National College of Ireland

Configuration Manual

MSc Research Project
MSc in Data Analytics

Subramanyam Dhandapani
Student ID: x22245421

School of Computing
National College of Ireland

Supervisor: Anderson Simiscuka

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Subramanyam Dhandapani
Student ID:	x22245421
Programme:	MSc in Data Analytics
Year:	2023-2024
Module:	MSc Research Project
Supervisor:	Anderson Simiscuka
Submission Due Date:	12/08/2024
Project Title:	Configuration Manual
Word Count:	678
Page Count:	11

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Subramanyam Dhandapani
Date:	12th August 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Subramanyam Dhandapani
x22245421

1 Introduction

The introduction of configuration manual consists of system requirements, environmental setup, tools and libraries used, machine learning algorithms used in predicting the mortality of person in motor collision. In this research both the ensemble and stacking approaches are used. Evaluation metrics such as precision, recall, accuracy and f1 score are used for assessing the model's performance.

2 System Specification

This section contains all the system specifications

2.1 Hardware Specifications

- Processor: AMD Ryzen 7 5800HS 3.20 GHz
- Installed RAM: 16.0 GB (15.4 GB usable)

2.2 Software Specifications

- Operating System: Windows 11 64-bit
- Python: Python programming language of version 3.11.4 is used for data preprocessing and model building.

```
3.11.4 | packaged by Anaconda, Inc. | (main, Jul  5 2023, 13:38:37) [MSC v.1916 64 bit (AMD64)]
```

Figure 1: Python Version

- Jupyter Notebook: Jupyter notebook is an open source computing environment used for developing codes and visualizations. The version 6.5.7 is used for carrying out this research. The code runs on anaconda navigator (anaconda 2.4.3)

Server Information:

You are using Jupyter Notebook.

The version of the notebook server is: **6.5.7**
The server is running on this version of Python:

```
Python 3.11.4 | packaged by Anaconda, Inc. | (main, Jul  5 2023, 13:38:37) [MSC v.1916 64
bit (AMD64)]
```

Figure 2: Jupyter Notebook Version

3 Data Source

The dataset is downloaded from new york government website. There are two datasets crashes which contains the data crash and the other one is person which contains the information about the person involved in the collision.

BOROUGH	NUMBER OF PERSONS INJURED	NUMBER OF PERSONS KILLED	CONTRIBUTING FACTOR VEHICLE 1	CONTRIBUTING FACTOR VEHICLE 2	PERSON_INJURY	PERSON_AGE	EJECTION	EMOTIONAL_STATUS	BODILY_INJURY
BRONX	2.0	0.0	Unspecified	Driver Inattention/Distraction	Injured	41.0	Not Ejected	Shock	Chest
BRONX	2.0	0.0	Unspecified	Driver Inattention/Distraction	Injured	20.0	Not Ejected	Shock	Head
MANHATTAN	0.0	0.0	Passing Too Closely	Driver Inattention/Distraction	Alive	22.0	Not Ejected	Does Not Apply	Back
QUEENS	0.0	0.0	Turning Improperly	Driver Inattention/Distraction	Alive	25.0	Not Ejected	Does Not Apply	Back
QUEENS	2.0	0.0	Reaction to Uninvolved Vehicle	Driver Inattention/Distraction	Injured	46.0	Not Ejected	Conscious	Face

Figure 3: Dataframe Overview

- Crashes Dataset: https://catalog.data.gov/dataset/motor-vehicle-collisions-crashes
- Person Dataset: https://catalog.data.gov/dataset/motor-vehicle-collisions-person

3.1 Data Cleaning and Analysis

Necessary libraries are imported in the jupyter notebook. The below figure shows the necessary libraries imported.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
from sklearn.metrics import roc_curve, auc, roc_auc_score
from sklearn.preprocessing import label_binarize
from sklearn.metrics import precision_recall_curve, average_precision_score
from sklearn.metrics import RocCurveDisplay
```

Figure 4: Libraries Imported

```
from sklearn.ensemble import VotingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

Figure 5: Voting Classifier and other libraries for assessing the model

```
from sklearn.ensemble import StackingClassifier
```

Figure 6: Stacking Library

The below figure is the filtered dataframe after the process of data cleaning by removing null values and outliers in the dataset.

```
merged_df_filtered.dtypes
```

```
CONTRIBUTING FACTOR VEHICLE 1     object
CONTRIBUTING FACTOR VEHICLE 2     object
PERSON_INJURY                     object
PERSON_AGE                       float64
EJECTION                          object
EMOTIONAL_STATUS                  object
BODILY_INJURY                     object
POSITION_IN_VEHICLE               object
SAFETY_EQUIPMENT                  object
dtype: object
```

Figure 7: Filtered Columns

The outliers in the person age column is handled using the interquartile range. The below figure shows the code for iterquartile range.

