A Documentation for the Building of a Model to Predict the Causes or Influences CAUSE OF DRIVER BEHAVIOR CLASSIFICATION

1. Introduction

- **Project Goals:** Build a classification model to identify the causes of various driving behaviours/patterns. The intention is to integrate it to our app in other to give some insight to the causes of some of the unsafe driving behaviours that would be detected from the analysis of drivers' raw driving data. A second use would be to make the data and benchmark performance metrics openly available to support driving behaviour research.
- **Data Source:** Real-world driving data collected from a previous experiment in Nigeria.
- **Modelling Building Approach:** The dataset has a column called'influencer' which is our target variable in the classification model, but this column contains multi-label data. This implies multi-label classification algorithms would be the most suitable for this kind of classification problem. In multi-label classification, each instance (row of data) can be assigned multiple labels simultaneously, unlike in standard single-label classification where each instance is assigned only one label. Some commonly used machine learning algorithms suitable for multi-label classification are: Decision Trees and Random Forest, Gradient Boosting Machines (GBM), Neural Networks, Linear Models (like Logistic Regression), k-Nearest Neighbors (kNN).

For this task, considering we have a mix of categorical and numerical data, and the potential complexity of multi-label interactions, **Random Forest** or a **Gradient Boosting Machine (GBM)** algorithm like XGBoost are good starting points, because they provide high performance without the need for extensive data preprocessing like scaling.

2. Project Overview

- **Target Variable:** The ‘influencer’ column in the dataset represents the target. It contains the labels (causes) for each datapoint.
- **Metrics:** The performance metrics used are accuracy, precision, recall and F1-score, Hamming and Jaccard. Due to class imbalance, cross-validation scores was also used to understand the performance of the model across different subsets of the dataset.

3. Data Loading and Exploration

- **Dataset Description:**
  - **Columns:** There are 20 columns in the dataset including *driver_id, driver_type, Address, SpeedLimit, totalMeters, speed, midSpeed, pointDate, latitude, longitude, acceleration (X,Y,Z), original acceleration (X,Y,Z), tickTimestamp, track_id, influencer (labels)*.
  - **Number of Drivers:** 21
  - **Dataset size:** The original dataset is made up of 436,714 data points with a disk size of 75mb. After data cleaning, the number of datapoints became 98,629 with a disk size of 31mb. The data was cleaned by deleting all datapoints with the value ‘no data’ in the ‘influencer’ column, this is because, the ‘no data’ values represent datapoints with no recorded influence for trips that drivers embarked on.

Exploratory Analysis:

Class Distribution (Influencer):

- The 'influencer' column, which is the target variable, shows a diverse range of classes with varying frequencies.
- The most common class is 'new driver' with 41,644 instances, followed by 'none' (which means; no influence) with 22,571 instances.
- Other classes such as 'alcohol', 'phone call', and 'bad road, busy mind, traffic jam' are also present but in smaller numbers.

  - This distribution suggests a class imbalance in the dataset, which is a key factor to consider in the model training and evaluation.
  - The class distribution was explored after data cleaning.

- **Summary Statistics:** Calculated for numerical features (count, mean, standard deviation, quartiles, min/max) see appendix B for details.
- Details of the dataset are summarized in Table 1 and 2 in Appendix B.

4. Data Preprocessing

- The following columns were dropped:

  Address (textual data), pointDate (temporal data), driver_id (may not be directly relevant to the classification task), track_id (may not be directly relevant to the classification task), Yaw (Extreme values (min -1.7T, max 1.67T), outliers.), Lateral (Constantly 0, non-variable or placeholder), 'SpeedLimit', 'tickTimestamp','Unnamed: 0'. Morso, the columns 'driver_id', 'Address', 'track_id', 'pointDate', 'tickTimestamp' will not enable the model to be used in a general context, and there are too many outliers in the 'yaw' and 'lateral' columns. Also, the 'Unnamed: 0' column is not necessary.

