Gesture UI Project Report

Rodrigo Almeida - G00377123

1 Introduction

THis project undertakes a comprehensive analysis of the dataset *Traffic, Driving Style and Road Surface Condition* [1]. The dataset includes low-level parameters gathered through On-Board Diagnosis II (OBD-II) interfaces in vehicles and accelerometers embedded in smartphones. The data collection approach aimed to capture the dynamics between the driver's vehicles, and environmental conditions. The dataset focuses on data related to two specific car models - Peugeot 207 1.4 HDi and Opel Corsa 1.3 HDi. The project aims to compare the efficacy of three classification algorithms - Support Vector Machine (SVM), Logistic Regression, and K-Nearest Neighbors, through deep data preprocessing, feature engineering, and application of cross-validation techniques, seeking to optimize model performance.

2 Methodology

2.1 Data Pre-processing

One of the biggest challenges in machine learning is the quality of the data. The quality of the data is crucial to the performance of the model. The raw dataset comes from Kaggle [1] and comprises data gathered from two diesel cars and are presumed to be two distinct road types, resulting in the dataset being partitioned across four CSV files. Each vehicle's data is represented in two separate files, each corresponding to a different road condition encountered during the data collection phase. The collected dataset was then labelled for three key attributes: *traffic conditions, driving style, and road surface quality*, which makes it well suited for supervised learning. With the attributes specifically labelled, it is a classification issue.

The data pre-processing phase is a critical step in the machine learning pipeline. It involves cleaning, transforming, and preparing the data for the model.

2.1.1 Imbalanced Dataset

On the dataset, the number of instances for each class was found to be imbalanced, which can lead to poor performance of the model.

Below is a description and number of instances for each class in the dataset:

- **Driving Style:**
  - EvenPaceStyle: 21,016 instances
  - AggressiveStyle: 2,759 instances
- **Road Surface Condition:**
  - SmoothCondition: 14,237 instances
  - UnevenCondition: 6,289 instances
  - FullOfHolesCondition: 3,249 instances
- **Traffic Condition:**
  - LowCongestionCondition: 17,764 instances
  - HighCongestionCondition: 3,017 instances
  - NormalCongestionCondition: 2,994 instances

For this project, *Driving Style* class column was chosen due to its significant imbalance and potential for improvement. The *AggressiveStyle* class has 2,759 instances while the *EvenPaceStyle* class has 21,016 instances. The dataset is imbalanced, and the model may be biased towards the majority class. To resolve the issue of the imbalanced dataset that was observed, the use of other techniques such as oversampling and under-sampling was considered. SMOTE (Synthetic Minority Over-sampling) which is a statistical technique for increasing the number of instances in the minority class was used. SMOTE works by selecting examples that are close to the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line. It does have some limitations, such as the generation of noisy samples, but it is a widely used technique for dealing with imbalanced datasets. [2]. Below is what the *Driving Style* dataset looks like after applying SMOTE.

- **Driving Style:**
  - EvenPaceStyle: 16,808 instances
  - AggressiveStyle: 16,808 instances

With the dataset now balanced, the next step was to remove the **Unnamed: 0** column. This column was not needed for the analysis and was removed from the dataset. After having that completed it was found that there were missing values in the dataset as shown in Table 1.

Column	Missing Values
VehicleSpeedInstantaneous	9
EngineLoad	5
EngineCoolantTemperature	5
ManifoldAbsolutePressure	5
EngineRPM	5
MassAirFlow	5
IntakeAirTemperature	5
FuelConsumptionAverage	5

TABLE 1: Missing Values in the Dataset

Due to the small number of missing values relative to the size of the dataset, imputation seems to be a practical approach. For the numerical columns with missing values, it can be filled with the mean or median of the column. For the categorical columns, the missing values can be filled with the mean of the column. In this case was chosen to fill the missing values with the mean of the column, as it data is normally distributed.

2.2 Data Labelling

For the categorical variables - *roadSurface*, *traffic*, *drivingStyle* - it was converted into a format that can be used for machine learning models. Since *drivingStyle* is the target variable it was encoded using the *LabelEncoder* from the *sklearn.preprocessing* module and is now represented as 0 and 1. The continuous feature, on columns such as *AltitudeVariation*, *VehicleSpeedInstantaneous*, *VehicleSpeedVariation*, are now standardized and centered around 0(mean) and 1(standard deviation). The categorical variables *roadSurface* and *traffic* are encoded using the *OneHotEncoder* from the *sklearn.preprocessing* module. Each column represents the presence (True/1) or absence (False/0) of a category.

2.3 Data Scaling

Feature scaling is a method used to standardize the range of independent variables or features of the data. In data processing, it is also known as data normalization and is generally performed during the data pre-processing step.

