```
# -*- coding: utf-8 -*-
"""
Created on Fri Apr 16 13:46:10 2021

@author: wille
"""
# Visualisations and Tests

# In[1]:

# Math
from math import ceil, floor, sqrt

# Plotting
import matplotlib.pyplot as plt

# Numpy & Pandas
import numpy as np
import pandas as pd

# Stats models
import statsmodels.api as sm
from matplotlib.pyplot import figure

# Linear Imputation
from scipy.interpolate import interp1d

# Machine Learning
from sklearn.metrics import mean_squared_error
from statsmodels.tsa.api import VAR
from statsmodels.tsa.arima.model import ARIMA

# Augmented Dickey Fuller Test
from statsmodels.tsa.stattools import adfuller

# Options
pd.options.display.max_rows = 2500

import warnings

warnings.filterwarnings("ignore", "statsmodels.tsa.arima_model.ARMA", FutureWarning)
warnings.filterwarnings("ignore", "statsmodels.tsa.arima_model.ARIMA", FutureWarning)
warnings.filterwarnings("ignore")

import os
import time
import tkinter as tk
import urllib
from tkinter import *
from tkinter import ttk


# In[2]:

# Clean data frame like in appley to reflect data

df = pd.read_csv("C:/Users/wille/OneDrive/MSF&MSBA/5. Spring 2021/IS 6496 MSBA Capstone 3/CapstoneMainDF.csv", index_col=0)
productsdf = pd.read_csv("C:/Users/wille/OneDrive/MSF&MSBA/5. Spring 2021/IS 6496 MSBA Capstone 3/ProductsDF.csv", index_col=0)

df.columns = map(str.lower, df.columns)


# impute item names
productdict = dict(zip(productsdf.Item_ID, productsdf.Item_Desc))
```

```
df["product_name"] = df.item_id
df.product_name = df.product_name.map(productdict)
df["date"] = pd.to_datetime(df["date"])

display(df.product_name.head(10))

del (productdict, productsdf)


# create new project data set
projectdf = df.loc[(df["project"] != "NONE")]
display(projectdf.head())
print("Format of ProjectDF subset = ", format(projectdf.shape))


# ## Data Cleaning


# Remove Project from DF
df = df.loc[(df["project"] == "NONE")]
print("Format of DF subset = ", format(df.shape))


# Imputation of Temperature
mintemp_bu = df["mintemp"]
maxtemp_bu = df["maxtemp"]


df["mintemp"].interpolate(method="linear", inplace=True)
df["maxtemp"].interpolate(method="linear", inplace=True)


del (mintemp_bu, maxtemp_bu)

# ### Outlier Detection

# outlier detection
df["price"] = df["sales"] / df["quantity_sold"]

df.head(5)

# most sold items and pre calculations
x = (
    df.groupby(["date", "product_name"])
    .agg(
        daily_sales=pd.NamedAgg(column="sales", aggfunc=sum),
        daily_quantity=pd.NamedAgg(column="quantity_sold", aggfunc=sum),
    )
    .reset_index()
)
x["date"] = pd.to_datetime(x["date"])

x2 = (
    x.groupby(["product_name"])
    .agg(
        Total_sales=pd.NamedAgg(column="daily_sales", aggfunc=sum),
        Total_quantity=pd.NamedAgg(column="daily_quantity", aggfunc=sum),
        mean_daily_sales=pd.NamedAgg(column="daily_sales", aggfunc="mean"),
        mean_daily_quantity=pd.NamedAgg(column="daily_quantity", aggfunc="mean"),
        Transaction_days=pd.NamedAgg(column="product_name", aggfunc="count"),
    )
    .sort_values("Total_sales", ascending=False)
)

x2["Total_Avg_Price"] = x2["Total_sales"] / x2["Total_quantity"]
# x2 = x2.round(2)
```

```
x2.head(15)

del (x2, x)


tootsiebf = (
    df.groupby(["date", "product_name"])
    .agg(
        daily_sales=pd.NamedAgg(column="sales", aggfunc=sum),
