Configuration Manual

Automotive Market Trend Prediction
and the Adoption of EV Technologies

Arun Das Mohandas

Student ID: x23136766

School of Computing

National College of Ireland

Supervisor: Mr. Hicham Rifai

National College of Ireland

MSc Data Analytics Project Submission Sheet

School of Computing

Student Name:	Arun Das Mohandas
Student ID:	x23136766
Programme:	M.sc Data Analytics **Year:** 2024
Module:	Research In computing
Lecturer:	Mr. Hicham Rifa
Submission Due Date:	12-12-2024
Project Title:	Automotive Market Trend Prediction and Adoption of EV Technologies
Word Count:	935 **Page Count:** 8

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 another author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Arun Das M

Date: 12-12-2024

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):	

1 Introduction

The presented configuration manual displays the demonstration of the required hardware and software with details on the execution for the application on *Automotive Market Trend Prediction and Adoption of EV Technologies.*

2 System Configuration

2.1 Hardware Configuration

Table 1 Hardware Config illustrates the hardware configuration manual with a 16-core system and 16 GB RAM.

Table 1 Hardware Config

Component	*Specification*
Processor	Intel Core i7-11800H (16 Cores, 20 Threads, 2.30 GHz Base, 4.0 GHz Turbo)
Graphics Card	NVIDIA RTX 3060 Ti (6 GB GDDR6X)
RAM	16 GB DDR5 (3200 MHz)
Storage	500 GB NVMe SSD
Operating System	Windows 11 Home
Motherboard	MSI KATANA GF76 11UE
Power Supply Unit	850W Gold Certified PSU
Cooling System	Turbo Air Cooler

2.2 Software Configuration

For the implementation of the project, the following software was relied upon:

- Anaconda 23.7.4
- CMD Prompt V0.1.1
- Jupiter Notebook V6.5.4
- Microsoft Excel

The above software was installed along with the following libraries:

Table 2 Libraries

LIBRARY	VERSION
PYTHON	3.12.0 or higher
NUMPY	1.26.4
PANDAS	2.2.2
MATPLOTLIB	3.7.2
SEABORN	0.12.2
PLOTLY	5.9.0
STATSMODEL	0.14.0
PROPHET	24
SCIKIT-LEARN	1.5.0
TENSORFLOW	2.16.1
KERAS TUNER	1.4.7

2.3 Environment Setup

This section will provide details on the installed software starting with the Anaconda. The Anaconda version 23.7.4 is installed with Anaconda Navigator 2.6.4 or higher installed. A new Anaconda environment is created along with the packages. Additionally, software such as Jupiter Notebook Version 6.5.4 for Python operations and CMD version 0.1.1 is installed. Python version 3.12.0 was installed on the environment.

2.4 Data Selection

The dataset for the project was opted from Kaggle https://www.kaggle.com/datasets/sahirmaharajj/electric-vehicle-population-size-2024?resource=download. The dataset contains details on the county-wise number of Vehicles sold in the United States of America in various categories. These categories are Non-Electric Vehicles, Battery Operated Electric Vehicles, and Plug-in Hybrid Electric Vehicles. The Dataset can be downloaded as a .csv file which will be used in the implementation.

3 Task Implementation and Execution

3.1 Packages and Libraries to Import

Given below is the list of libraries required for the current implementation in Figure 1 below.

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import plotly.express as px
import warnings
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from prophet import Prophet
from prophet.diagnostics import cross_validation, performance_metrics
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller, kpss
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import TimeSeriesSplit
from statsmodels.tsa.statespace.sarimax import SARIMAX
import xgboost as xgb
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.optimizers import Adam, RMSprop, Nadam
from keras_tuner import Hyperband
import math
```

Fig 1. Libraries loaded

3.2 Data Preparation and Preprocessing

The code below is executed to process and transform the data to prepare it for time series modeling.

```python
DF_EV_sales = DF_EV_sales.dropna(subset=['County', 'State'])

# Verify the removal of null values
print("\nMissing Values after removing rows with nulls in 'County' and 'State':")
print(DF_EV_sales.isnull().sum())
```

```python
# Step 1: Convert the 'Date' column to datetime format
DF_EV_sales['Date'] = pd.to_datetime(DF_EV_sales['Date'])

# Step 2: Create separate columns for day, month, and year
DF_EV_sales['Day'] = DF_EV_sales['Date'].dt.day
DF_EV_sales['Month'] = DF_EV_sales['Date'].dt.month
DF_EV_sales['Year'] = DF_EV_sales['Date'].dt.year

# Step 2: Reconstruct the full date using 'Year', 'Month', and 'Day'
DF_EV_sales['Recon_Date'] = pd.to_datetime(DF_EV_sales[['Year', 'Month', 'Day']])

