Configuration Manual

MSc Research Project
Data Analytics

Gopi krishna Marlapati
Student ID: x23224061

School of Computing
National College of Ireland

Supervisor: Dr Anu Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Gopi Krishna Marlapati		
Student ID:	x23224061		
Programme:	MSc in Data Analytics	**Programme:**	MSc in Data Analytics
Module:	Research Project		
Supervisor:	Dr Anu sahni		
Submission Due Date:	12/12/2024		
Project Title:	Potential Improvement in Sales of EVs through Effective Use of Sales Trend Analysis in Strategic Decision Making		
Word Count:	1240	**Page Count:**	6

Signature: Gopi Krishna 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):	

Configuration Manual

Gopi Krishna Marlapati
Student ID: x23224061

1 Introduction

This document describes in detail how to setup and configure the environment for time series analysis using Machine Learning models, that is ARIMA and SARIMA. In the accompanying README file the setup involves working with large datasets sourced from multiple repositories. Since computing and processing of these datasets requires both hardware as well as software, this document details the configuration of both. The emphasis is on the ease of integration of machine learning models into infrastructure with the use of sophisticated computational power for accurate time series forecasting. It provides step by step instructions for setting up a robust environment for managing and analyze of big data efficiently.

2 Hardware Requirements

Processor: Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz 2.11 GHz
Memory (RAM) Installed: 16 GB
System Type: Windows 11 Pro, 64 Bit
Operating System with x64-based
processor
Storage: 500 GB SSD and 1TB HDD
GPU: 8 GB, internal graphics card

3 Software Requirements

Jupyter Notebook was utilized as the Integrated Development Environment (IDE) for this This section outlines the software requirements essential for configuring and executing the time series analysis setup:

1. **Python 3.9.12**
   The Machine Learning ARIMA and SARIMA models are implemented in Python. It is a version that guarantees to work with required libraries and frameworks for time series analysis and data processing.

2. **Jupyter Notebook 7.0.8**
   Jupyter Notebook allows you to code and visualize as well as document your work, in an interactive environment! This helps make the analysis run and document, and ease of use and reproducibility.

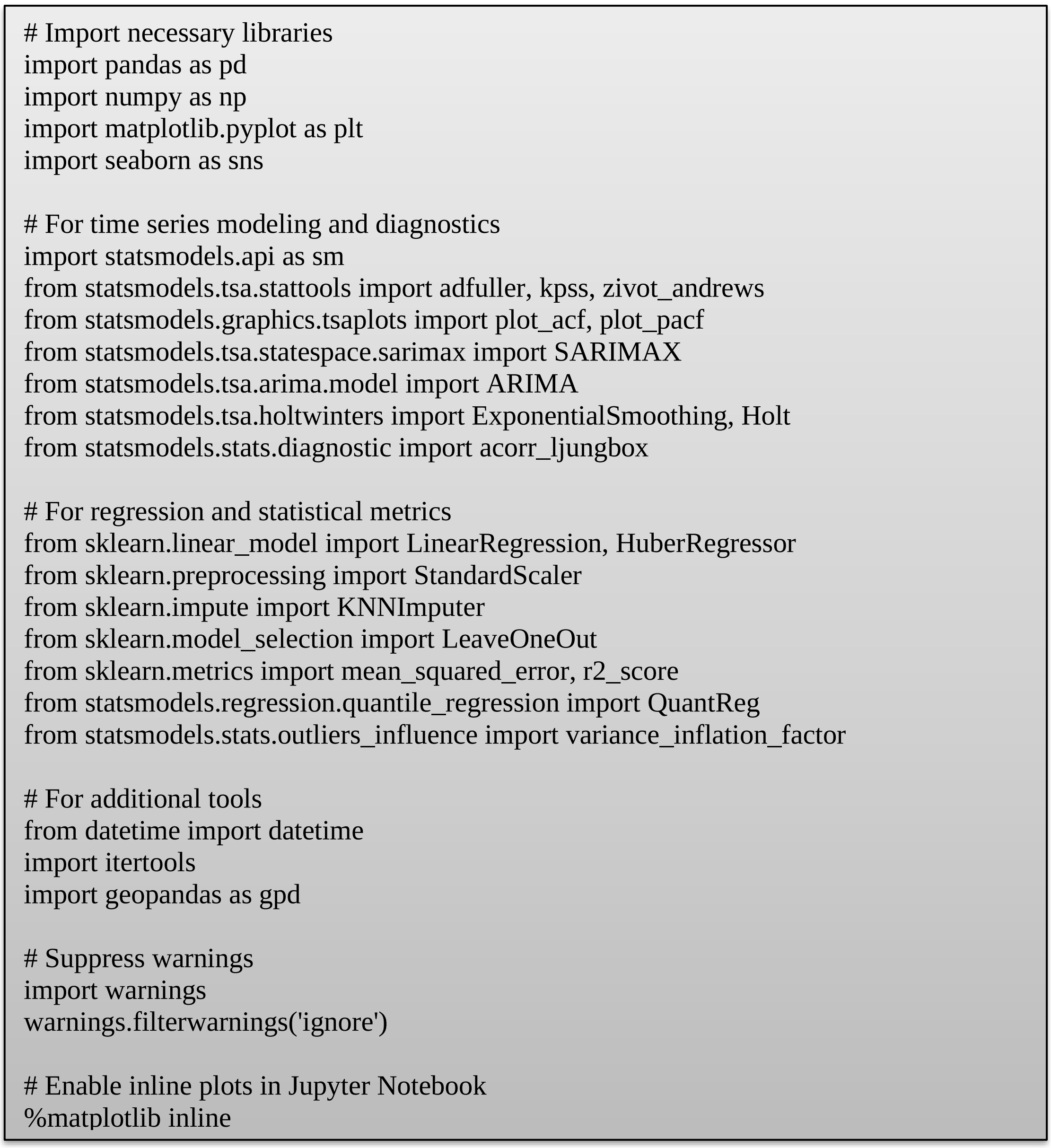
3. **API Data Retrievals**
   To fetch datasets from external sources dynamically you need to have some tools and

