

```

1  # -*- coding: utf-8 -*-
2  """
3  Created on Fri Apr 16 13:46:10 2021
4
5  @author: wille
6  """
7  # Visualisations and Tests
8
9  # In[1]:
10
11  # Math
12  from math import ceil, floor, sqrt
13
14  # Plotting
15  import matplotlib.pyplot as plt
16
17  # Numpy & Pandas
18  import numpy as np
19  import pandas as pd
20
21  # Stats models
22  import statsmodels.api as sm
23  from matplotlib.pyplot import figure
24
25  # Linear Imputation
26  from scipy.interpolate import interp1d
27
28  # Machine Learning
29  from sklearn.metrics import mean_squared_error
30  from statsmodels.tsa.api import VAR
31  from statsmodels.tsa.arima.model import ARIMA
32
33  # Augmented Dickey Fuller Test
34  from statsmodels.tsa.stattools import adfuller
35
36  # Options
37  pd.options.display.max_rows = 2500
38
39  import warnings
40
41  warnings.filterwarnings("ignore", "statsmodels.tsa.arima_model.ARMA", FutureWarning)
42  warnings.filterwarnings("ignore", "statsmodels.tsa.arima_model.ARIMA", FutureWarning)
43  warnings.filterwarnings("ignore")
44
45  import os
46  import time
47  import tkinter as tk
48  import urllib
49  from tkinter import *
50  from tkinter import ttk
51
52
53  # In[2]:
54
55  # Clean data frame like in appley to reflect data
56
57  df = pd.read_csv("C:/Users/wille/OneDrive/MSF&MSBA/5. Spring 2021/IS 6496 MSBA Capstone
58  3/CapstoneMainDF.csv", index_col=0)
59  productsdf = pd.read_csv("C:/Users/wille/OneDrive/MSF&MSBA/5. Spring 2021/IS 6496 MSBA
60  Capstone 3/ProductsDF.csv", index_col=0)
61
62  df.columns = map(str.lower, df.columns)
63
64  # impute item names
65  productdict = dict(zip(productsdf.Item_ID, productsdf.Item_Desc))

```

```

65 df["product_name"] = df.item_id
66 df.product_name = df.product_name.map(productdict)
67 df["date"] = pd.to_datetime(df["date"])
68
69 display(df.product_name.head(10))
70
71 del (productdict, productsdf)
72
73
74 # create new project data set
75 projectdf = df.loc[(df["project"] != "NONE")]
76 display(projectdf.head())
77 print("Format of ProjectDF subset = ", format(projectdf.shape))
78
79
80 # ## Data Cleaning
81
82
83 # Remove Project from DF
84 df = df.loc[(df["project"] == "NONE")]
85 print("Format of DF subset = ", format(df.shape))
86
87
88 # Imputation of Temperature
89 mintemp_bu = df["mintemp"]
90 maxtemp_bu = df["maxtemp"]
91
92
93 df["mintemp"].interpolate(method="linear", inplace=True)
94 df["maxtemp"].interpolate(method="linear", inplace=True)
95
96
97 del (mintemp_bu, maxtemp_bu)
98
99 # ### Outlier Detection
100
101 # outlier detection
102 df["price"] = df["sales"] / df["quantity_sold"]
103
104 df.head(5)
105
106 # most sold items and pre calculations
107 x = (
108     df.groupby(["date", "product_name"])
109     .agg(
110         daily_sales=pd.NamedAgg(column="sales", aggfunc="sum"),
111         daily_quantity=pd.NamedAgg(column="quantity_sold", aggfunc="sum"),
112     )
113     .reset_index()
114 )
115 x["date"] = pd.to_datetime(x["date"])
116
117 x2 = (
118     x.groupby(["product_name"])
119     .agg(
120         Total_sales=pd.NamedAgg(column="daily_sales", aggfunc="sum"),
121         Total_quantity=pd.NamedAgg(column="daily_quantity", aggfunc="sum"),
122         mean_daily_sales=pd.NamedAgg(column="daily_sales", aggfunc="mean"),
