National College of Ireland

Predictive Analysis of Road Accidents in India: A Machine Learning Approach Configuration Manual

MSc Research Project
Data Analytics

Harini Manjunatha
Student ID: x22169288

School of Computing
National College of Ireland

Supervisor: Shubham Subhnil

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Harini Manjunatha
Student ID:	x22169288
Programme:	Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Shubham Subhnil
Submission Due Date:	14/12/2023
Project Title:	Predictive Analysis of Road Accidents in India: A Machine Learning Approach
Word Count:	679
Page Count:	6

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:	Harini Manjunatha
Date:	31st January 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):	

Predictive Analysis of Road Accidents in India: A Machine Learning Approach - Configuration Manual

Harini Manjunatha
x22169288

1 Introduction

The configuration manual contains the detailed information on setting up system. The objective of this manual is to outline the process on conducting the study 'Predictive Analysis of Road Accidents in India: A Machine Learning Approach'. It specifies the necessary machine setup needed to create and execute the models. The instructions cover installing the required programmes and packages in addition to setting up the fundamental configuration that is necessary for the project to succeed.

2 System Specification

In this section, the hardware and software requirements used for the research is presented.

2.1 Hardware specification

Operating System	Mac OS
RAM	8 GB
System processor	Apple M2 (8-core)
Speed in Hz	3.49 G HZ
GPU	M2 10-Core GPU 13.5 W 1398 MHz

Figure 1: Hardware specification

2.2 Software specification

Programming Language	Python 3.9
Softwares	Anaconda and Jupyter notebook
Web browser	Safari

Figure 2: Software specification

3 Software Used

- Microsoft excel: Used for initial exploration of data.
- Jupyter Notebook [1]: Jupyter Notebook provides an interactive environment where we execute code in cells and visualize the results. It makes process easier to experiment with different machine learning models, algorithms, visualising the data and data preprocessing techniques.
- smartdraw [2] : Online platform used for creating flowcharts.

4 Installation

The First step in the process involves launching the Anaconda application. The application provides an extensive set of software designed to perform diverse needs. Jupyter notebook is launched which offers an envirnoment where we can execute the code in cells and visualise the results. Numerous exceptional Python libraries are available for comprehensive data analysis purposes. These libraries enhance the capability to explore, process, and derive insights from datasets efficiently. Figure 3 shows the interface of an Anaconda navigator.

