Configuration Manual

MSc Research Project
Data Analytics

Jyothirmai Myneni
Student ID: X21235325

School of Computing
National College of Ireland

Supervisor: Vladimir Milosavljevic

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: …….…………………………Jyothirmai Myneni…………………………………………

Student ID: ………………………………X21235325……………………………………….……

Programme: …Data Analytics…………………………… **Year:** …2022-2023..

Module: …MSc Research Project………………………………………………………….………

Supervisor: …… Vladimir Milosavljevic …………………………………………….………

Submission Due Date: ………14/08/23………………………………………………………………….………

Project Title: Revolutionizing Demand Sales Forecasting: A Novel Approach through Ensemble of Statistical Time Series and Machine Learning Techniques

Word Count: ………722……………… **Page Count**………07……………………………………..……..

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: ……………………………Jyothirmai Myneni……………………………………………

Date: ……………………14/08/2023…………………………………………………………………………

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 primary objective of the project is to develop a strong and trustworthy demand forecasting model that can foretell future consumer wants across a wide variety of product and service categories. By analysing historical data and applying cutting-edge statistical and machine learning techniques to derive conclusions, the study aims to assist organizations with their planning. Demand-driven production, effective inventory management, clever marketing, and supply chain optimization might all profit from this. The ultimate objective is for businesses to have access to the resources they require in order to make educated decisions, modify their strategies in response to shifting market conditions, and improve operational effectiveness in order to more effectively satisfy customer expectations.

2 System Configuration

RAM: 16 GB
Processor: i5
OS: Windows
IDE: Jupyter
Language: Python

3 Importing the Packages

The relevant libraries and modules are imported in this part so they may be used throughout the script. Among these libraries are pandas, numpy, matplotlib, stats models, scikit-learn, and others, which are frequently used for tasks including data processing, visualization, statistical analysis, and machine learning.

```python
import pandas as pd
import numpy as np
import json
import math
import copy
import os
import math
from math import sqrt
import matplotlib.pyplot as plt
import statistics
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima_model import ARMA
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
from statsmodels.tsa.stattools import acf
from statsmodels.tsa.stattools import pacf
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.linear_model import ElasticNet
from sklearn.linear_model import HuberRegressor
from sklearn.linear_model import LassoLars
from sklearn.linear_model import PassiveAggressiveRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.tree import ExtraTreeRegressor
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sktime.forecasting.arima import AutoARIMA
from scipy.stats import randint as sp_randint
from sktime.forecasting.trend import STLForecaster
# from IPython import get_ipython
import logging
import warnings
warnings.filterwarnings('ignore')
```

4 Importing the Data

The data is imported into the Jypiter notebook.

```
# In[2]:
# async def forecast_main(input_data_json):
if True:
    def user_input():
        """

        This Function takes into the account for both types of Data input. Whether the data is in .csv or .json.

        Future will try to input the excel as well

        """

        forecast_period = input("What is the Future Forecast period? ")
        forecast_period = int(forecast_period)
        input_data = input("Enter dataset: ")
    #    details=input("Enter details dataset: ")

        file_type = os.path.splitext(input_data)[1]
        if file_type == '.csv':

            dataset = pd.read_csv(input_data)
        elif file_type == '.json':

            dataset = retrieve_data(input_data)
```

5 Data Imputation

Data Imputation is carried on the dataset. The missing values are replaced with zeros using this function, which takes a dataset as input. The dataset is filled with zeros to represent ‘NaN’ values using the ‘fillna’ technique. Returned is the updated dataset.

```
def data_imputation_zero(dataset):
    dataset.fillna(0, inplace=True)
    return dataset

def Moving_Average(data, tsize):
    rmse = dict()
    model_predictions = dict()
    key = ['sma', 'wma']
    train, test = data[0:-tsize], data[-tsize:]
    test = pd.DataFrame(test)
    test = test.reset_index(drop=True)
#    expected = test
    test_shape = len(test)
    if len(key) > 0:
        if len(key) == 2:
            predictions, rmse_i = model_MA(
                key[0], train, test_shape, train_flag=1)
            rmse[key[0]] = rmse_i
            model_predictions[key[0]] = predictions

            predictions, rmse_i = model_MA(
                key[1], train, test_shape, train_flag=1)
            rmse[key[1]] = rmse_i
            model_predictions[key[1]] = predictions
        else:
            predictions, rmse_i = model_MA(
                key[0], train, test_shape, train_flag=1)
            rmse[key[0]] = rmse_i
            model_predictions[key[0]] = predictions

    print("Moving_Average done")

    return rmse, model_predictions
```

