# Contraceptive Time Series Analysis and Forecast

January 24, 2022

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

[ ]:
```
df= pd.read_csv('./data/Train.csv')
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35753 entries, 0 to 35752
Data columns (total 14 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   year                         35753 non-null  int64
 1   month                        35753 non-null  int64
 2   region                       35753 non-null  object
 3   district                     35753 non-null  object
 4   site_code                    35753 non-null  object
 5   product_code                 35753 non-null  object
 6   stock_initial                35753 non-null  int64
 7   stock_received               35753 non-null  int64
 8   stock_distributed            35753 non-null  int64
 9   stock_adjustment             35753 non-null  int64
 10  stock_end                    35753 non-null  int64
 11  average_monthly_consumption  35753 non-null  int64
 12  stock_stockout_days          35753 non-null  int64
 13  stock_ordered                34990 non-null  float64
dtypes: float64(1), int64(9), object(4)
memory usage: 3.8+ MB
```

[ ]:
```
df.describe()
```

[ ]:
```
               year         month  stock_initial  stock_received  \
count  35753.000000  35753.000000   35753.000000    35753.000000
mean    2017.433782      6.169412      63.245518       14.846055
std        1.019933      3.429079     168.661538       70.631782
min     2016.000000      1.000000       0.000000        0.000000
```

```
25%     2017.000000     3.000000      0.000000       0.000000
50%     2017.000000     6.000000     12.000000       0.000000
75%     2018.000000     9.000000     69.000000       0.000000
max     2019.000000    12.000000   4320.000000    3534.000000

       stock_distributed  stock_adjustment     stock_end  \
count       35753.000000      35753.000000  35753.000000
mean           14.764327          0.961150     64.288395
std            39.848242         37.883099    170.848479
min             0.000000      -1440.000000      0.000000
25%             0.000000          0.000000      0.000000
50%             1.000000          0.000000     13.000000
75%            13.000000          0.000000     70.000000
max          1728.000000       3003.000000   4320.000000

       average_monthly_consumption  stock_stockout_days  stock_ordered
count                 35753.000000              35753.0   34990.000000
mean                     14.606439                  0.0      26.658102
std                      32.521384                  0.0     107.166082
min                       0.000000                  0.0       0.000000
25%                       0.000000                  0.0       0.000000
50%                       3.000000                  0.0       0.000000
75%                      16.000000                  0.0      20.000000
max                     864.000000                  0.0   10240.000000
```

### 0.0.1 Filling Nulls with Mean

[ ]:
```
df.fillna(round(df.mean()), inplace=True, axis=0)
```

[ ]:
```
df.iloc[436]
```

[ ]:
```
year                                             2019
month                                               1
region                         ABIDJAN 1-GRANDS PONTS
district                    ADJAME-PLATEAU-ATTECOUBE
site_code                                       C1034
product_code                                  AS27134
stock_initial                                       0
stock_received                                      0
stock_distributed                                   0
stock_adjustment                                    0
stock_end                                           0
average_monthly_consumption                         9
stock_stockout_days                                 0
stock_ordered                                    27.0
Name: 436, dtype: object
```

### 0.0.2 Creating a Time Feature from the Month and Year Feature

[ ]:
```
# Creating a date feature

df['date'] = pd.to_datetime(df[['year', 'month']].assign(DAY=28))
```

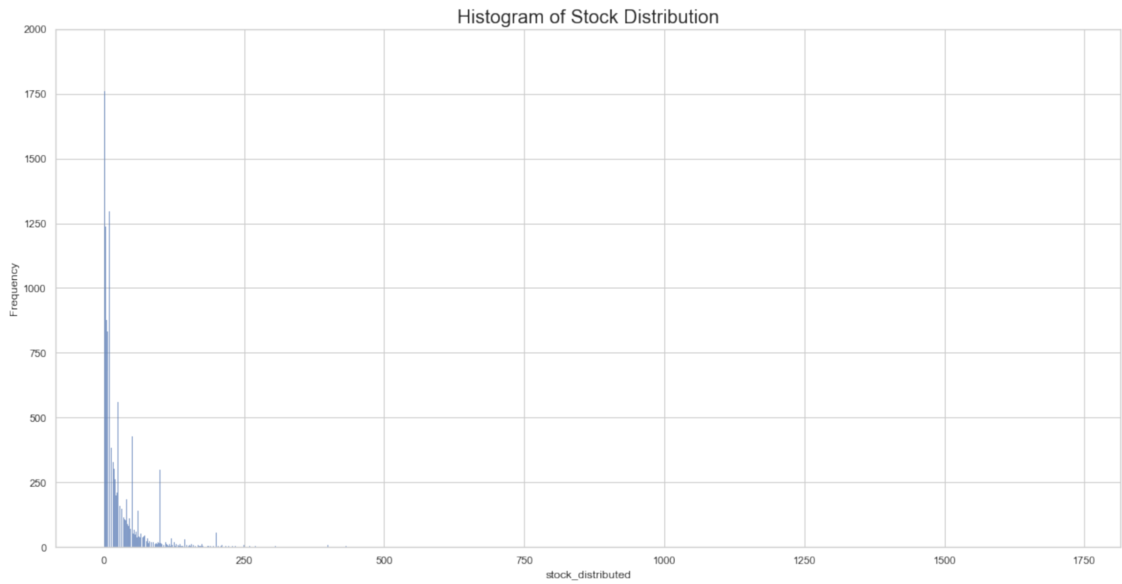
#### Inversing and Sorting the DataFrame to start with the smallest date