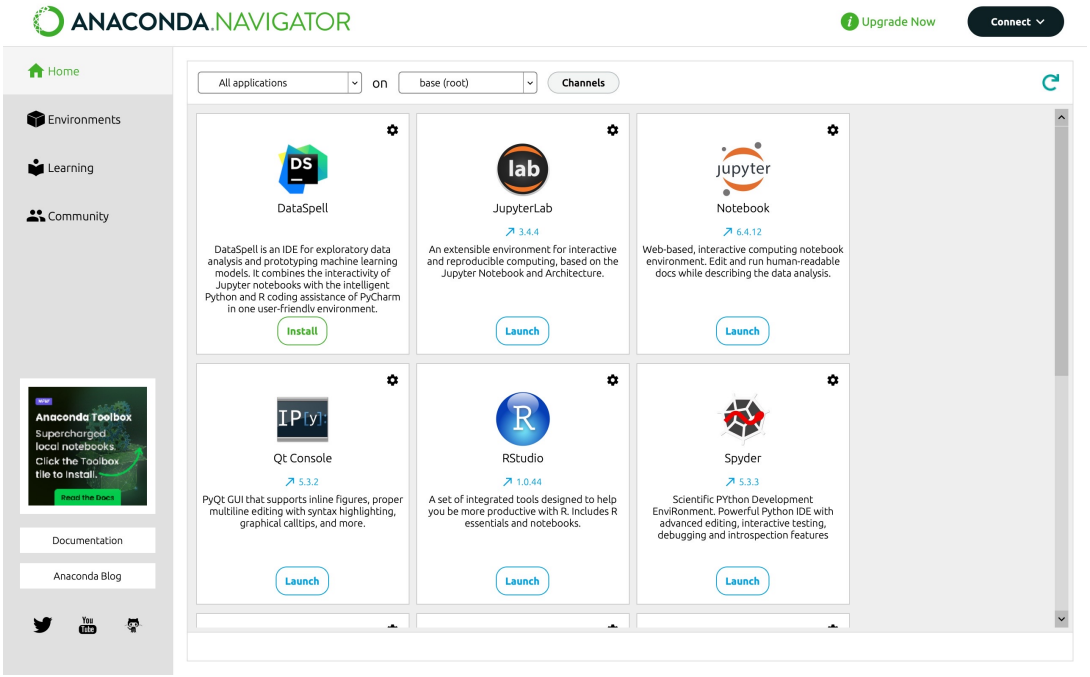

Figure 3: Anaconda Interface

Figure 4 is the main page of the jupyter notebook. The new ipynb file is created to begin coding.

Figure 5 represents the code file of the project. The file is executed using the command restart and run all

[1]Jupyter notebook: `https://jupyter.org`

[2]SmartDraw: `https://cloud.smartdraw.com//?flags=129`

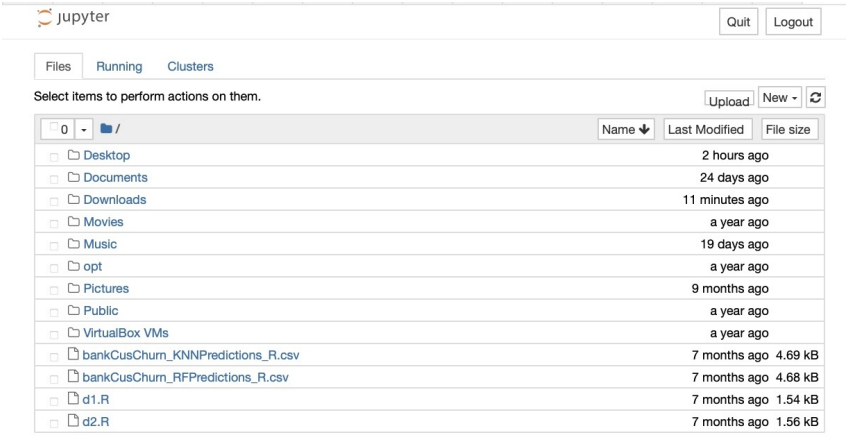

Figure 4: Jupyter Notebook

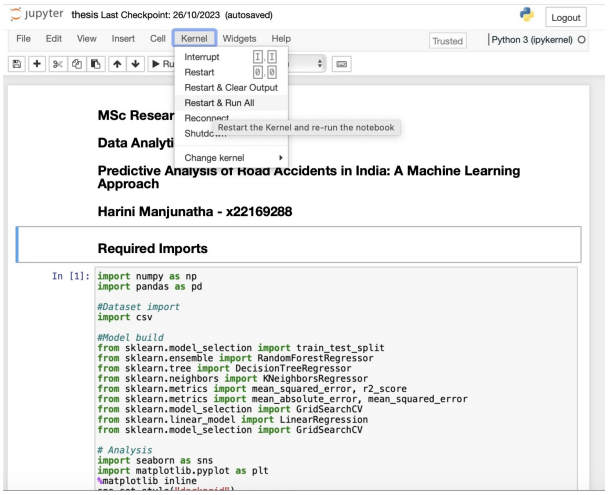

Figure 5: Code File on jupyter notebook

5 Project Development

5.1 Importing necessary library

Figure 6 shows the list of all necessary library imports used in the project. The numpy [3] and pandas [4] libraries enable efficient manipulation and analysis of data through functionalities like arrays, matrices, and data structures like DataFrames. Scikit-learn modules [5], regression models like RandomForestRegressor, DecisionTreeRegressor, LinearRegression and GridSearchCV are imported to construct, evaluate, and tune machine learning models.