- **Encoding:**
  - One-Hat label encoding was applied to categorical columns: driver_type, influencer (target variable)
- **Scaling:**
  - Used a StandardScaler to standardize numerical features, achieving zero mean and unit variance.
- **Data Cleaning:**
  - Removed special characters ("*" and "**") from the labels.
  - Removed rows where the "influencer" column had the value "no data".
  - The labels (values in the influencer column) were split and converted to a list of labels for each row (datapoint)
  - Replaced any label containing the word "phone" with "phone" and "alcohol" with "alcohol" in the labels

- Transformed the values in the "influencer" column which represents the labels for each datapoint into a list of labels using the MultiBinarizer

- **Data Splitting:** The cleaned dataset is initially split into 70% training and 30% testing.

5. Feature Engineering

- **Initial Feature Set:** speed, midSpeed , height, course, acceleration, deceleration, accelerationX, accelerationY, accelerationZ, accelerationXOriginal, accelerationYOriginal, accelerationZOriginal, driver_type_prd (Driver type= private driver), driver_type_pud(Driver type=public driver)
- **Feature Creation:** No new features were created

6. Model Development

Model Training and Evaluation

The following functions were defined to setup and finetune the model as required at any stage or iteration of the model building process. Functions specifically are defined to train, evaluate, do cross validation, and generate a classification report of the model at each stage. The functions are as below.

```
        def model_config(n_e=100, random_s=42, max_d=None, n_j=-1):
            # Initializing the RandomForestClassifier with correct parameter names
            rf_classifier = RandomForestClassifier(n_estimators=n_e, random_state=random_s,
max_depth=max_d)

            # Using MultiOutputClassifier to handle multiple labels with correct parameter name
            multi_label_model = MultiOutputClassifier(estimator=rf_classifier, n_jobs=n_j)

            return multi_label_model

        def evaluate_model(y_t,X_t):
            y_pred = multi_label_model.predict(X_t)

            # Evaluation metrics for multi-label classification
            accuracy = accuracy_score(y_t, y_pred)
```

```
    precision = precision_score(y_t, y_pred, average='micro')
    recall = recall_score(y_t, y_pred, average='micro')
    f1 = f1_score(y_test, y_pred, average='micro')
    hamming = hamming_loss(y_t, y_pred)
    jaccard = jaccard_score(y_t, y_pred, average='micro')

    print(f'accuracy: {accuracy}')
    print(f'precision: {precision}')
    print(f'recall: {recall}')
    print(f'f1: {f1}')
    print(f'hamming: {hamming}')
    print(f'jaccard: {jaccard}')
    return y_pred


def perform_cross_validation(model, X_train, y_train, n_folds=5):
    """
    Perform K-fold cross-validation for the multi-label model.

      # Define the KFold cross-validator
    kf = KFold(n_splits=n_folds, shuffle=True, random_state=42)

    # Perform cross-validation

    # Perform cross-validation and compute scores
    cv_scores = cross_val_score(model, X_train, y_train, cv=kf)

    # Print the cross-validation scores
    print("Cross-validation scores:", cv_scores)

    # Print the mean and standard deviation of the cross-validation scores
    print("Mean CV score:", cv_scores.mean())
    print("Standard deviation of CV scores:", cv_scores.std())


    return cv_scores

def generate_classification_report(y_t, y_p):
    report = classification_report(y_t, y_p, target_names=y_t.columns)
    print(report)
```

Basic Model Setup

- A **Random Forest classifier** is initialized with the settings in the code snippet. It is trained on the training set and wrapped in a **Multioutput Classifier** to handle the multi-label classification task as is a standard approach for multi-label problems. Predictions are made using the test set.
- **Code Snippets of Initial Model Setup:**

```
multi_label_model = model_config(n_e=1 random_s=1 max_d=None, n_j=-1)
```

- **Initial Performance Metrics and Interpretation:**

  Accuracy: 0.92 , Precision: 0.96, Recall: 0.96, F1: 0.96, Hamming: 0.0052, Jaccard: 0.92
  Cross-validation scores: [0.91 0.91 0.91 0.90 0.91], Mean CV score: 0.91,
  Standard deviation of CV scores: 0.00381
  As evidenced by the high scores in key metrics. The model maintains robust performance across various classes, with some variation that suggests potential areas for further improvement or investigation. The consistency in cross-validation scores further reinforces the model's reliability.

  Class-wise Performance

  The classsification report for this setup of the model as **shown in Table 1 apendix A** shows varying levels of precision, recall, and F1-scores across different classes.