        daily_quantity=pd.NamedAgg(column="quantity_sold", aggfunc=sum),
    )
    .reset_index()
)


df.loc[df.product_name == "TTS TOOTSIE ROLL $.10", "price"] = 0.10
df.sales = df.quantity_sold * df.price

tootsieafter = (
    df.groupby(["date", "product_name"])
    .agg(
        daily_sales=pd.NamedAgg(column="sales", aggfunc=sum),
        daily_quantity=pd.NamedAgg(column="quantity_sold", aggfunc=sum),
    )
    .reset_index()
)

# plot
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(8, 6), sharex=True, sharey=True)
fig.suptitle("Tootsie Roll Daily Sales values Before and After Price Imputation")

ax1.plot(tootsiebf.date, tootsiebf.daily_sales, color="#bf021f", linewidth=0.75)
ax1.tick_params("x", labelrotation=45)
ax1.set_ylabel("Daily Sales before Imputation")

ax2.plot(tootsieafter.date, tootsieafter.daily_sales, color="#bf021f", linewidth=0.75)
ax2.set_xlabel("Date")
ax2.tick_params("x", labelrotation=45)
ax2.set_ylabel("Daily Sales after Imputation")

plt.show()

max(tootsiebf.daily_sales)

temp = (
    df.groupby(["date"])
    .agg(
        daily_sales=pd.NamedAgg(column="sales", aggfunc=sum),
        daily_quantity=pd.NamedAgg(column="quantity_sold", aggfunc=sum),
    )
    .reset_index()
)


# Create Aggregated Data Set
x = (
    df.groupby(["date", "site_id"])
    .agg(
        daily_sales=pd.NamedAgg(column="sales", aggfunc=sum),
        daily_quantity=pd.NamedAgg(column="quantity_sold", aggfunc=sum),
    )
    .reset_index()
)
x["date"] = pd.to_datetime(x["date"])
x["price"] = (x["daily_sales"] / x["daily_quantity"]).round(2)

```

```
# Create average sales Dict and Impute into X
z = (
    df.groupby(["date", "site_id"])
    .agg(total_sales=pd.NamedAgg(column="sales", aggfunc=sum))
    .reset_index()
)
z = (
    z.groupby(["site_id"])
    .agg(average_sales=pd.NamedAgg(column="total_sales", aggfunc="mean"))
    .reset_index()
)
salesdict = dict(zip(z.site_id, z.average_sales))
x["average_sales"] = x.site_id
x.average_sales = x.site_id.map(salesdict)

# Calculate Difference between Daily Sales and Average Sales
x["Sales_Difference"] = np.absolute(
    ((x["daily_sales"] - x["average_sales"]) / x["average_sales"]) * 100
).round(2)
x = x.sort_values("Sales_Difference", ascending=False)

del (z, salesdict)

x_before = x

# Calculate a Standard Deviation
th_4std = np.std(x.Sales_Difference) * 4 + np.mean(x.Sales_Difference)

# Find Sale Differences larger then 4 Std
x2 = x.loc[(x["Sales_Difference"] >= th_4std)].sort_values(
    "Sales_Difference", ascending=False
)

print(
    "This method at 4 std classifies",
    round((len(x2) / len(df) * 100), 4),
    "% as outliers or ",
    len(x2),
    "days",
)

# Create Imputation List

site_vector = list(x2["site_id"])
date_vector = list(x2["date"])
index_list = []
df.reset_index(inplace=True)
for i in range(len(site_vector)):
    x = (
        df.loc[(df["site_id"] == site_vector[i]) & (df["date"] == date_vector[i])]
        .sort_values("sales", ascending=False)
        .head(1)
    )
    x = x.drop(
        columns=[
            "location_id",
            "open_date",
            "sq_footage",
            "locale",
            "maxtemp",
            "mintemp",
            "fiscal_period",
            "periodic_gbv",
            "current_gbv",
            "mpds",
        ]