123         mean_daily_quantity=pd.NamedAgg(column="daily_quantity", aggfunc="mean"),
124         Transaction_days=pd.NamedAgg(column="product_name", aggfunc="count"),
125     )
126     .sort_values("Total_sales", ascending=False)
127 )
128
129 x2["Total_Avg_Price"] = x2["Total_sales"] / x2["Total_quantity"]
130 # x2 = x2.round(2)

```

```

131 x2.head(15)
132
133 del (x2, x)
134
135
136 tootsiebf = (
137     df.groupby(["date", "product_name"])
138     .agg(
139         daily_sales=pd.NamedAgg(column="sales", aggfunc=sum),
140         daily_quantity=pd.NamedAgg(column="quantity_sold", aggfunc=sum),
141     )
142     .reset_index()
143 )
144
145
146 df.loc[df.product_name == "TTS TOOTSIE ROLL $.10", "price"] = 0.10
147 df.sales = df.quantity_sold * df.price
148
149 tootsieafter = (
150     df.groupby(["date", "product_name"])
151     .agg(
152         daily_sales=pd.NamedAgg(column="sales", aggfunc=sum),
153         daily_quantity=pd.NamedAgg(column="quantity_sold", aggfunc=sum),
154     )
155     .reset_index()
156 )
157
158 # plot
159 fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(8, 6), sharex=True, sharey=True)
160 fig.suptitle("Tootsie Roll Daily Sales values Before and After Price Imputation")
161
162 ax1.plot(tootsiebf.date, tootsiebf.daily_sales, color="#bf021f", linewidth=0.75)
163 ax1.tick_params("x", labelrotation=45)
164 ax1.set_ylabel("Daily Sales before Imputation")
165
166 ax2.plot(tootsieafter.date, tootsieafter.daily_sales, color="#bf021f", linewidth=0.75)
167 ax2.set_xlabel("Date")
168 ax2.tick_params("x", labelrotation=45)
169 ax2.set_ylabel("Daily Sales after Imputation")
170
171 plt.show()
172
173 max(tootsiebf.daily_sales)
174
175 temp = (
176     df.groupby(["date"])
177     .agg(
178         daily_sales=pd.NamedAgg(column="sales", aggfunc=sum),
179         daily_quantity=pd.NamedAgg(column="quantity_sold", aggfunc=sum),
180     )
181     .reset_index()
182 )
183
184
185 # Create Aggregated Data Set
186 x = (
187     df.groupby(["date", "site_id"])
188     .agg(
189         daily_sales=pd.NamedAgg(column="sales", aggfunc=sum),
190         daily_quantity=pd.NamedAgg(column="quantity_sold", aggfunc=sum),
191     )
192     .reset_index()
193 )
194 x["date"] = pd.to_datetime(x["date"])
195 x["price"] = (x["daily_sales"] / x["daily_quantity"]).round(2)
196

```

```

197 # Create average sales Dict and Impute into X
198 z = (
199     df.groupby(["date", "site_id"])
200     .agg(total_sales=pd.NamedAgg(column="sales", aggfunc=sum))
201     .reset_index()
202 )
203 z = (
204     z.groupby(["site_id"])
205     .agg(average_sales=pd.NamedAgg(column="total_sales", aggfunc="mean"))
206     .reset_index()
207 )
208 salesdict = dict(zip(z.site_id, z.average_sales))
209 x["average_sales"] = x.site_id
210 x.average_sales = x.site_id.map(salesdict)
211
212 # Calculate Difference between Daily Sales and Average Sales
213 x["Sales_Difference"] = np.absolute(
214     ((x["daily_sales"] - x["average_sales"]) / x["average_sales"]) * 100
215 ).round(2)
216 x = x.sort_values("Sales_Difference", ascending=False)
217
218 del (z, salesdict)
219
220 x_before = x
221
222 # Calculate a Standard Deviation
223 th_4std = np.std(x.Sales_Difference) * 4 + np.mean(x.Sales_Difference)
224
225 # Find Sale Differences larger then 4 Std