[ ]:
```
df = df[::-1]
```

[ ]:
```
df = df.sort_values(["year", "month"], ascending = (True, True))
```

#### Distribution of Stock Distributed

[ ]:
```
plt.figure(figsize=(20, 10))
sns.set_theme(style="whitegrid")
sns.histplot(data=df, x="stock_distributed", stat='frequency')
plt.ylim(0, 2000)
plt.title(
    'Histogram of Stock Distribution', fontsize=20)
plt.show()
```




Clearly, there are outliers and this informs the choice of using RMSE in measuring error

[ ]:
```
df.groupby(['date','month','site_code','product_code']).sum().reset_index().
 ↪set_index('date')
```

[ ]:
```
            month site_code product_code  year  stock_initial  stock_received  \
date
```

```
2016-01-28      1      C1008      AS27000  2016            127            0
2016-01-28      1      C1008      AS27132  2016             15            0
2016-01-28      1      C1008      AS27133  2016              0          100
2016-01-28      1      C1008      AS27134  2016             80          100
2016-01-28      1      C1008      AS27137  2016              0            0
…             …       …             …    …              …           …
2019-06-28      6      C5063      AS27138  2019             13            0
2019-06-28      6      C5063      AS46000  2019             85            0
2019-06-28      6      C5066      AS27133  2019             37            0
2019-06-28      6      C5066      AS27137  2019              9            0
2019-06-28      6      C5066      AS27138  2019              3            0

            stock_distributed  stock_adjustment  stock_end  \
date
2016-01-28                 90                90        127
2016-01-28                 15                 0          0
2016-01-28                 50               -50          0
2016-01-28                 15               -85         80
2016-01-28                  0                 0          0
…                       …                …        …
2019-06-28                  2                 0         11
2019-06-28                 12                 0         73
2019-06-28                 28                 0          9
2019-06-28                  0                 0          9
2019-06-28                  3                 0          0

            average_monthly_consumption  stock_stockout_days  stock_ordered
date
2016-01-28                           90                    0          100.0
2016-01-28                           15                    0            0.0
2016-01-28                           50                    0            0.0
2016-01-28                           15                    0            0.0
2016-01-28                            0                    0            0.0
…                                  …                   …            …
2019-06-28                            5                    0            4.0
2019-06-28                            4                    0            0.0
2019-06-28                           22                    0           57.0
2019-06-28                            0                    0            0.0
2019-06-28                            8                    0           24.0

[35753 rows x 12 columns]
```

### 0.0.3 Top Averages of Site Contraceptive Consumption Across the Period(2016-2019)

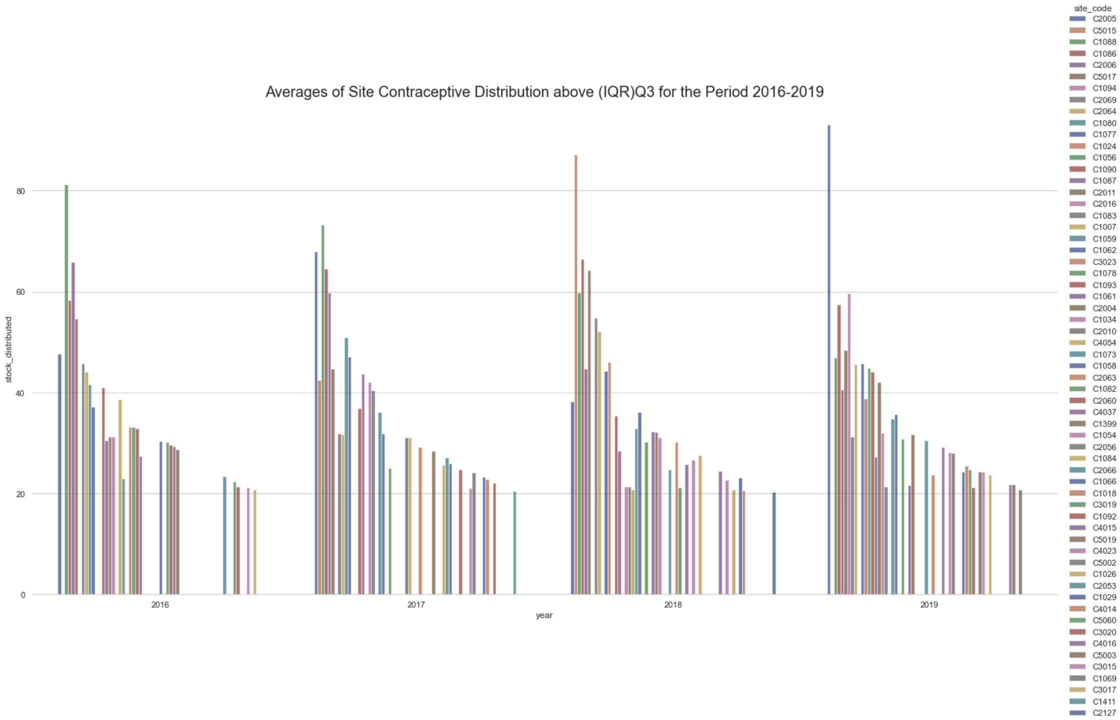
[ ]:
```
df_rank_site_avg = df.groupby(['year', 'site_code']).mean().reset_index()[
    ['year', 'site_code', 'stock_distributed']].
 ↪sort_values(['stock_distributed'], ascending=False)
```