```

```
    )
    index_list.append(x.iloc[0, 0])

df_bu = df

df.rename(columns={df.columns[0]: "index_set"}, inplace=True)

df = df[~df.index_set.isin(index_list)]

del index_list

temp2 = (
    df.groupby(["date"])
    .agg(
        daily_sales=pd.NamedAgg(column="sales", aggfunc=sum),
        daily_quantity=pd.NamedAgg(column="quantity_sold", aggfunc=sum),
    )
    .reset_index()
)
# Plot
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(8, 6), sharex=True, sharey=True)
fig.suptitle("Aggregate daily sales over all sites before and after outlier removal")

ax1.plot(temp.date, temp.daily_sales, color="#bf021f", linewidth=0.75)
ax1.tick_params("x", labelrotation=45)
ax1.set_ylabel("Agg. Sales before outlier removal")

ax2.plot(temp2.date, temp2.daily_sales, color="#bf021f", linewidth=0.75)
ax2.set_xlabel("Date")
ax2.tick_params("x", labelrotation=45)
ax2.set_ylabel("Agg. Sales after outlier removal")

plt.show()

# In[12]

# Run ETS
from statsmodels.tsa.exponential_smoothing.ets import ETSModel


#model = ETSModel(df.quantity_sold.astype('float64'))
#fit = model.fit(maxiter=5)
#print(fit.summary())

# In[13]

# Test for Seasonality and Trends and Visualaize them
from statsmodels.tsa.seasonal import seasonal_decompose

input_df = pd.DataFrame()
dfx = df.loc[
    (df.product_name == "MAR KG 3.29z SNICKERS 2 PIECE") & (df.site_id == 280)
].reset_index(drop=True)

sd = seasonal_decompose(dfx.quantity_sold, period=7)
input_df["date"] = dfx.date
input_df["quantity_sold"] = dfx.quantity_sold
input_df["observed"] = sd.observed
input_df["residual"] = sd.resid
input_df["seasonal"] = sd.seasonal
input_df["trend"] = sd.trend


def mround(x, m=5):
    """Helper method for multiple round"""
    return int(m * round(float(x) / m))
```

```
329
330
331 def plot_components(df):
332     """Plot data for initial visualization, ultimately visualized in Power BI
333     Args:
334         df (pandas dataframe)
335     """
336     df_axis = df.fillna(0)
337     ymin = mround(
338         np.min([df_axis.observed, df_axis.trend, df_axis.seasonal, df_axis.residual]), 5
339     )
340     ymax = mround(
341         np.max([df_axis.observed, df_axis.trend, df_axis.seasonal, df_axis.residual]), 5
342     )
343     ymin -= 5
344     ymax += 5
345
346     plt.figure(figsize=(20, 20))
347     plt.subplot(4, 1, 1)
348     plt.title(
349         "Original Data [Site_id: 280, Item: MAR KG 3.29z SNICKERS 2 PIECE] Period = 7",
350         fontsize=30,
351     )
352     plt.ylim(ymin, ymax)
353     plt.xticks(fontsize=20)
354     plt.yticks(fontsize=20)
355     plt.plot(df.index, df.observed, color="#bf021f", linewidth=0.75)
356
357     plt.subplot(4, 1, 2)
358     plt.title(
359         "Trend [Site_id: 280, Item: MAR KG 3.29z SNICKERS 2 PIECE] Period = 7",
360         fontsize=30,
361     )
362     plt.ylim(ymin, ymax)
363     plt.xticks(fontsize=20)
364     plt.yticks(fontsize=20)
365     plt.plot(df.index, df.trend, color="#bf021f", linewidth=0.75)
366
367     plt.subplot(4, 1, 3)
368     plt.title(
369         "Seasonal [Site_id: 280, Item: MAR KG 3.29z SNICKERS 2 PIECE] Period = 7",
370         fontsize=30,
371     )
372     plt.ylim(ymin, ymax)
373     plt.xticks(fontsize=20)
374     plt.yticks(fontsize=20)
375     plt.plot(df.index, df.seasonal, color="#bf021f", linewidth=0.75)