3.2 Exploratory Data Analysis

This section explains about the exploratory data analysis where a number of count plots have been plotted against the independent and the target variables.

```
Q1 = merged_df['PERSON_AGE'].quantile(0.25)
Q3 = merged_df['PERSON_AGE'].quantile(0.75)

# Computing the IQR
IQR = Q3 - Q1

# Determining the outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR


outliers = merged_df[(merged_df['PERSON_AGE'] < lower_bound) | (merged_df['PERSON_AGE'] > upper_bound)]
print("Outliers in the 'PERSON_AGE' column:")
print(outliers[['PERSON_AGE']])

merged_df_cleaned = merged_df[(merged_df['PERSON_AGE'] >= 1) & (merged_df['PERSON_AGE'] <= upper_bound)]
print("\nDataFrame after removing outliers:")
print(merged_df_cleaned)
```

Figure 8: Removal of outliers from age column

```
plt.figure(figsize=(10, 6))
sns.countplot(data=merged_df_cleaned, x='BOROUGH', hue='PERSON_INJURY')
plt.title('Count of Injuries by Borough')
plt.xlabel('Borough')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.legend(title='Person Injury')
plt.tight_layout()
plt.show()
```

Figure 9: Count plot of collisions per borough

```
plt.figure(figsize=(10, 6))
sns.histplot(merged_df_cleaned['PERSON_AGE'], kde=True)
plt.title('Distribution of Person Age')
plt.xlabel('Person Age')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```

Figure 10: Distribution of the age column

```
plt.figure(figsize=(10, 12))
sns.countplot(data=merged_df_cleaned, y='BODILY_INJURY')
plt.title('Number of Persons Injured/Killed by Contributing Factor')
plt.xlabel('Count')
plt.ylabel('Contributing Factor')
plt.show()
```

Figure 11: Bar plot of bodily injured column

4 Data Preprocessing

This section contains the steps involved in data preprocessing such as feature engineering techniques, label encoding techniques and oversampling techniques (SMOTE).

```
#feature engineering derving a new feature from the date column

merged_df['CRASH_DATE'] = pd.to_datetime(merged_df['CRASH_DATE'])

merged_df['CRASH_TIME'] = pd.to_datetime(merged_df['CRASH_TIME'],format='%H:%M')

merged_df['Year'] = merged_df['CRASH_DATE'].dt.year
merged_df['Day'] = merged_df['CRASH_DATE'].dt.strftime('%A')
```

Figure 12: Extracting year using feature engineering

Label encoding is done to all the categorical columns of the dataset. The age column is excluded since that is a numerical column.

```
df = pd.DataFrame(merged_df_filtered)

label_encoders={}

# Encode categorical variables
for column in df:
    if column != 'PERSON_AGE':
        le = LabelEncoder()
        df[column] = le.fit_transform(df[column])
        label_encoders[column] = le
```

Figure 13: Applying Label encoding

```
smote = SMOTE(sampling_strategy=sampling_strategy, random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
```

Figure 14: Applying SMOTE

5 Model Building

This has the overview of the models and machine learning approaches used in this research. Before model building the dataset is split into X and y where X has the independent variables and y contains target variable. The train and test is split with a ratio of 70:30.

```
# Split data into features and labels
X = df.drop('PERSON_INJURY', axis=1)
y = df['PERSON_INJURY']

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, stratify=y,random_state=42)
```

Figure 15: Splitting the data to train and test

The machine learning approaches used in this research are ensemble and stacking by combination of shuffling the base models in it.