libraries to get data from APIs. This allows for real time data integration with the analysis workflow.

4. **Draw.io**
   Visual diagrams, which often illustrate strategies, procedures and workflows are made using Draw.io. That is, it helps to document and explain the setup and analysis methodology effectively.

The backbone of the configuration, comprised of these software components guarantees a hassle-free setup for machine learning and time series analysis tasks.

4 Library Package Requirements



```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# For time series modeling and diagnostics
import statsmodels.api as sm
from statsmodels.tsa.stattools import adfuller, kpss, zivot_andrews
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.holtwinters import ExponentialSmoothing, Holt
from statsmodels.stats.diagnostic import acorr_ljungbox

# For regression and statistical metrics
from sklearn.linear_model import LinearRegression, HuberRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.impute import KNNImputer
from sklearn.model_selection import LeaveOneOut
from sklearn.metrics import mean_squared_error, r2_score
from statsmodels.regression.quantile_regression import QuantReg
from statsmodels.stats.outliers_influence import variance_inflation_factor

# For additional tools
from datetime import datetime
import itertools
import geopandas as gpd

# Suppress warnings
import warnings
warnings.filterwarnings('ignore')

# Enable inline plots in Jupyter Notebook
%matplotlib inline
```

Figure 1: impots used

The required python packages installed in the environment using ’pip’ command is used to install all packages.

5 Dataset Description

IEA_data.csv is the dataset that contains geo specific electric vehicle (EV) metrics data for 2011 and ahead. The dataset includes the following columns:

- **region**: Such as Australia and the country or area.
- **category**: Tells if the data is "Historical" or another classification.
- **parameter**: It describes the metric such as EV stock share, EV sales share, EV stock.
- **model**: It indicates type of vehicle (eg: Cars).
- **powertrain**: It is a type of EV such as BEV (Battery Electric Vehicle).
- **year**: The year of the data point.
- **unit**: The unit of measurement (percent, vehicles).
- **value**: The recorded value for the given metric.

The dataset is used to follow sales and stock of EV with time to analyze progress in EV adoption.

This infrastructure dataset yields location specific details about electric vehicle (EV) charging points in different towns, states and countries. Attributes such as ID, town, state, charger power (Power KW), quantity of units available are included in it. The dataset includes: the type of charger connection (e.g., CCS Type 1 or Type 2), and the current type: AC or DC. The period information will be put in additional columns as country, general comments on the chargers, the date that the information was created. This data can be spread across the world to be able to analyze distribution and infrastructure of the EV charging points.

6 Dataset Preparation and Pre-processing

pandas Data Frame is loaded from the dataset using pd.read_csv() as o First. (EV) data, the following steps are essential to ensure the dataset is clean, consistent, and ready for analysis:

1. **Loading and Exploring the Data**:
    - First, load the dataset into a pandas Data Frame using pd.read_csv(). This will allow us to inspect the data structure, check for missing values, and analyze the distribution of variables.