226 x2 = x.loc[(x["Sales_Difference"] >= th_4std)].sort_values(
227     "Sales_Difference", ascending=False
228 )
229
230 print(
231     "This method at 4 std classifies",
232     round((len(x2) / len(df) * 100), 4),
233     "% as outliers or ",
234     len(x2),
235     "days",
236 )
237
238 # Create Imputation List
239
240 site_vector = list(x2["site_id"])
241 date_vector = list(x2["date"])
242 index_list = []
243 df.reset_index(inplace=True)
244 for i in range(len(site_vector)):
245     x = (
246         df.loc[(df["site_id"] == site_vector[i]) & (df["date"] == date_vector[i])]
247         .sort_values("sales", ascending=False)
248         .head(1)
249     )
250     x = x.drop(
251         columns=[
252             "location_id",
253             "open_date",
254             "sq_footage",
255             "locale",
256             "maxtemp",
257             "mintemp",
258             "fiscal_period",
259             "periodic_gbv",
260             "current_gbv",
261             "mpds",
262         ]

```

```

263     )
264     index_list.append(x.iloc[0, 0])
265
266 df_bu = df
267
268 df.rename(columns={df.columns[0]: "index_set"}, inplace=True)
269
270 df = df[~df.index_set.isin(index_list)]
271
272 del index_list
273
274 temp2 = (
275     df.groupby(["date"])
276     .agg(
277         daily_sales=pd.NamedAgg(column="sales", aggfunc=sum),
278         daily_quantity=pd.NamedAgg(column="quantity_sold", aggfunc=sum),
279     )
280     .reset_index()
281 )
282 # Plot
283 fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(8, 6), sharex=True, sharey=True)
284 fig.suptitle("Aggregate daily sales over all sites before and after outlier removal")
285
286 ax1.plot(temp.date, temp.daily_sales, color="#bf021f", linewidth=0.75)
287 ax1.tick_params("x", labelrotation=45)
288 ax1.set_ylabel("Agg. Sales before outlier removal")
289
290 ax2.plot(temp2.date, temp2.daily_sales, color="#bf021f", linewidth=0.75)
291 ax2.set_xlabel("Date")
292 ax2.tick_params("x", labelrotation=45)
293 ax2.set_ylabel("Agg. Sales after outlier removal")
294
295 plt.show()
296
297 # In[12]
298
299 # Run ETS
300 from statsmodels.tsa.exponential_smoothing.ets import ETSModel
301
302
303 #model = ETSModel(df.quantity_sold.astype('float64'))
304 #fit = model.fit(maxiter=5)
305 #print(fit.summary())
306
307 # In[13]
308
309 # Test for Seasonality and Trends and Visualaize them
310 from statsmodels.tsa.seasonal import seasonal_decompose
311
312 input_df = pd.DataFrame()
313 dfx = df.loc[
314     (df.product_name == "MAR KG 3.29z SNICKERS 2 PIECE") & (df.site_id == 280)
315 ].reset_index(drop=True)
316
317 sd = seasonal_decompose(dfx.quantity_sold, period=7)
318 input_df["date"] = dfx.date
319 input_df["quantity_sold"] = dfx.quantity_sold
320 input_df["observed"] = sd.observed
321 input_df["residual"] = sd.resid
322 input_df["seasonal"] = sd.seasonal
323 input_df["trend"] = sd.trend
324
325
326 def mround(x, m=5):
327     """Helper method for multiple round"""
328     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

```

```

395 input_df = pd.DataFrame()
396 dfx = df.loc[
397     (df.product_name == "MAR KG 3.29z SNICKERS 2 PIECE") & (df.site_id == 280)
398 ].reset_index(drop=True)
399
400 sd = seasonal_decompose(dfx.quantity_sold, period=365)