[3]Numpy: `https://numpy.org`
[4]Pandas: `https://pandas.pydata.org`
[5]Scikit-learn: `https://scikit-learn.org/stable/`

Required Imports

In [1]:
```
import numpy as np
import pandas as pd

#Dataset import
import csv

#Model build
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import GridSearchCV

# Analysis
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set_style("darkgrid")
import seaborn as sns
import plotly.graph_objs as go
import plotly.express as px
sns.set()
import plotly.express as px
from tabulate import tabulate
```

Figure 6: Library Imports]

5.2 Import dataset

Figure 7 is the code snippet used for importing the dataset.

Importing dataset of state wise road accident based on time occurance between the year 2001 - 2021

In [2]:
```
df=pd.read_csv('/Users/harinim/Downloads/new.csv')
```

Figure 7: Dataset Import

5.3 Model Building

The dataset is spli into two sets. One for training and another for testing. Data from the year 2001 to 2020 is used for training whereas data of 2021 is used for testing. The code snippet which splits the data is shown in the Figure 8

In [ ]:
```
train_data = df[df['Year'] <= 2020][['Year', 'State']]
train_labels = df[df['Year'] <= 2020][['6-9', '9-12', '12-15','15-18','18-21','21-24',
                                       '0-3','3-6']]  # Multiple target variables

test_data = df[df['Year'] == 2021][['Year', 'State']]
test_labels = df[df['Year'] == 2021][['6-9', '9-12', '12-15','15-18','18-21','21-24',
                                      '0-3','3-6']]  # Multiple target variables
```

Figure 8: Dataset Splitting

The code snippet in Figure 9 identify and eliminate outliers from the dataset using the z-score method.

Then code snippet in the Figure 10 implements a Random Forest regression model using scikit-learn's RandomForestRegressor [6]. It initiates a grid search for the best hyperparameters using GridSearchCV [7] to optimize the model's performance. The code eval-

[6]Scikit-learn: `https://scikit-learn.org/stable/`

[7]Grid search: `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`

```
In [ ]: # Outlier detection
        numeric_columns = df.select_dtypes(include=[np.number]).columns.tolist()
        numeric_columns.remove('Year')

        # Outlier detection using z-score method for each numeric column
        outlier_threshold = 3
        outlier_indices = []

        for col in numeric_columns:
            mean = df[col].mean()
            std_dev = df[col].std()
            outliers = df[(df[col] - mean).abs() > outlier_threshold * std_dev]
            outlier_indices.extend(outliers.index.tolist())

        unique_outlier_indices = list(set(outlier_indices))


        df_outliers = df.drop(unique_outlier_indices)


        print("Indices of Detected Outliers:", unique_outlier_indices)
```

Figure 9: Outlier detection

uates the model's performance metrics using Mean Absolute Error (MAE), R-squared, Root Mean Squared Error (RMSE) for each target variable in test set.

```
# Model 1 - Random forest

# Define the parameter grid to search
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20,15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Initialize the Random Forest Regressor
rf_regressor = RandomForestRegressor(random_state=42)

# Initialize Grid Search Cross Validation
grid_search = GridSearchCV(estimator=rf_regressor, param_grid=param_grid,
                           scoring='neg_mean_squared_error', cv=5, verbose=1, n_jobs=-1

# Perform the grid search
grid_search.fit(train_data, train_labels)

best_params = grid_search.best_params_
print(f'Best Hyperparameters: {best_params}')

best_rf_regressor = grid_search.best_estimator_

predictions = best_rf_regressor.predict(test_data)

results = []

# Validate the model
for i, col in enumerate(train_labels.columns):
    mae = mean_absolute_error(test_labels[col], predictions[:, i])
    mse = mean_squared_error(test_labels[col], predictions[:, i])
    r2 = r2_score(test_labels[col], predictions[:, i])
    rmse = np.sqrt(mse)

    # Append the results to the list
    results.append({
        'Time': col,
        'MAE': mae,
        'R-squared': r2,
        'RMSE': rmse
    })

# Create a DataFrame from the results list
results_df = pd.DataFrame(results)

print(tabulate(results_df, headers='keys', tablefmt='fancy_grid'))
```