  Classes like ‘alcohol’, ‘excitement’, ‘new driver’, and ‘none’ exhibit high precision and recall, indicating strong performance. ‘Listening to loud music’ and ‘tiredness’ have comparatively lower scores, suggesting these classes might be more challenging for the model to predict accurately.

  The ‘micro avg’, ‘macro avg’ and ‘weighted avg’ scores reflect high overall precision, recall, and F1-scores, which align with the general model performance metrics.

Model Finetuning and Feature Engineering Iterations

Iteration 1

Code Snippet of Model setup

```
multi_label_model = model_config(n_e=100, random_s=42, max_d=None, n_j=-1)
```

Performance Metrics and Interpretation:

Accuracy: 0.99, Precision: 1.0, Recall: 0.99, F1: 1.0, Hamming: 0.0007, Jaccard: 0.99
Cross-validation scores: [0.99, 0.99, 0.99 0.99 0.99] Mean CV score: 0.99 Standard deviation of CV scores: 0.0009

This finetuned iteration has resulted in a model with exceptional accuracy, precision, recall, and F1-scores. It shows remarkable consistency across different classes and datasets. While performance is uniformly high, the relatively lower score in 'listening to loud music' might warrant targeted improvements or investigations.

Class-wise Performance

The model displays near-perfect precision and recall across almost all classes, with F1-scores mostly in the high 90s or 100%. 'Listening to loud music' is an outlier with a lower F1-score (0.90), suggesting that this class is more challenging for the model. 'Micro avg', 'macro avg', 'weighted avg', and 'samples avg' scores are consistently high, reinforcing the model's overall excellent performance **See Table 2 in appendix A for classification report.**

Iteration 2

Code Snippet of Feature Engineering

```
# Second Feature Engineering step, this is to make the model location and distance travelled agnostic.
        X_train = X_train.drop(['longitude', 'latitude', 'totalMeters'], axis=1)
        X_test=X_test.drop(['longitude', 'latitude', 'totalMeters'], axis=1)
```

Performance Metrics and Interpretation:

Accuracy: 0.86, Precision: 0.97, Recall: 0.85, F1: 0.91, Hamming: 0.01106, Jaccard:0.83
Cross-validation scores: [0.85, 0.85, 0.86, 0.85, 0.85] Mean CV score: 0.85 Standard deviation of CV scores: 0.00345

In this iteration, we did feature engineering to drop the columns 'longitude', 'latitude', 'totalMeters'. This is to make the model location and distance travelled agnostic.

While precision remains high, the model's ability to correctly identify all relevant instances (recall) has been affected. This indicates that the removed features had a significant impact on the model's predictive capabilities.

Class-wise Performance

The model shows high precision across most classes but varying levels of recall, indicating challenges in correctly identifying all instances in certain classes.

Classes such as 'alcohol', 'new driver', and 'none' perform well in terms of precision and recall.

'Listening to loud music', 'tiredness', and 'unstable mind' have significantly lower recall rates, suggesting difficulty in correctly identifying these classes.

The 'micro avg', 'macro avg', 'weighted avg', and 'samples avg' scores reflect a good overall balance of precision and recall, though they indicate a dip in recall. **See Table 3 in appendix A for classification report.**

Iteartion 3

Code Snippet of Feature Engineering

```
multi_label_model = model_config(
    n_e=200,
    max_d=15,
    min_sa_sp=10,
    min_sa_leaf=3,
    max_feat='sqrt',
    random_st=42,
    class_w=None,
    n_j=-1
)
```

Performance Metrics and Interpretation:

Accuracy: 0.81, Precision: 0.96, Recall: 0.81, F1: 0.88, Hamming: 0.01436, Jaccard: 0.78
Cross-validation scores: [0.81, 0.81, 0.82, 0.8108343 0.80], Mean CV score: 0.81 Standard deviation of CV scores: 0.00399

In the this iteration, the model was finetuned with specific configurations as shown in the code above, to address the challenges identified in the previous iteration. However, this iteration reveals a decrease in some performance metrics, particularly in recall as shown above. Despite maintaining high precision, the model with this configuration faces a notable decrease in recall and accuracy. This suggests that the model's ability to correctly identify all relevant instances has been compromised, possibly due to the finetuning adjustments made. Future iterations are focused on balancing the high precision with improved recall, ensuring that relevant instances are not overlooked

Class-wise Performance

The precision remains high for most classes, but there is a significant drop in recall across several classes, affecting the overall effectiveness.