2. **Handling Missing Values**:

    - Investigate missing values using df.isnull().sum() to understand which columns have missing data. For numerical columns, imputation techniques like KNN imputation can be used via KNNImputer from sklearn. For categorical columns, the missing values can be handled by replacing them with the mode of the column or using forward/backward fill if appropriate.

3. **Converting Data Types**:

    - Ensure the year and value columns are in the correct data types. The year column should be converted to an integer, and the value column should be a float. The category, parameter, and powertrain columns should remain as categorical data types to ensure optimal performance during analysis.

4. **Handling Date Columns**:

    - If there are any time-related columns, ensure they are in datetime format using pd.to_datetime(). This will facilitate time series analysis and allow us to create time-based features, like extracting the month or quarter.

    -

5. **Outlier Detection**:

    - Visualizations like box plots (sns.boxplot()) aand statistical methods can be used to de-tect outliers in the value column.. If the criteria and the analysis objective consider the data's context, extreme outliers can be capped or re-removed.

6. **Data Normalization/Standardization**:

    - For that, you can normalize / standardize the numeric using Standard scaler sklearn if required. The importance of this comes in when feature comparison from varying unit or scale have to be made.

    -

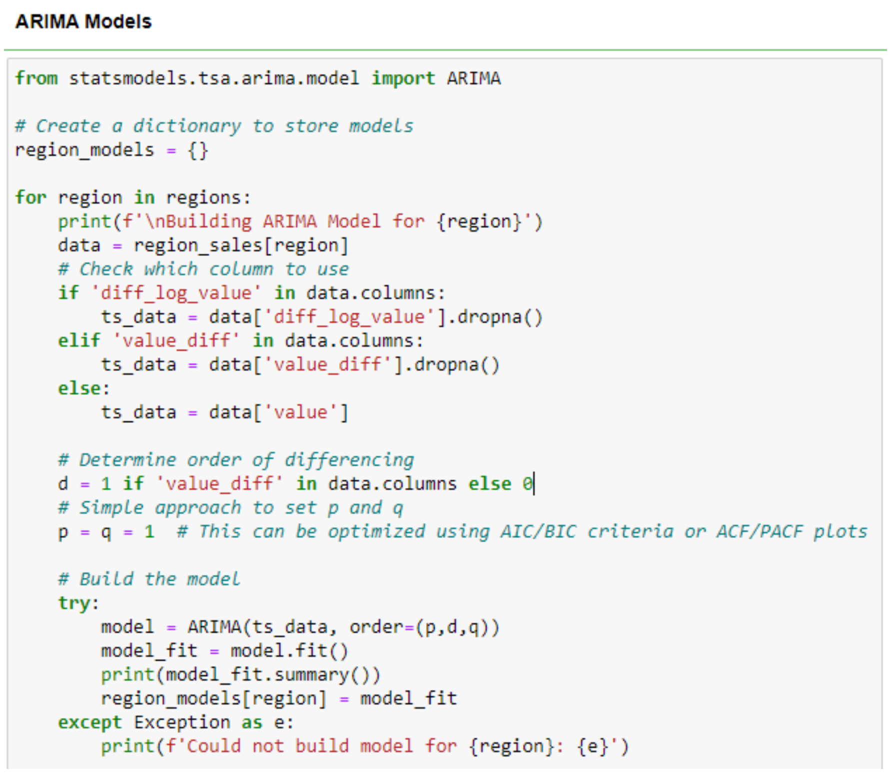
7. **Feature Engineering**:

    - Additional features like moving averages or lag features may be created for time series modeling to capture trends and seasonality.

Model Preparation:

The first step is to make sure the data is stationary (that is ARIMA requires this property). The Augmented Dickey–Fuller (ADF) test can test for stationarity and from there, if not stationary, can be differenced to get the data to be stationary. The next step then, is finding the best values for p, d, q where p is the number of lag observations, d is the degree of differencing, and q is the size of the moving average window. The Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots are analyzed to achieve this. Having chosen the parameters, we then fit the ARIMA model into the historical data. Once the residuals are checked to make sure that the model is still well specified, we then check the model's residuals to make sure no patterns remain unexplained. After that, the ARIMA model is utilized to forecast future sales, and then later its performance is evaluated by comparing forecast values against the actual data using accuracy metrics like RMSE, or MAPE.