376
377     plt.subplot(4, 1, 4)
378     plt.title(
379         "Residual [Site_id: 280, Item: MAR KG 3.29z SNICKERS 2 PIECE] Period = 7",
380         fontsize=30,
381     )
382     plt.xticks(fontsize=20)
383     plt.yticks(fontsize=20)
384     plt.ylim(ymin, ymax)
385     plt.plot(df.index, df.residual, color="#bf021f", linewidth=0.75)
386
387     plt.tight_layout(pad=1.0, w_pad=1.0, h_pad=1.0)
388
389
390 plot_components(input_df)
391
392
393
394 # Change Period
```

```
input_df = pd.DataFrame()
dfx = df.loc[
    (df.product_name == "MAR KG 3.29z SNICKERS 2 PIECE") & (df.site_id == 280)
].reset_index(drop=True)

sd = seasonal_decompose(dfx.quantity_sold, period=365)
input_df["date"] = dfx.date
input_df["quantity_sold"] = dfx.quantity_sold
input_df["observed"] = sd.observed
input_df["residual"] = sd.resid
input_df["seasonal"] = sd.seasonal
input_df["trend"] = sd.trend


def mround(x, m=5):
    """Helper method for multiple round"""
    return int(m * round(float(x) / m))


def plot_components(df):
    """Plot data for initial visualization, ultimately visualized in Power BI
    Args:
        df (pandas dataframe)
    """
    df_axis = df.fillna(0)
    ymin = mround(
        np.min([df_axis.observed, df_axis.trend, df_axis.seasonal, df_axis.residual]), 5
    )
    ymax = mround(
        np.max([df_axis.observed, df_axis.trend, df_axis.seasonal, df_axis.residual]), 5
    )
    ymin -= 5
    ymax += 5

    plt.figure(figsize=(20, 20))
    plt.subplot(4, 1, 1)
    plt.title(
        "Original Data [Site_id: 280, Item: MAR KG 3.29z SNICKERS 2 PIECE] Period = 365",
        fontsize=30,
    )
    plt.ylim(ymin, ymax)
    plt.xticks(fontsize=20)
    plt.yticks(fontsize=20)
    plt.plot(df.index, df.observed, color="#bf021f", linewidth=0.75)

    plt.subplot(4, 1, 2)
    plt.title(
        "Trend [Site_id: 280, Item: MAR KG 3.29z SNICKERS 2 PIECE] Period = 365",
        fontsize=30,
    )
    plt.ylim(ymin, ymax)
    plt.xticks(fontsize=20)
    plt.yticks(fontsize=20)
    plt.plot(df.index, df.trend, color="#bf021f", linewidth=0.75)

    plt.subplot(4, 1, 3)
    plt.title(
        "Seasonal [Site_id: 280, Item: MAR KG 3.29z SNICKERS 2 PIECE] Period = 365",
        fontsize=30,
    )
    plt.ylim(ymin, ymax)
    plt.xticks(fontsize=20)
    plt.yticks(fontsize=20)
    plt.plot(df.index, df.seasonal, color="#bf021f", linewidth=0.75)

    plt.subplot(4, 1, 4)
```

```
        plt.title(
            "Residual [Site_id: 280, Item: MAR KG 3.29z SNICKERS 2 PIECE] Period = 365",
            fontsize=30,
        )
        plt.xticks(fontsize=20)
        plt.yticks(fontsize=20)
        plt.ylim(ymin, ymax)
        plt.ylim(ymin, ymax)
        plt.plot(df.index, df.residual, color="#bf021f", linewidth=0.75)

        plt.tight_layout(pad=1.0, w_pad=1.0, h_pad=1.0)


    plot_components(input_df)

    # In[15]:

    model = ETSModel(dfx.quantity_sold.astype('float64'))
    fit = model.fit(maxiter=10)
    print(fit.summary())


    # In[16]:



    input_df1 = pd.DataFrame()
    dfx = df.loc[
        (df.product_name == "MAR KG 3.29z SNICKERS 2 PIECE") & (df.site_id == 280)
    ].reset_index(drop=True)

    model = ETSModel(dfx.quantity_sold.astype('float64'))
    fit = model.fit(maxiter=10)
    print(fit.summary())

    sd = seasonal_decompose(dfx.quantity_sold, period=7)