# Step 3: Drop the 'Date' column
DF_EV_sales = DF_EV_sales.drop(columns=['Date'])

# Display the first few rows to verify
print(DF_EV_sales.head())
```

```python
# Step 3: Reshape the data into separate rows for BEVs, PHEVs, and Non-Electric vehicles
columns_to_melt = ['Battery Electric Vehicles (BEVs)', 'Plug-In Hybrid Electric Vehicles (PHEVs)', 'Non-Electric Vehicle Total']
RE_EV_sales = pd.melt(
    DF_EV_sales,
    id_vars=['Recon_Date', 'County', 'State', 'Vehicle Primary Use', 'Total Vehicles', 'Percent Electric Vehicles'],
    value_vars=columns_to_melt,
    var_name='Vehicle_types',
    value_name='Vehicle Count'
)

# Define the vehicle types
Vehicle_types = ['Battery Electric Vehicles (BEVs)',
                 'Plug-In Hybrid Electric Vehicles (PHEVs)',
                 'Non-Electric Vehicle Total']

# Step 4: Group by the reconstructed date and vehicle type, summing the counts
RE_EV_sales['Vehicle Count'] = RE_EV_sales['Vehicle Count'].replace(',', '', regex=True).astype(float)
grouped_data = RE_EV_sales.groupby(['Recon_Date', 'Vehicle_types'])['Vehicle Count'].sum().unstack().fillna(0)

RE_EV_sales
```

```python
#Additional Data Cleaning Post Data Transformation

# Count the number of rows before filtering
initial_count = len(RE_EV_sales)

# Remove rows with zero or negative values in 'Vehicle Count'
RE_EV_sales = RE_EV_sales[RE_EV_sales['Vehicle Count'] > 0]