6.1 ARIMA Model

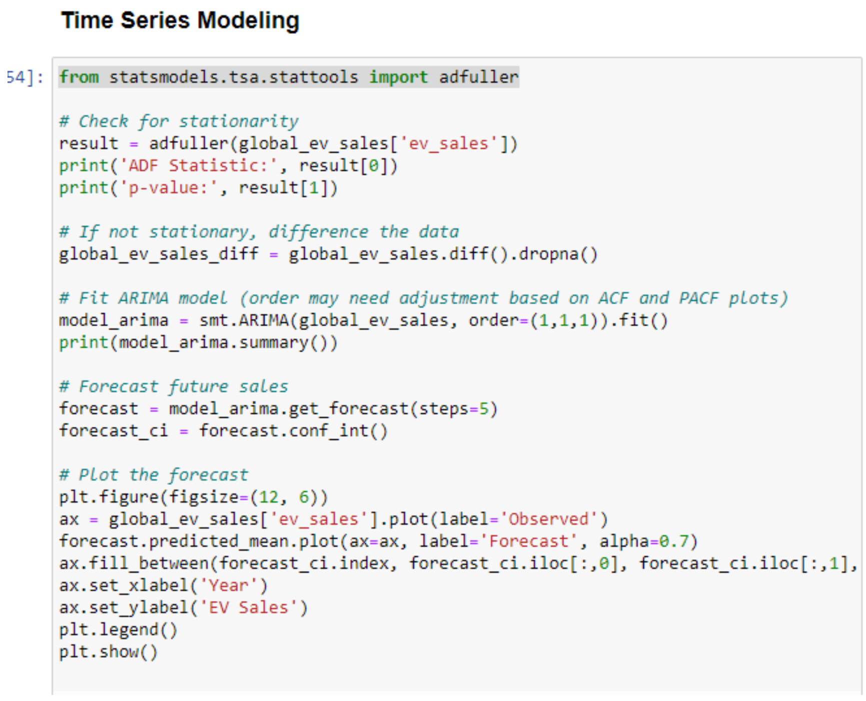


ARIMA Models

```
from statsmodels.tsa.arima.model import ARIMA

# Create a dictionary to store models
region_models = {}

for region in regions:
    print(f'\nBuilding ARIMA Model for {region}')
    data = region_sales[region]
    # Check which column to use
    if 'diff_log_value' in data.columns:
        ts_data = data['diff_log_value'].dropna()
    elif 'value_diff' in data.columns:
        ts_data = data['value_diff'].dropna()
    else:
        ts_data = data['value']

    # Determine order of differencing
    d = 1 if 'value_diff' in data.columns else 0
    # Simple approach to set p and q
    p = q = 1  # This can be optimized using AIC/BIC criteria or ACF/PACF plots

    # Build the model
    try:
        model = ARIMA(ts_data, order=(p,d,q))
        model_fit = model.fit()
        print(model_fit.summary())
        region_models[region] = model_fit
    except Exception as e:
        print(f'Could not build model for {region}: {e}')
```

Figure 2: *ARIMA Model*

6.2 SARIMA Model



Time Series Modeling

```
54]: from statsmodels.tsa.stattools import adfuller

# Check for stationarity
result = adfuller(global_ev_sales['ev_sales'])
print('ADF Statistic:', result[0])
print('p-value:', result[1])

# If not stationary, difference the data
global_ev_sales_diff = global_ev_sales.diff().dropna()

# Fit ARIMA model (order may need adjustment based on ACF and PACF plots)
model_arima = smt.ARIMA(global_ev_sales, order=(1,1,1)).fit()
print(model_arima.summary())

# Forecast future sales
forecast = model_arima.get_forecast(steps=5)
forecast_ci = forecast.conf_int()

# Plot the forecast
plt.figure(figsize=(12, 6))
ax = global_ev_sales['ev_sales'].plot(label='Observed')
forecast.predicted_mean.plot(ax=ax, label='Forecast', alpha=0.7)
ax.fill_between(forecast_ci.index, forecast_ci.iloc[:,0], forecast_ci.iloc[:,1],
ax.set_xlabel('Year')
ax.set_ylabel('EV Sales')
plt.legend()
plt.show()
```