6 Calculate Forecast Accuracy

This program defines a function called calculate_forecast_accuracy that computes several accuracy measures for assessing a forecasting model's performance. Expected (the actual value of the expectation) and prediction (the anticipated value of the prediction) are the two arguments that the function accepts. It returns three indicators for accuracy: bias, mean absolute percentage error (MAPE), and forecast accuracy.

```
def calculate_forecast_accuracy(expected, forecast):
    if math.isnan(expected):
        expected = 0
    else:
        expected = int(expected)
    expected = int(expected)
    forecast = int(forecast)
    print("calculate_forecast_accuracy")
    facc = (1 - (np.abs(expected - forecast)) /
            (expected+(expected == 0))) * 100
    if facc < 0:
        facc = 0
    mape = (np.abs(expected - forecast) / expected) * 100
    bias = (expected - forecast)

    if np.isnan(facc) == True or np.isfinite(facc) == False:
        facc = 0
    if np.isnan(mape) == True or np.isfinite(mape) == False:
        mape = 0
    if np.isnan(bias) == True or np.isfinite(bias) == False:
        mape = 0
    return float(format(facc, '.3f')), float(format(mape, '.3f')), float(format(bias, '.3f'))
```

7 Training and Implementing the Model

The below figures depict the training and implementation of the model. Data frame is used in the training process. On the pre-processed data, a number of time series models are trained and assessed. (Time series using ML) Machine learning models. Models with the time_series_models function include ARIMA, Exponential Smoothing (ES), naïve, and moving average (MA).The Croston_TSB function, or Croston's approach.

```
def training(datasets, forecast_period):
    # forecast_period=forecast_period+1
    forecast_results = []
    num = 0
    col = ['sku', 'model', 'rmse', 'mape']  # changed 2
    fc = []
    for i in range(1, forecast_period+1):
        fc.append('forecast'+str(i))
    col.extend(fc)
    output_all = pd.DataFrame(columns=col)
    models_out = pd.DataFrame(columns=['sku', 'model_ts', 'model_ml'])
    output_best = pd.DataFrame(columns=col)

    for incr, sku in enumerate(datasets):
        try:
            # if "ANZ_Hardware Billed" not in sku: continue
            num += 1

            print("------------------------------------------------------------")
            print("Running SKU %d: %s..." % (num, sku))
            stp_copy = copy.deepcopy(datasets[sku].T)

            raw_data = copy.deepcopy(datasets[sku].T)
            output = init_output(forecast_period, raw_data)

            dataset = raw_data.copy()

            dataset = dataset[:-1]
            interval = find_interval(dataset.index)

            logging.info(interval.days)
```

```
data_copy = sku_data.copy()
data_copy = np.array(data_copy)

index1, index2, sflag1, sflag2 = Sesonal_detection(sku_data)
sku_data = outlier_treatment_tech(sku_data, interval, size)
sku_data = np.array(sku_data[0])

# print(sku_data)

if sflag1 == 1:
    sku_data[index1] = data_copy[index1]
if sflag2 == 1:
    sku_data[index2] = data_copy[index2]
else:
    sku_data = sku_data

sku_data = pd.DataFrame(sku_data)

# Testing Stationarity
d = 0
df_test_result = dickeyfullertest(
    sku_data.T.squeeze())  # pd.Series(sku_data[0])

while df_test_result == 0:
    d += 1
    if d == 1:
        new_data = difference(sku_data[0].tolist())
    else:
        new_data = difference(new_data)
    df_test_result = dickeyfullertest(new_data)
sample = np.array(sku_data)
repeat = check_repetition(sample, freq, 1, len(sample))
# Finding p and q value
try:
    if d == 0:
        p1, ps, pl = acf_plot(sku_data, freq)
        q = pacf_plot(sku_data, freq)
        data = sku_data
    else:

        p, ps, pl = acf_plot(new_data, freq)
        q = pacf_plot(new_data, freq)
        data = new_data
```