The *Standard Scaler* was used to scale the data. The Standard Scaler standardizes the features by removing the mean and scaling to unit variance. This results in a distribution with a standard deviation of 1 and a mean of 0. The formula for standard scaling is given by:

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

Where x is the original feature value, μ is the mean of the feature, and σ is the standard deviation of the feature. This process of subtracting the mean and dividing by the standard deviation ensures that all features contribute equally to the result, a critical consideration for algorithms like Support Vector Machines (SVM), k-Nearest Neighbors (kNN), and Logistic Regression were used in this project.

2.4 Data Analysis and Visualisation

As part of the analysis aiming to classify the driving styles, the data was visualized to understand the distribution of the features and the relationship between the features and the target variable. The PCA (Principal Component Analysis) was used to visualize the data in 2D. The PCA is a dimensionality reduction technique that is used to reduce the dimensionality of the data to 2 or 3 dimensions so that it can be visualized. [3]

Figure 1 shows the PCA visualization of the data. The classes are not well separated, suggesting the data is not linearly separable and a linear model may not be the best choice for this dataset. The axes are labelled as PC1: *VehicleSpeedInstantaneous* and PC2: *RoadSurface_UnevenCondition*, implying that these two features contribute most to the variance in the data. *VehicleSpeedInstantaneous* might be a defining characteristic of the driving style, while *RoadSurface_UnevenCondition* might also influence the driving style. Despite some level of separation, there is also significant overlap between the classes, which indicates that not all driving style characteristics are captured by these two features, or that there maybe a wide range of behaviours within each style that could cause this overlap. The presence of clusters in the data suggests a potential for classification models to predict the driving style based on the features.

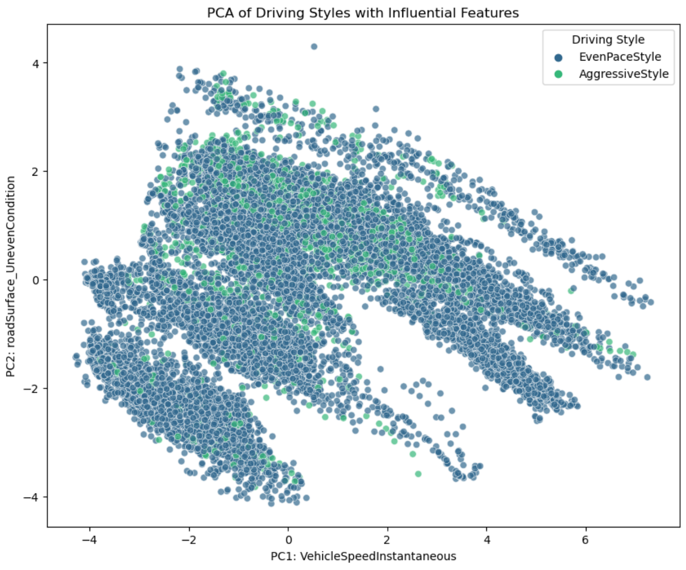

Fig. 1: PCA Visualization of the Data

3 Experiments and Results

3.1 Linear Regression Model

Now, with the knowledge about the imbalanced dataset, the new dataset processed with SMOTE was used to train the linear regression model. To find the best parameters for the model, the *GridSearchCV* method was used to find the best parameters for the model. The results from *GridSearchCV* indicated that the best parameter for C was *0.1*, which was used to train the model. The model was then evaluated, using the balanced dataset and the results are shown in Table 2 and Table 3. The F1-Score of *0.36* indicates that the model was able to identify *AggressiveStyle* to some extent, but it is not very precise. On the other hand an F1-Score of *0.80* for *EvenPaceStyle* indicates that despite some misclassifications, the models generally perform well for this class. The overall accuracy of the model was *70%*.

Class	**Precision**	**Recall**	**F1-score**	**Support**
AggressiveStyle	0.24	0.73	0.36	547
EvenPaceStyle	0.95	0.69	0.80	4208
Accuracy		0.70		
Macro Avg	0.59	0.71	0.58	4755
Weighted Avg	0.87	0.70	0.75	4755

TABLE 2: Classification Report for Logistic Regression

After optimizing the model parameters and evaluating its predictive performance, it is important to understand

Confusion Matrix for Logistic Regression		
Actual/Predicted	**AggressiveStyle**	**EvenPaceStyle**
AggressiveStyle	398	149
EvenPaceStyle	1284	2924

TABLE 3: Confusion Matrix for Logistic Regression

which features contribute most significantly to the model's predictions. In logistic regression, the coefficients associated with each feature can indicate the importance and influence of that feature on the model's output.

To visualize the impact of each feature, a coefficient plot was constructed. This plot ranks the features by their coefficients, providing clear insights into which features are most informative for predicting the target variable.