    input_df1["date"] = dfx.date
    input_df1["quantity_sold"] = dfx.quantity_sold
    input_df1["observed"] = sd.observed
    input_df1["residual"] = sd.resid
    input_df1["seasonal"] = sd.seasonal
    input_df1["trend"] = sd.trend

    input_df2 = pd.DataFrame()
    dfx = df.loc[
        (df.product_name == "HER KG 2.8z REESES PB CUP") & (df.site_id == 580)
    ].reset_index(drop=True)

    model = ETSModel(dfx.quantity_sold.astype('float64'))
    fit = model.fit(maxiter=10)
    print(fit.summary())

    sd = seasonal_decompose(dfx.quantity_sold, period=7)
    input_df2["date"] = dfx.date
    input_df2["quantity_sold"] = dfx.quantity_sold
    input_df2["observed"] = sd.observed
    input_df2["residual"] = sd.resid
    input_df2["seasonal"] = sd.seasonal
    input_df2["trend"] = sd.trend

    input_df3 = pd.DataFrame()
    dfx = df.loc[
        (df.product_name == "TTS TOOTSIE ROLL $.10") & (df.site_id == 380)
    ].reset_index(drop=True)

    model = ETSModel(dfx.quantity_sold.astype('float64'))
```

```
fit = model.fit(maxiter=10)
print(fit.summary())

sd = seasonal_decompose(dfx.quantity_sold, period=7)
input_df3["date"] = dfx.date
input_df3["quantity_sold"] = dfx.quantity_sold
input_df3["observed"] = sd.observed
input_df3["residual"] = sd.resid
input_df3["seasonal"] = sd.seasonal
input_df3["trend"] = sd.trend

input_df4 = pd.DataFrame()
dfx = df.loc[
    (df.product_name == "HER KG 3z KIT KAT") & (df.site_id == 399)
].reset_index(drop=True)

model = ETSModel(dfx.quantity_sold.astype('float64'))
fit = model.fit(maxiter=10)
print(fit.summary())

sd = seasonal_decompose(dfx.quantity_sold, period=7)
input_df4["date"] = dfx.date
input_df4["quantity_sold"] = dfx.quantity_sold
input_df4["observed"] = sd.observed
input_df4["residual"] = sd.resid
input_df4["seasonal"] = sd.seasonal
input_df4["trend"] = sd.trend



plt.figure(figsize=(20, 25))
plt.subplot(4, 1, 1)
plt.title(
 "Seasonal [Site_id: 280, Item: MAR KG 3.29z SNICKERS 2 PIECE] Period = 7",
 fontsize=30,
 )
plt.ylim(-25, 25)
plt.xticks(fontsize = 20)
plt.yticks(fontsize = 20)
plt.plot(input_df1.index, input_df1.seasonal, color="#bf021f", linewidth=0.5)
plt.plot(input_df1.index, [0]*len(input_df1.index), color="black", linewidth=0.5)

plt.subplot(4, 1, 2)
plt.title(
 "Seasonal [Site_id: 580, Item: HER KG 2.8z REESES PB CUP] Period = 7",
 fontsize=30,
 )
plt.ylim(-25, 25)
plt.xticks(fontsize = 20)
plt.yticks(fontsize = 20)
plt.plot(input_df2.index, input_df2.seasonal, color="#bf021f", linewidth=0.5)
plt.plot(input_df2.index, [0]*len(input_df2.index), color="black", linewidth=0.5)

plt.subplot(4, 1, 3)
plt.title(
 "Seasonal [Site_id: 380, Item: TTS TOOTSIE ROLL $.10] Period = 7",
 fontsize=30,
 )
plt.ylim(-25, 25)
plt.xticks(fontsize = 20)
plt.yticks(fontsize = 20)
plt.plot(input_df3.index, input_df3.seasonal, color="#bf021f", linewidth=0.5)
plt.plot(input_df3.index, [0]*len(input_df3.index), color="black", linewidth=0.5)

plt.subplot(4, 1, 4)
plt.title(
```

```
    "Seasonal [Site_id: 399, Item: HER KG 3z KIT KAT] Period = 7",
    fontsize=30,
    )
  plt.ylim(-25, 25)
  plt.xticks(fontsize = 20)
  plt.yticks(fontsize = 20)
  plt.plot(input_df4.index, input_df4.seasonal, color="#bf021f", linewidth=0.5)
  plt.plot(input_df4.index, [0]*len(input_df4.index), color="black", linewidth=0.5)
```