Model is implemented and the RMSE values are calculated for the model.

```
print("Modeling done")
rmse_TS = rmse_ARIMA.copy()
rmse_TS.update(rmse_ES)
rmse_TS.update(rmse_naive)
rmse_TS.update(rmse_ma)

predictions = predictions_ML
predictions.update(predictions_ARIMA)
predictions.update(predictions_ES)
predictions.update(predictions_naive)
predictions.update(predictions_ma)

rmse_Croston, predictions_Croston = Croston_TSB(
    sku_data, tsize)
rmse_TS.update(rmse_Croston)
predictions.update(predictions_Croston)

for key in rmse_ML:

    forecast_period_loc = forecast_period
    models_to_shift = ["lr", "ridge",
                       "lasso", "en", "llars", "pa", "knn"]
    model_to_be_shifted = True if key in models_to_shift else False
    if model_to_be_shifted:
        forecast_period_loc = forecast_period_loc + 1
    model_forecast = model_predict(
        key, best_order, data, forecast_period_loc)
    if model_to_be_shifted:
        model_forecast = model_forecast[1:]

    # model_forecast = model_predict(
    #     key, best_order, data, forecast_period)
    model_forecast = [0 if i < 0 else int(
        i) for i in model_forecast]
    rmse_val = np.mean(rmse_ML[key])
    mape_val = np.mean(calculate_validation_mape(
        expected, predictions[key]))
    output_dict = {'sku': sku, 'model': key,
                   'rmse': rmse_val, 'mape': mape_val}
```

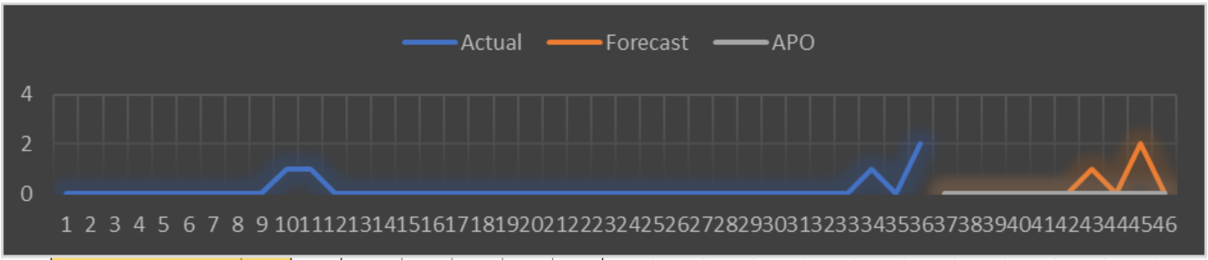
8 Output

The results are calculated for all the ensemble and time series models , some of them are shown below:

```
RMSE FOR HWES: 45472077
RMSE FOR naive: 46273951
RMSE FOR naive: 43442727
RMSE FOR naive: 9523575
RMSE FOR naive_rept: 159537470
RMSE FOR naive_rept: 120620917
RMSE FOR naive_rept: 67748667
RMSE FOR naive3: 171599400
RMSE FOR naive3: 185869464
RMSE FOR naive3: 189006779
RMSE FOR naive6: 160422139
RMSE FOR naive6: 104488121
RMSE FOR naive6: 17818697
RMSE FOR naive12: 46273951
RMSE FOR naive12: 43442727
RMSE FOR naive12: 9523575
RMSE FOR naive12wa: 78244194
RMSE FOR naive12wa: 71518711
RMSE FOR naive12wa: 25099943
RMSE FOR sma: 180741458
RMSE FOR sma: 152295176
RMSE FOR sma: 192464581
RMSE FOR wma: 182634980
RMSE FOR wma: 154393536
RMSE FOR wma: 201180328
Moving_Average done
Modeling done
RMSE FOR Croston: 80592523
BEST MODELS : ['llars', 'naive']
ERRORS OF BEST MODELS : 21026602.934839252 33080084.872166004
weight ts: 0.44222225815815647
weight ml: 0.5577777418418436
RMSE FOR Ensemble: 94715599
RMSE FOR naive6wa: 165163649
RMSE FOR naive6wa: 100332950
RMSE FOR naive6wa: 48885502
Best forecast from ML
Validation Accuracy
RMSE FOR : 144713794
```

9 Excel Graphs:

The excel files are used for comparing our model with other models like, APO. We can observe that APO is unable to fit the actual model and predicting the straight line, while our model is predicting the future sales very accurately.



Input		Step	Prof	Seas-Y/P/N	Trend	NPI?	Sparse?	Maturing
Zfuso_Material	MK671693	1200		Partial-Seasc	Normal	NPI	High	27

Zfuso_Material	Zfuso_Plant_Location	Zfuso_Mat	Keyfigure	Model	201608	201609	201610	201611	201612	201701	201702	201703	201704	201705	201706	201707	201708	201709	201710	201711
#N/A	#N/A	#N/A	#N/A	Actual	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
				Forecast																
				APO