Some classes like 'new driver' and 'tricycle drivers' still perform well in both precision and recall.

Classes like 'listening to loud music' and 'unstable mind' have particularly low recall, indicating a challenge in correctly identifying instances of these classes.

The 'micro avg', 'macro avg', 'weighted avg', and 'samples avg' scores show a notable decrease in recall, leading to lower F1-scores.

. **See Table 4 in appendix A for classification report.**

Iteration 4

Code Snippet for Model configuration

```
multi_label_model = model_config_det(
    n_e=300,
    max_d=20,
    min_sa_sp=13,
    min_sa_leaf=1,
    max_feat='sqrt',
    random_st=1,
    class_w='balanced',
    n_j=-1
)
```

Performance Metrics and Interpretation:

Accuracy: 0.83, Precision: 0.86, recall: 0.92, F1: 0.89, Hamming: 0.01457, Jaccard: 0.80
Cross-validation scores: [0.83, 0.83, 0.84, 0.83, 0.83], Mean CV score: 0.83 Standard deviation of CV scores: 0.00449

In this iteration, the model was configured with specific adjustments as shown above. The iteration is aimed to balance precision with improved recall. It presents a more balanced model, with a significant improvement in recall and a satisfactory level of precision. This iteration has successfully addressed the issue of missed relevant instances from the previous iteration while maintaining a high degree of accuracy. The trade-off between precision and recall seems more balanced, making the model more reliable for practical applications.

Class-wise Performance

The model's precision and recall vary across different classes. While precision has decreased in some classes, recall has generally improved, indicating better identification of positive instances.

Classes like ‘alcohol’, 'new driver' and 'none' show high performance in both precision and recall.

Some classes, such as 'busy mind', 'excitement', and 'unstable mind', exhibit lower precision but significantly higher recall, suggesting the model's improved sensitivity in these categories.

'Micro avg', 'macro avg', 'weighted avg', and 'samples avg' scores show an overall improvement, especially in recall, indicating that the model is now better at identifying relevant instances across all classes.

See Table 5 in appendix A for classification report.

7.0 Final Model Selection

Based on the Performance benchmarks of the metrics used to evaluate each iteration of the model’s configurations. I think the model configuration in Iteration 4 should be used for integration with our app, based on its improved sensitivity and overall improvement in all metrics, and also because it appears to be more balanced.

Appendix A

Table 1. Base Model Classification Report

Class	Precision	Recall	F1-Score	Support
Alcohol	0.96	0.97	0.96	4482
Bad Road	0.89	0.87	0.88	1303
Busy Mind	0.92	0.93	0.93	674
Deasiness	0.94	1.00	0.96	247
Distraction	0.90	0.89	0.89	1346
Excitement	0.96	0.98	0.97	213
Fatigue	0.93	0.91	0.92	524
Fuel Queue	0.87	0.84	0.86	290
Listening to Loud Music	0.75	0.76	0.75	54
New Driver	0.99	0.99	0.99	12568

None	0.96	0.96	0.96	6699
Overloading	0.91	0.89	0.90	780
Phone	0.95	0.94	0.95	2608
Rash	0.89	0.84	0.87	512
Tiredness	0.73	0.80	0.76	86
Traffic	0.92	0.93	0.92	775
Tricycle Drivers	0.90	0.91	0.90	215
Unstable Mind	0.90	0.93	0.92	146
Micro Avg	0.96	0.96	0.96	33522
Macro Avg	0.90	0.91	0.91	33522
Weighted Avg	0.96	0.96	0.96	33522
Samples Avg	0.95	0.96	0.95	33522

Table 2 : Iteration 1 Classification Report

Class	Precision	Recall	F1-Score	Support
Alcohol	1.00	0.99	0.99	4482
Bad Road	1.00	0.98	0.99	1303
Busy Mind	1.00	0.98	0.99	674
Deasiness	1.00	0.99	0.99	247
Distraction	1.00	0.97	0.99	1346
Excitement	1.00	1.00	1.00	213
Fatigue	1.00	0.97	0.98	524
Fuel Queue	1.00	0.95	0.98	290
Listening to Loud Music	1.00	0.81	0.90	54
New Driver	1.00	1.00	1.00	12568
None	1.00	0.99	1.00	6699
Overloading	1.00	0.96	0.98	780
Phone	1.00	0.98	0.99	2608
Rash	1.00	0.97	0.99	512
Tiredness	1.00	0.95	0.98	86
Traffic	1.00	0.97	0.99	775
Tricycle Drivers	1.00	0.97	0.98	215
Unstable Mind	1.00	0.99	0.99	146
Micro Avg	1.00	0.99	0.99	33522
Macro Avg	1.00	0.97	0.98	33522
Weighted Avg	1.00	0.99	0.99	33522
Samples Avg	0.99	0.99	0.99	33522