# Count the number of rows after filtering
filtered_count = len(RE_EV_sales)

# Calculate the number of rows removed
rows_removed = initial_count - filtered_count

# Display the result
print(initial_count)
print(filtered_count)
print(f"\nNumber of rows removed with zero or negative 'Vehicle Count': {rows_removed}")
print("\nFiltered Dataset (No zeros or negative values in 'Vehicle Count'):")
print(RE_EV_sales.head())
```

Figure 2. Data Preprocessing

3.3 Data Visualization and Exploration

Code Sections 9 to 12 does the exploratory Data Analysis on the dataset to provide valuable visual information like in the Figure 3. The Figure has a bar chart that represents the state-wise vehicle registered count for each category of vehicles. The Pie Chart shows the county-wise vehicle registered. And The line graph displays the Time Series from 2017 to 2024.

Figure 3. The EDA of all three categories of vehicles

3.4 Seasonal Decomposition and Stationarity Check

The code sections from 13 to 16 are executed to perform seasonal decomposition, stationarity check, and differencing.

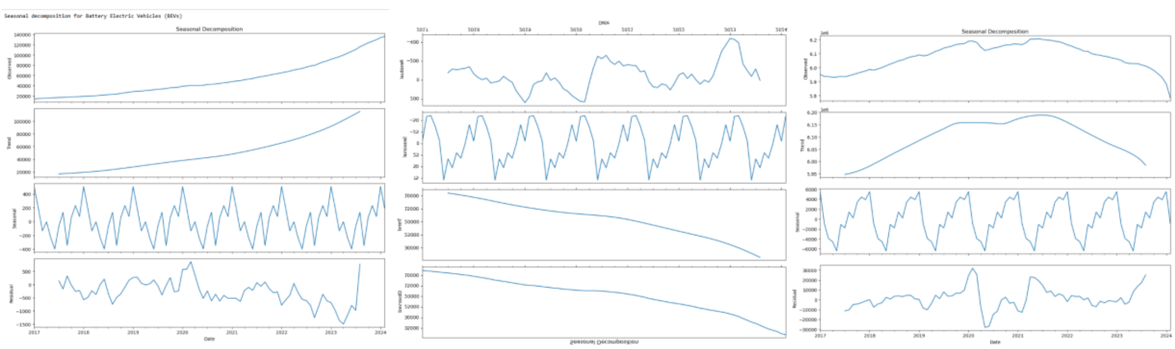


Figure 4. The seasonal decomposition of various categories of vehicles

3.5 Time Series Modeling

Further code sections from 17 to 27 will be executed to run the various models namely ARIMA Model, SARIMA Model, Exponential Smoothing Model, Prophet Model, and LSTM Model. The data is split into training and testing datasets and the models are tested on this. Further sections display code for hyperparameter tuning to find the optimum configuration for the LSTM model. Properties such as learning rates, epoch size, and factors are used to perform the same.

3.6 Evaluation

The model can be evaluated using metrics such as AIC, RMSE, MAE, MAPE, and R^2 by executing the evaluate_and_forecast custom functions. For visual evaluation, there is a predicted vs actual comparison chart programmed to visualize as in Figures 5.1, 5.2, and 5.3 for all categories of vehicles.

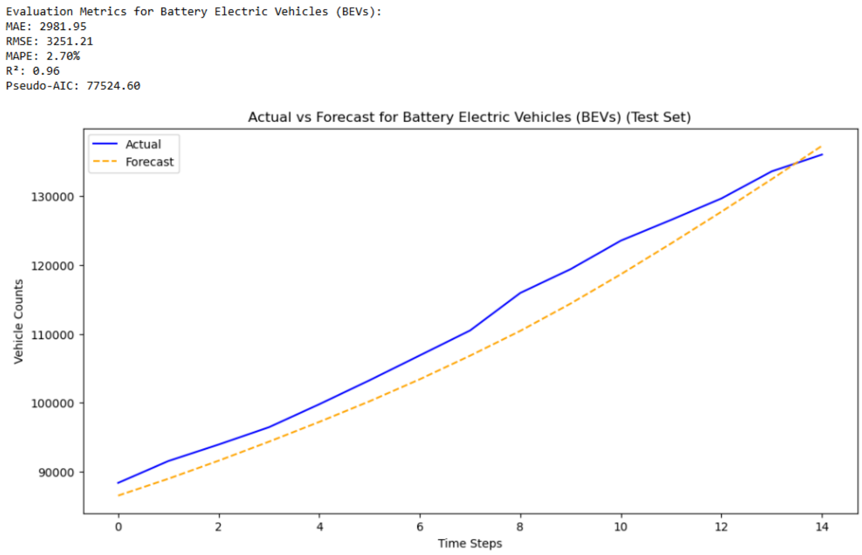


Figure 5.1 LSTM with Hyperparameter Tuning Model Evaluation for Battery-Operated Electric Vehicles

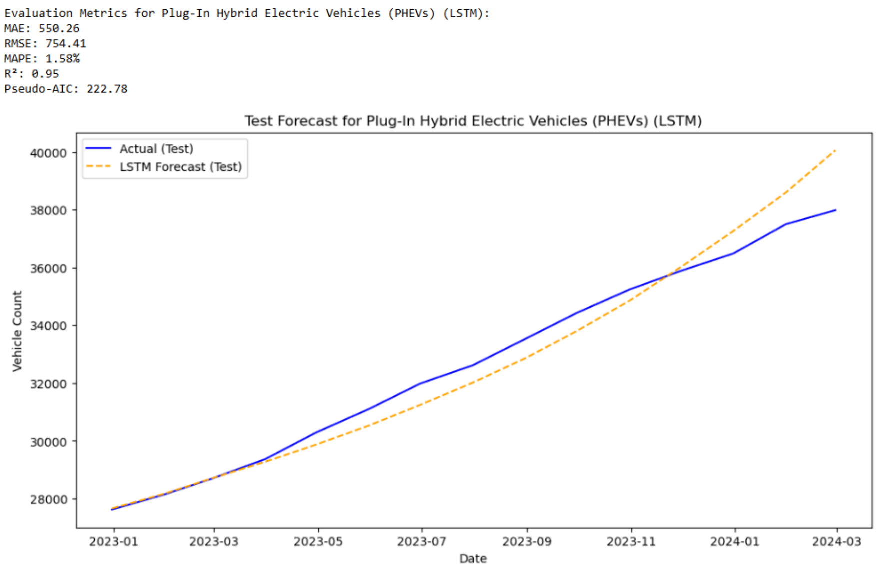
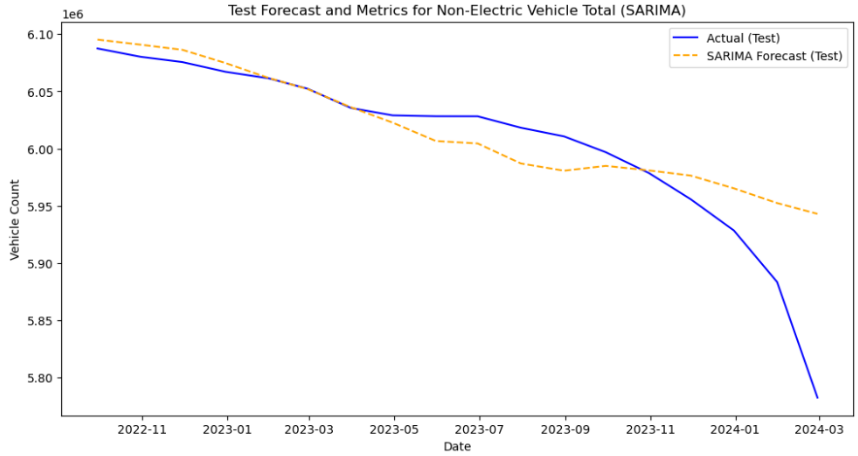


Figure 5.2 LSTM Model Evaluation for Plug-in Hybrid Electric Vehicles




```
Evaluation Metrics for Non-Electric Vehicle Total (SARIMA):
MAE: 25103.75
RMSE: 44543.92
MAPE: 0.43%
R²: 0.65
AIC: 881.96
```

Figure 5.3 SARIMA Model Evaluation for Non-Electric Vehicles

3.7 Conclusion

We use various models to perform prediction on the Time Series Model. A number of Models performed better for a particular category of vehicle. As displayed in Figure 5.1, for battery-operated vehicles, the LSTM model using Hyperparameter tuning produced the best fit for the model explaining 96% percent of the data. As in Figure 5.2, a normal LSTM model in its default configuration performed well for Plug-in Hybrid with 95% of data explained by the model. Further in Figure 5.3, for Non-electric vehicles, the SARIMA Model worked out well with 73% of data being explained by the model.