401 input_df["date"] = dfx.date
402 input_df["quantity_sold"] = dfx.quantity_sold
403 input_df["observed"] = sd.observed
404 input_df["residual"] = sd.resid
405 input_df["seasonal"] = sd.seasonal
406 input_df["trend"] = sd.trend
407
408
409 def mround(x, m=5):
410     """Helper method for multiple round"""
411     return int(m * round(float(x) / m))
412
413
414 def plot_components(df):
415     """Plot data for initial visualization, ultimately visualized in Power BI
416     Args:
417         df (pandas dataframe)
418     """
419     df_axis = df.fillna(0)
420     ymin = mround(
421         np.min([df_axis.observed, df_axis.trend, df_axis.seasonal, df_axis.residual]), 5
422     )
423     ymax = mround(
424         np.max([df_axis.observed, df_axis.trend, df_axis.seasonal, df_axis.residual]), 5
425     )
426     ymin -= 5
427     ymax += 5
428
429     plt.figure(figsize=(20, 20))
430     plt.subplot(4, 1, 1)
431     plt.title(
432         "Original Data [Site_id: 280, Item: MAR KG 3.29z SNICKERS 2 PIECE] Period = 365",
433         fontsize=30,
434     )
435     plt.ylim(ymin, ymax)
436     plt.xticks(fontsize=20)
437     plt.yticks(fontsize=20)
438     plt.plot(df.index, df.observed, color="#bf021f", linewidth=0.75)
439
440     plt.subplot(4, 1, 2)
441     plt.title(
442         "Trend [Site_id: 280, Item: MAR KG 3.29z SNICKERS 2 PIECE] Period = 365",
443         fontsize=30,
444     )
445     plt.ylim(ymin, ymax)
446     plt.xticks(fontsize=20)
447     plt.yticks(fontsize=20)
448     plt.plot(df.index, df.trend, color="#bf021f", linewidth=0.75)
449
450     plt.subplot(4, 1, 3)
451     plt.title(
452         "Seasonal [Site_id: 280, Item: MAR KG 3.29z SNICKERS 2 PIECE] Period = 365",
453         fontsize=30,
454     )
455     plt.ylim(ymin, ymax)
456     plt.xticks(fontsize=20)
457     plt.yticks(fontsize=20)
458     plt.plot(df.index, df.seasonal, color="#bf021f", linewidth=0.75)
459
460     plt.subplot(4, 1, 4)

```

```

461     plt.title(
462         "Residual [Site_id: 280, Item: MAR KG 3.29z SNICKERS 2 PIECE] Period = 365",
463         fontsize=30,
464     )
465     plt.xticks(fontsize=20)
466     plt.yticks(fontsize=20)
467     plt.ylim(ymin, ymax)
468     plt.ylim(ymin, ymax)
469     plt.plot(df.index, df.residual, color="#bf021f", linewidth=0.75)
470
471     plt.tight_layout(pad=1.0, w_pad=1.0, h_pad=1.0)
472
473
474 plot_components(input_df)
475
476 # In[15]:
477
478 model = ETSMModel(dfx.quantity_sold.astype('float64'))
479 fit = model.fit(maxiter=10)
480 print(fit.summary())
481
482
483 # In[16]:
484
485
486
487 input_df1 = pd.DataFrame()
488 dfx = df.loc[
489     (df.product_name == "MAR KG 3.29z SNICKERS 2 PIECE") & (df.site_id == 280)
490 ].reset_index(drop=True)
491
492 model = ETSMModel(dfx.quantity_sold.astype('float64'))
493 fit = model.fit(maxiter=10)
494 print(fit.summary())
495
496 sd = seasonal_decompose(dfx.quantity_sold, period=7)
497 input_df1["date"] = dfx.date
498 input_df1["quantity_sold"] = dfx.quantity_sold
499 input_df1["observed"] = sd.observed
500 input_df1["residual"] = sd.resid
501 input_df1["seasonal"] = sd.seasonal
502 input_df1["trend"] = sd.trend
503
504 input_df2 = pd.DataFrame()
505 dfx = df.loc[
506     (df.product_name == "HER KG 2.8z REESES PB CUP") & (df.site_id == 580)