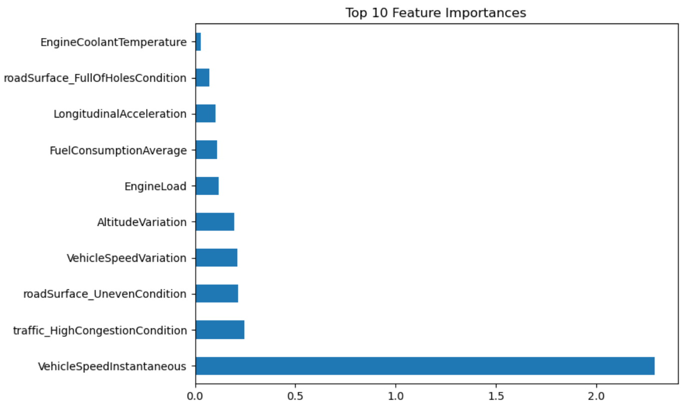

Fig. 2: Top 10 Feature Importance in the Logistic Regression Model

This coefficient plot shows the relative importance of features in predicting driving styles. Features with larger absolute values of coefficients have a greater impact on the model's decision-making process. It is a good way to understand which features are most important for the model's predictions. The top 10 features are shown in Figure 2.

3.2 Support Vector Machine Model

Before training the model with the Support Vector Machine (SVM), a range of C values and Kernels - linear, poly, rbf and sigmoid were tested to find the optimal value. The model was trained using the *GridSearchCV* method from the *sklearn.model_selection* module. The best C and best $Kernel$ parameters were identified through cross-validation, optimizing for model accuracy. The best parameters and corresponding score obtained from the grid search were accessed via `svm_grid.best_params_` and `svm_grid.best_score_`, respectively. For this model, the best parameters were $C = 10$, suggesting that the model has settled on a middle ground and $Kernel = rbf$ as it can handle non-linear data.

Before proceeding to evaluate the model with the balanced and pre-processed dataset, the model was trained using the original dataset to see how it would perform. The accuracy of the model was evaluated returning a score of *0.88*. But then, when running the classification report, it was found that the model was biased towards the majority class as shown in Table 4. It is shown that the model was able to predict the majority class with a precision of *0.88* and a recall of *1.00*, but the minority class was not predicted at all. This is a clear indication of the model bias towards the majority class, this is due to the imbalanced nature of the dataset.

Class	Precision	Recall	F1-score	Support
AggressiveStyle	0.00	0.00	0.00	547
EvenPaceStyle	0.88	1.00	0.94	4208
Accuracy		0.88		
Macro Avg	0.44	0.50	0.47	4755
Weighted Avg	0.78	0.88	0.83	4755

TABLE 4: SVM Classification Report with imbalanced dataset

Confusion Matrix		
Actual/Predicted	**AggressiveStyle**	**EvenPaceStyle**
AggressiveStyle	0	547
EvenPaceStyle	0	4208

TABLE 5: SVM Confusion Matrix with imbalanced dataset

After applying SMOTE to the dataset, Table 6 shows improvement in the model's ability to predict the minority class, which was not predicted at all in the previous trial. There seems to be a trade-off; as recall increased, precision decreased. This is expected as the model is now predicting more instances of the minority class. A precision of *0.43* for *AggressiveStyle* and a high precision of *0.98* for *EvenPaceStyle* show a significant improvement compared to the Logistic Regression Model. The recall of *0.84* and *0.86* respectively, demonstrate the model's sensitivity to picking up on the driving behaviours but also indicate that the model misses about *15%* of actual driving instances. The F1-score of*0.57* for *AggressiveStyle* show a balance between precision and recall, considering both false positives and false negatives for this class. The F1-Score of *0.91* for *EvenPaceStyle* is very good and indicates the model's performance in classifying the majority class. The overall accuracy of the model was *0.85* meaning it correctly classifies *85%* of all instances, showing a greate

Class	Precision	Recall	F1-score	Support
AggressiveStyle	0.43	0.84	0.57	547
EvenPaceStyle	0.98	0.86	0.91	4208
Accuracy		0.85		
Macro Avg	0.70	0.85	0.74	4755
Weighted Avg	0.91	0.85	0.87	4755

TABLE 6: SVM Classification Report with dataset processed with SMOTE

Confusion Matrix		
Actual/Predicted	**AggressiveStyle**	**EvenPaceStyle**
AggressiveStyle	458	89
EvenPaceStyle	606	3602

TABLE 7: SVM Confusion Matrix with dataset processed with SMOTE

3.3 K-Nearest Neighbors Model

The k-Nearest Neighbors (kNN) model was trained using the balanced dataset and the results are shown in Table

8 and Table 9. With a precision of *0.59*, the kNN model correctly predicts *AggressiveStyle 59%* of the time, this is an improvement compared to both the SVM and Logistic Regression models, suggesting fewer false positives for *AggressiveStyle*. The precision for *EvenPaceStyle* is very high at *0.98* similar to SVM, indicating that the model is highly accurate when predicting even-paced driving. The F1-Score of *0.69* for *AggressiveStyle* show a balance between precision and recall with a significant improvement over both SVM and Logistic Regression models. The F1-Score for *EvenPaceStyle* was *0.95*, which is very strong showing a good performance when classifying the majority class. The overall accuracy of *0.91* shows that the kNN model correctly classifies *91%* of all instances, which is the highest accuracy among the three models.