Table 3 : Iteration 2 Classification Report

Class	Precision	Recall	F1-Score	Support
Alcohol	0.96	0.83	0.90	4482
Bad Road	0.91	0.66	0.77	1303
Busy Mind	0.92	0.61	0.74	674
Deasiness	1.00	0.77	0.87	247
Distraction	0.99	0.74	0.85	1346
Excitement	0.93	0.74	0.83	213
Fatigue	0.96	0.70	0.81	524
Fuel Queue	0.98	0.62	0.76	290
Listening to Loud Music	0.96	0.41	0.57	54
New Driver	0.99	0.97	0.98	12568
None	0.95	0.87	0.91	6699
Overloading	0.95	0.64	0.76	780
Phone	0.96	0.78	0.86	2608
Rash	0.95	0.50	0.66	512
Tiredness	1.00	0.38	0.55	86
Traffic	0.93	0.66	0.77	775
Tricycle Drivers	0.98	0.92	0.95	215
Unstable Mind	0.97	0.25	0.40	146
Micro Avg	0.97	0.85	0.91	33522
Macro Avg	0.96	0.67	0.77	33522
Weighted Avg	0.97	0.85	0.90	33522
Samples Avg	0.87	0.87	0.87	33522

Table 4 : Iteration 3 Classification Report

Class	Precision	Recall	F1-Score	Support
Alcohol	0.96	0.73	0.83	4482
Bad Road	0.91	0.59	0.72	1303
Busy Mind	0.90	0.55	0.68	674
Deasiness	1.00	0.70	0.83	247
Distraction	0.99	0.59	0.74	1346
Excitement	0.96	0.69	0.81	213
Fatigue	0.95	0.65	0.77	524
Fuel Queue	0.99	0.57	0.72	290
Listening to Loud Music	1.00	0.19	0.31	54
New Driver	0.98	0.96	0.97	12568
None	0.93	0.85	0.89	6699
Overloading	0.96	0.55	0.70	780
Phone	0.96	0.72	0.83	2608
Rash	0.96	0.37	0.54	512

Tiredness	1.00	0.31	0.48	86
Traffic	0.93	0.59	0.72	775
Tricycle Drivers	0.99	0.90	0.94	215
Unstable Mind	1.00	0.11	0.20	146
Micro Avg	0.96	0.80	0.88	33522
Macro Avg	0.97	0.59	0.70	33522
Weighted Avg	0.96	0.80	0.87	33522
Samples Avg	0.83	0.82	0.83	33522

Table 5 : Iteration 4 Classification Report

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Alcohol	0.87	0.92	0.90	4482
Bad Road	0.69	0.86	0.77	1303
Busy Mind	0.58	0.88	0.70	674
Deasiness	0.90	0.88	0.89	247
Distraction	0.77	0.87	0.81	1346
Excitement	0.59	0.86	0.70	213
Fatigue	0.65	0.86	0.74	524
Fuel Queue	0.65	0.82	0.73	290
Listening to Loud Music	0.63	0.59	0.61	54
New Driver	0.98	0.97	0.98	12568
None	0.88	0.92	0.90	6699
Overloading	0.73	0.82	0.77	780
Phone	0.83	0.88	0.85	2608
Rash	0.68	0.76	0.72	512
Tiredness	0.69	0.69	0.69	86
Traffic	0.62	0.88	0.73	775
Tricycle Drivers	0.96	0.93	0.94	215
Unstable Mind	0.33	0.89	0.48	146
Micro Avg	0.86	0.92	0.89	33522
Macro Avg	0.72	0.85	0.77	33522
Weighted Avg	0.87	0.92	0.89	33522
Samples Avg	0.88	0.93	0.90	33522