507 ].reset_index(drop=True)
508
509 model = ETSMModel(dfx.quantity_sold.astype('float64'))
510 fit = model.fit(maxiter=10)
511 print(fit.summary())
512
513 sd = seasonal_decompose(dfx.quantity_sold, period=7)
514 input_df2["date"] = dfx.date
515 input_df2["quantity_sold"] = dfx.quantity_sold
516 input_df2["observed"] = sd.observed
517 input_df2["residual"] = sd.resid
518 input_df2["seasonal"] = sd.seasonal
519 input_df2["trend"] = sd.trend
520
521 input_df3 = pd.DataFrame()
522 dfx = df.loc[
523     (df.product_name == "TTS TOOTSIE ROLL $.10") & (df.site_id == 380)
524 ].reset_index(drop=True)
525
526 model = ETSMModel(dfx.quantity_sold.astype('float64'))

```



```

527 fit = model.fit(maxiter=10)
528 print(fit.summary())
529
530 sd = seasonal_decompose(dfx.quantity_sold, period=7)
531 input_df3["date"] = dfx.date
532 input_df3["quantity_sold"] = dfx.quantity_sold
533 input_df3["observed"] = sd.observed
534 input_df3["residual"] = sd.resid
535 input_df3["seasonal"] = sd.seasonal
536 input_df3["trend"] = sd.trend
537
538 input_df4 = pd.DataFrame()
539 dfx = df.loc[
540     (df.product_name == "HER KG 3z KIT KAT") & (df.site_id == 399)
541 ].reset_index(drop=True)
542
543 model = ETSModel(dfx.quantity_sold.astype('float64'))
544 fit = model.fit(maxiter=10)
545 print(fit.summary())
546
547 sd = seasonal_decompose(dfx.quantity_sold, period=7)
548 input_df4["date"] = dfx.date
549 input_df4["quantity_sold"] = dfx.quantity_sold
550 input_df4["observed"] = sd.observed
551 input_df4["residual"] = sd.resid
552 input_df4["seasonal"] = sd.seasonal
553 input_df4["trend"] = sd.trend
554
555
556
557 plt.figure(figsize=(20, 25))
558 plt.subplot(4, 1, 1)
559 plt.title(
560     "Seasonal [Site_id: 280, Item: MAR KG 3.29z SNICKERS 2 PIECE] Period = 7",
561     fontsize=30,
562 )
563 plt.ylim(-25, 25)
564 plt.xticks(fontsize = 20)
565 plt.yticks(fontsize = 20)
566 plt.plot(input_df1.index, input_df1.seasonal, color="#bf021f", linewidth=0.5)
567 plt.plot(input_df1.index, [0]*len(input_df1.index), color="black", linewidth=0.5)
568
569 plt.subplot(4, 1, 2)
570 plt.title(
571     "Seasonal [Site_id: 580, Item: HER KG 2.8z REESES PB CUP] Period = 7",
572     fontsize=30,
573 )
574 plt.ylim(-25, 25)
575 plt.xticks(fontsize = 20)
576 plt.yticks(fontsize = 20)
577 plt.plot(input_df2.index, input_df2.seasonal, color="#bf021f", linewidth=0.5)
578 plt.plot(input_df2.index, [0]*len(input_df2.index), color="black", linewidth=0.5)
579
580 plt.subplot(4, 1, 3)
581 plt.title(
582     "Seasonal [Site_id: 380, Item: TTS TOOTSIE ROLL $.10] Period = 7",
583     fontsize=30,
584 )
585 plt.ylim(-25, 25)
586 plt.xticks(fontsize = 20)
587 plt.yticks(fontsize = 20)
588 plt.plot(input_df3.index, input_df3.seasonal, color="#bf021f", linewidth=0.5)
589 plt.plot(input_df3.index, [0]*len(input_df3.index), color="black", linewidth=0.5)
590
591 plt.subplot(4, 1, 4)
592 plt.title(

```

```
593     "Seasonal [Site_id: 399, Item: HER KG 3z KIT KAT] Period = 7",
594     fontsize=30,
595 )
596 plt.ylim(-25, 25)
597 plt.xticks(fontsize = 20)
598 plt.yticks(fontsize = 20)
599 plt.plot(input_df4.index, input_df4.seasonal, color="#bf021f", linewidth=0.5)
600 plt.plot(input_df4.index, [0]*len(input_df4.index), color="black", linewidth=0.5)
```