Figure 10: Random Forest Regressor Model

Figure 11 visualizes the distribution of differences between the actual and predicted values from the regression model. Normal distribution plot provides an immediate understanding of the spread and density of prediction errors. The inclusion of vertical lines denoting the mean difference and standard deviations aids in interpreting the central tendency and variability of these prediction errors, offering valuable insights into the model's performance and the nature of discrepancies between actual and predicted values.

The Predicted output from the best performing model is visualised to understand the data pattern. The code snippet of the visualisation is given in Figure 12

```python
# Calculate the differences between actual and predicted values
differences = test_labels.values - predictions

# Flatten the differences array for plotting
differences_flat = differences.flatten()

# Plot the normal distribution
plt.figure(figsize=(12, 6))
sns.histplot(differences_flat, kde=True, color='blue', stat='density')
plt.title('Difference between Actual and Predicted Values')
plt.xlabel('Difference')
plt.ylabel('Density')

# Add a vertical line at mean and standard deviations
mean_diff = np.mean(differences_flat)
std_dev_diff = np.std(differences_flat)
plt.axvline(mean_diff, color='red', linestyle='dashed', linewidth=1)
plt.axvline(mean_diff + std_dev_diff, color='green', linestyle='dashed', linewidth=1)
plt.axvline(mean_diff - std_dev_diff, color='green', linestyle='dashed', linewidth=1)

# Add legend and show plot
plt.legend(['Mean', 'Mean + 1 Std Dev', 'Mean - 1 Std Dev'])
plt.show()
```

Figure 11: Normal distribution

In [ ]:
```python
# Parallel Coordinates Plot for State-wise Accident - 2022
plt.figure(figsize=(10, 6))
sns.set_theme(style="whitegrid")
sns.lineplot(data=df_regrouped.T, palette="tab10", linewidth=3)
plt.xticks(rotation=90)
plt.title('State-wise Accident - 2022')
plt.xlabel('Time Intervals')
plt.ylabel('Accident Count')
plt.legend(title='States', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

In [ ]:
```python
# Area Plot for each state's data
for state in df_regrouped.index:
    plt.figure(figsize=(10, 6))
    plt.fill_between(df_regrouped.columns, df_regrouped.loc[state])
    plt.title(f'Data for {state}')
    plt.xlabel('Time Period')
    plt.ylabel('Value')
    plt.show()
```

In [ ]:
```python

# Calculating total accidents for each state in the year 2022
accidents_2022 = df_regrouped.sum(axis=1)

# Sorting states based on total accidents in descending order
sorted_states = accidents_2022.sort_values(ascending=False)

# Selecting top 10 states with the highest accident rate
top_states = sorted_states.head(10)

# Selecting bottom 10 states with the lowest accident rate
bottom_states = sorted_states.tail(10)

# Pie chart for top 10 states with highest accident rate
plt.figure(figsize=(10, 8))
plt.pie(top_states, labels=top_states.index, autopct='%1.1f%%', startangle=140)
plt.title('Top 10 States with Highest Accident Rate in 2022')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
plt.show()

# Pie chart for bottom 10 states with lowest accident rate
plt.figure(figsize=(10, 8))
plt.pie(bottom_states, labels=bottom_states.index, autopct='%1.1f%%', startangle=140)
plt.title('Bottom 10 States with Lowest Accident Rate in 2022')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
plt.show()
```

Figure 12: Predicted result Visualisation