```
# Defining individual classifiers
knn = KNeighborsClassifier(n_neighbors=5)
rf = RandomForestClassifier(n_estimators=100, random_state=42)
dt = DecisionTreeClassifier(random_state=42)

# Create the ensemble model
ensemble = VotingClassifier(estimators=[
    ('knn', knn),
    ('rf', rf),
    ('dt', dt)
], voting='hard')

# Train the ensemble model
ensemble.fit(X_resampled, y_resampled)

# Make predictions
y_pred_ensemble = ensemble.predict(X_test)


# Calculate accuracy
accuracy_ensemble = accuracy_score(y_test, y_pred_ensemble)
print(f'Ensemble Accuracy: {accuracy_ensemble}')
```

Figure 16: Voting classifier using all the base models

```
ensemble_1 = VotingClassifier(estimators=[
    ('knn', knn),
    ('rf', rf)
], voting='hard')

# Train the ensemble model
ensemble_1.fit(X_resampled, y_resampled)

# Make predictions
y_pred_ensemble_1 = ensemble_1.predict(X_test)

# Calculate accuracy
accuracy_ensemble_1 = accuracy_score(y_test, y_pred_ensemble_1)
print(f'Ensemble Accuracy: {accuracy_ensemble_1}')
```

Figure 17: Voting classifier with KNN and random forest

```
ensemble_3 = VotingClassifier(estimators=[
    ('rf', rf),
    ('dt', dt)
], voting='hard')

# Train the ensemble model
ensemble_3.fit(X_resampled, y_resampled)

# Make predictions
y_pred_ensemble_3 = ensemble_3.predict(X_test)

# Calculate accuracy
accuracy_ensemble_3 = accuracy_score(y_test, y_pred_ensemble_3)
print(f'Ensemble Accuracy: {accuracy_ensemble_3}')
```

Figure 18: Voting classifier with random forest and decision tree

```
base_models = [
    ('dt', DecisionTreeClassifier(random_state=42))
]

# Define the meta-learner as Decision Tree
meta_learner_combination4 =  RandomForestClassifier(n_estimators=100, random_state=42)


stacking_clf_combination4 = StackingClassifier(estimators=base_models, final_estimator=meta_learner_combination4)

stacking_clf_combination4.fit(X_resampled, y_resampled)

# Make predictions
y_pred_stacking_combination4 = stacking_clf_combination4.predict(X_test)

# Calculate accuracy
accuracy_stacking_combination4 = accuracy_score(y_test, y_pred_stacking_combination4)
print(f'Stacking Classifier with Decision Tree Meta-Learner Accuracy: {accuracy_stacking_combination4}')
```

Figure 19: Stacking with meta learner as random forest and base learner as decision tree

```
# Defining the base models
base_models = [
    ('rf', RandomForestClassifier(n_estimators=100, random_state=42))
]

# Defining the meta-learner
meta_learner_combination5 = KNeighborsClassifier(n_neighbors=10)

stacking_clf_combination5 = StackingClassifier(estimators=base_models, final_estimator=meta_learner_combination5)


stacking_clf_combination5.fit(X_resampled, y_resampled)


y_pred_stacking_combination5 = stacking_clf_combination5.predict(X_test)

# Calculating accuracy
accuracy_stacking_combination5 = accuracy_score(y_test, y_pred_stacking_combination5)
print(f'Stacking Classifier Accuracy: {accuracy_stacking_combination5}')
```

Figure 20: Stacking with meta learner as KNN and base learner as random forest

```
base_models = [
    ('dt', DecisionTreeClassifier(random_state=42))
]

meta_learner_combination2 = KNeighborsClassifier(n_neighbors=10)

stacking_clf_combination2 = StackingClassifier(estimators=base_models, final_estimator=meta_learner_combination2)

stacking_clf_combination2.fit(X_resampled, y_resampled)

# Make predictions
y_pred_stacking_combination2 = stacking_clf_combination2.predict(X_test)

# Calculating accuracy
accuracy_stacking_combination2 = accuracy_score(y_test, y_pred_stacking_combination2)
print(f'Stacking Classifier with KNN Meta-Learner Accuracy: {accuracy_stacking_combination2}')
```

Figure 21: Stacking with meta learner as KNN and base learner as decision tree

6 Comparison of Evaluation Metrics

This section provides overview of the comparison of the evaluation metrics such as accuracy, precision, recall, f1 score and confusion matrix between the machine learning approaches used in this research.