***Figure** 3: SARIMA Model*

6.3 Strategies for Improving the sales

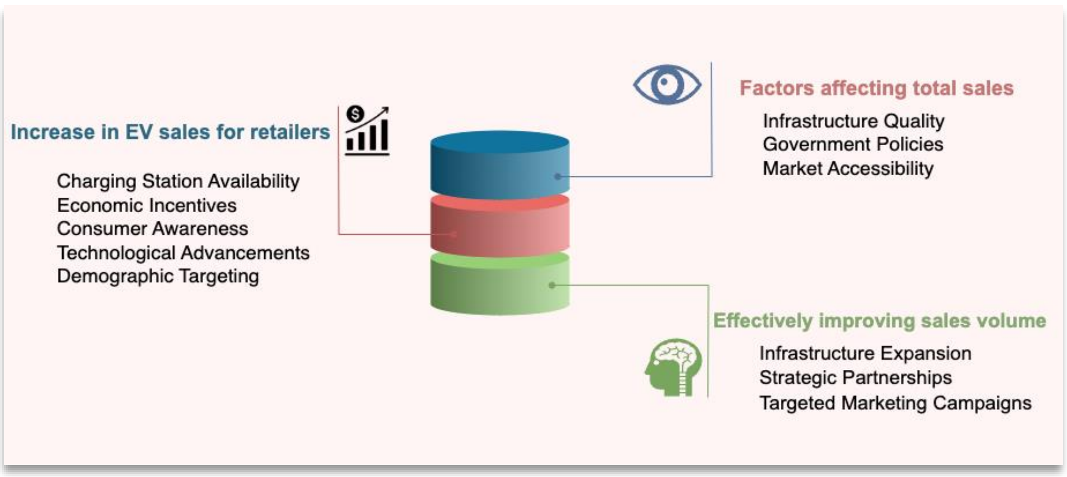

Factors affecting total sales
Infrastructure Quality
Government Policies
Market Accessibility
Increase in EV sales for retailers
Charging Station Availability
Economic Incentives
Consumer Awareness
Technological Advancements
Demographic Targeting
Effectively improving sales volume
Infrastructure Expansion
Strategic Partnerships
Targeted Marketing Campaigns

***Figure** 4: Sales Trend Analysis*

7 Applied ARIMA/SARIMA

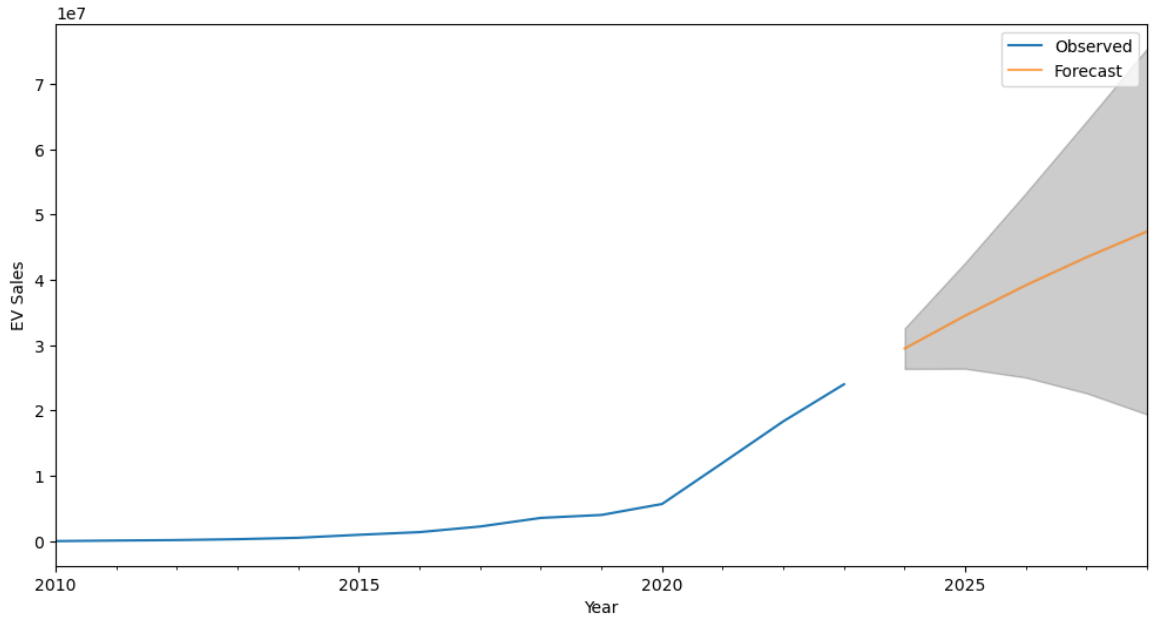


***Figure** 4: ARIMA/SARIMA*