88%. Precision for the minority class has decreased, indicating a higher false positive rate, while the F1-score has improved, showcasing a better balance between precision and recall. The recall rates for the minority class have seen a monumental average increase of 709%, increasing from as low as 5% to 75% in the case of Logistic Regression. This significant increase indicates that the models are considerably better at identifying true positives in the minority class. This improvement shows our models are now better suited to handle both classes overall. The majority class continues to perform well after these changes, revealing that the data balancing does not subtract from the overall model reliability.

3.2 Cross Validation

Finally, we will utilize cross validation to test the models in an iterative fashion, ensuring they are consistent and perform well on unseen data [4].



The cross validation results reflect the models generalized ability across both driving behaviours. This leads to a decline in the accuracy, recall and F1 score metrics, with the exception of precision. This decline occurs as the models, are now accounting for both the majority and the minority classes together. The models all exhibit high precision, which tells us that they are generally more accurate when they predict a class. After cross validation, k-Nearest Neighbors (kNN) once again outputs the best results.

Conclusion

In conclusion, all models performed well in the majority classes case. However, across each scenario, k-Nearest Neighbors (kNN) outperformed the other models, consistently scoring the highest in each relevant classification metric. Post data balancing and cross validation, kNN demonstrated proficient performance, maintaining a respectable accuracy of 68% and an F1 score of 79%. These scores show that kNN is a reliable and efficient model across complex scenarios and data imbalances. It's clear, after these considerations, k-Nearest Neighbors is the most efficient of the three models in the classification of driving styles.

References

1. Scikit-learn documentation
2. Classification report - Scikit-learn
3. SMOTE - Imbalanced-learn
4. Cross-validation - Scikit-learn