```
cm = confusion_matrix(y_test, y_pred_ensemble_1)
print(f'Confusion Matrix:\n{cm}')
```

```
#Display classification report
print("\nClassification Report for Ensemble Combination1 Model:")
print(classification_report(y_test, y_pred_ensemble_1))
```

```
Classification Report for Ensemble Combination1 Model:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98    165878
           1       0.98      0.86      0.92     43714
           2       0.39      0.20      0.26       101

    accuracy                           0.97    209693
   macro avg       0.78      0.68      0.72    209693
weighted avg       0.97      0.97      0.97    209693
```

Figure 22: Confusion matrix and classification report of ensemble with random forest and KNN

```
cm = confusion_matrix(y_test, y_pred_ensemble_3)
print(f'Confusion Matrix:\n{cm}')
```

```
#Display classification report
print("\nClassification Report for Ensemble Combination3 Model:")
print(classification_report(y_test, y_pred_ensemble_3))
```

```
Classification Report for Ensemble Combination3 Model:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99    165878
           1       0.97      0.98      0.97     43714
           2       0.76      0.68      0.72       101

    accuracy                           0.99    209693
   macro avg       0.91      0.88      0.90    209693
weighted avg       0.99      0.99      0.99    209693
```

```
# Mapping values to descriptive labels
label_map = {0: 'Alive', 1: 'Injured', 2: 'Killed'}
mapped_y_test_3 = [label_map[val] for val in y_test]
mapped_y_pred_ensemble_3 = [label_map[val] for val in y_pred_ensemble_3]

# Computing confusion matrix
cm = confusion_matrix(mapped_y_test_3, mapped_y_pred_ensemble_3, labels=['Alive', 'Injured', 'Killed'])


plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='d', annot_kws={"size": 12,"ha": 'center', "va": 'center'}, cbar=True)
plt.xlabel('Predicted')
plt.ylabel('Actual')

plt.xticks(ticks=np.arange(3) + 0.5, labels=['Alive', 'Injured', 'Killed'])
plt.yticks(ticks=np.arange(3) + 0.5, labels=['Alive', 'Injured', 'Killed'])

plt.title('Confusion Matrix Heatmap')
plt.show()
```

Figure 23: Evaluation metrics of ensemble with decision tree and random forest

```
# Generate the classification report
classification_rep = classification_report(y_test, y_pred_stacking_combination4)
print(f'Classification Report:\n{classification_rep}')
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99    165878
           1       0.97      0.99      0.98     43714
           2       0.42      0.72      0.53       101

    accuracy                           0.99    209693
   macro avg       0.80      0.90      0.83    209693
weighted avg       0.99      0.99      0.99    209693
```

```
# Record the start time
start_time = time.time()

stacking_clf_combination4.fit(X_resampled, y_resampled)

end_time = time.time()

training_time = end_time - start_time
print(f"Training time: {training_time:.2f} seconds")
```

```
Training time: 12.25 seconds
```

```
# Defining the class labels
class_labels = ['Alive', 'Injured', 'Killed']

cm = confusion_matrix(y_test, y_pred_stacking_combination4)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, cmap='Greens', fmt='d', annot_kws={"size": 12, "ha": 'center', "va": 'center'}, cbar=True)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix Heatmap')
plt.xticks(ticks=range(len(class_labels)), labels=class_labels)
plt.yticks(ticks=range(len(class_labels)), labels=class_labels, rotation=0)
plt.show()
```

Figure 24: Stacking evaluation metrics and training time with base learner as decision tree and random forest as meta learner

```
# Generate the classification report
classification_rep = classification_report(y_test, y_pred_stacking_combination5)
print(f'Classification Report:\n{classification_rep}')
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99    165878
           1       0.97      0.99      0.98     43714
           2       0.83      0.64      0.73       101

    accuracy                           0.99    209693
   macro avg       0.93      0.87      0.90    209693
weighted avg       0.99      0.99      0.99    209693
```

```
# Plot feature importances
plt.figure(figsize=(10, 6))
plt.barh(importance_df['Feature'], importance_df['Importance'], color='green')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.title('Base Model Feature Random Forest Feature Importance')
plt.gca().invert_yaxis()
plt.show()
```

Figure 25: Feature importance of random forest as base learner and classification report with base learner as random forest and meta learner as KNN