Class	**Precision**	**Recall**	**F1-score**	**Support**
AggressiveStyle	0.59	0.84	0.69	547
EvenPaceStyle	0.98	0.92	0.95	4208
Accuracy		0.91		
Macro Avg	0.78	0.88	0.82	4755
Weighted Avg	0.93	0.91	0.92	4755

TABLE 8: Classification Report for k-Nearest Neighbors (kNN)

Confusion Matrix for k-Nearest Neighbors (kNN)		
Actual/Predicted	**AggressiveStyle**	**EvenPaceStyle**
AggressiveStyle	458	89
EvenPaceStyle	320	3888

TABLE 9: Confusion Matrix for k-Nearest Neighbors (kNN)

3.4 Results

The kNN generally showed the best performance across most metrics, especially in terms of balanced precision and recall for both classes. SVM came after, showing its strength in recall for *AggressiveStyle*. Logist regression, while less precise for *AgressiveStyle*, still provided insights due to the model's simplicity and interpretability. All models had to deal with class imbalances present in the data, which influenced their ability to predict the less represented *AgressiveStyle* accurately. The SMOTE technique helped address this issue, improving all model's performance.

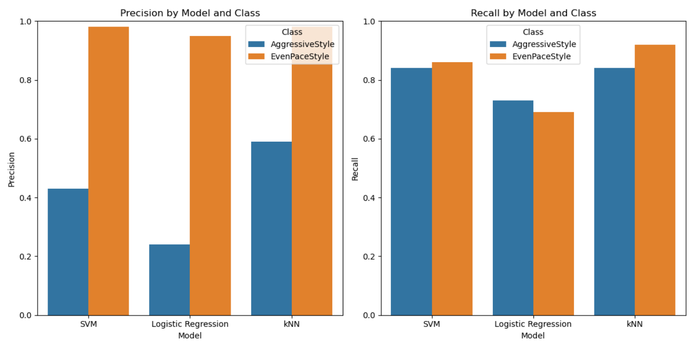

Fig. 3: Precision and Recall comparison

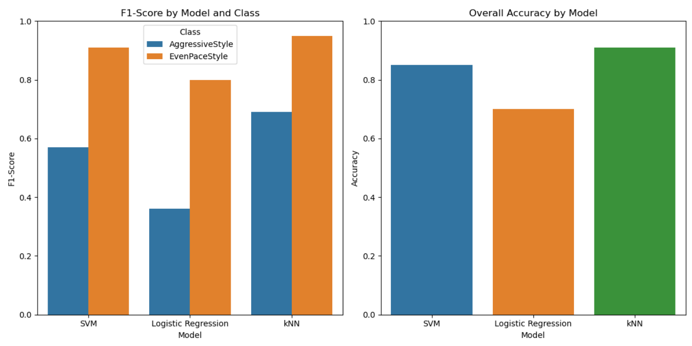

Fig. 4: F1-Score and Accuracy comparison

4 Conclusion

By comparing Logistic Regression, Support Vector Machine (SVM) and K-Nearest Neighbors (KNN) algorithms, the analysis revealed that kNN displayed a superior accuracy and balanced metrics, proving most effective to this application. SVM shoed an impressive recall, particularly for identifying the **AggressiveStyle** driving, with Logistic Regression offering interpretability, although with lower precision, showcasing the strengths of each algorithm. In conclusion, the project highlights the importance of rigorous data preprocessing and cross-validation. While the performance may vary across different attributes, it can be concluded that the classifiers generally perform well for the driving style prediction, with F1-Scores above *0.8* in the most effective classifier. One observation is that models prioritising interpretability tend to perform poorly, highlighting the trade-off between interpretability and accuracy.

References

[1] Traffic, Driving Style and Road Surface Condition. URL: https://www.kaggle.com/datasets/gloseto/traffic-driving-style-road-surface-condition.

[2] Alberto Fernández et al. "SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary". In: *Journal of artificial intelligence research* 61 (2018), pp. 863–905.

[3] Tomasz Niedoba. "Multi-parameter data visualization by means of principal component analysis (PCA) in qualitative evaluation of various coal types". In: *physicochemical problems of Mineral processing* 50.2 (2014), pp. 575–589.