[ ]:
```
df_rank_site_avg.describe()
```

[ ]:

| | year | stock_distributed |
|---|---|---|
| count | 583.000000 | 583.000000 |
| mean | 2017.555746 | 14.816789 |
| std | 1.115294 | 13.914502 |
| min | 2016.000000 | 0.000000 |
| 25% | 2017.000000 | 5.743421 |
| 50% | 2018.000000 | 10.920635 |
| 75% | 2019.000000 | 19.170170 |
| max | 2019.000000 | 93.083333 |

[ ]:
```
sns.set_theme(style="whitegrid")
g = sns.catplot(
    data=df_rank_site_avg[df_rank_site_avg['stock_distributed']
                              > 20], kind="bar",
    x="year", y="stock_distributed", hue="site_code",
    ci="sd", palette="dark", alpha=.6, height=10, aspect=2
)
g.despine(left=True)
plt.title(
    'Averages of Site Contraceptive Distribution above (IQR)Q3 for the Period␣
 ↪2016-2019', fontsize=20)
plt.show()
```




### 0.0.4 Top Averages of District Contraceptive Consumption Across the Period(2016-2019)

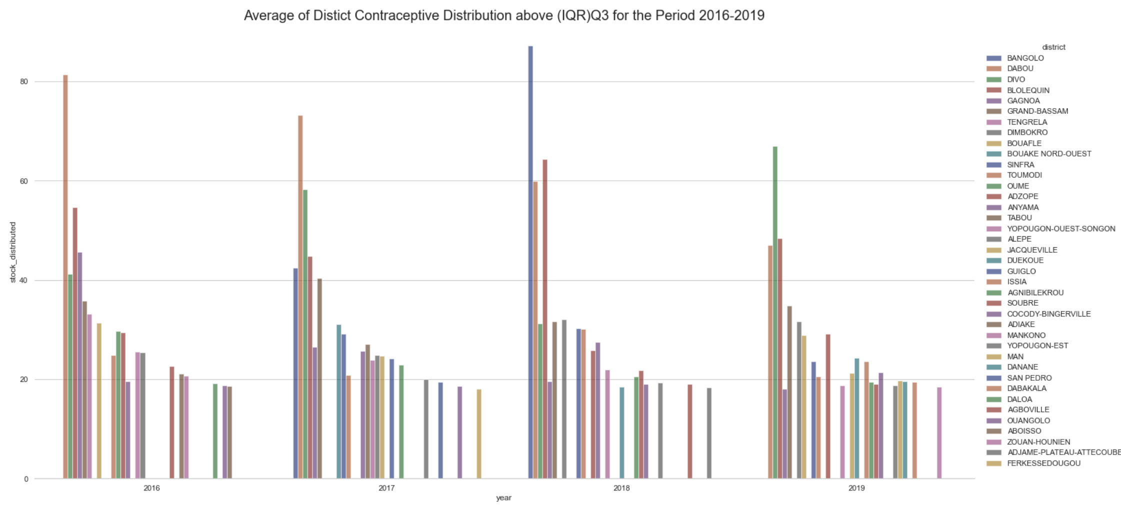
```
[ ]: df_rank_district_avg = df.groupby(['year','district',]).mean().reset_index()[
         ['year', 'district', 'stock_distributed']].
      ↪sort_values(['stock_distributed'], ascending=False)
```

```
[ ]: df_rank_district_avg.describe()
```

```
[ ]:               year  stock_distributed
     count   315.000000         315.000000
     mean   2017.523810          14.648553
     std       1.112425          12.362179
     min    2016.000000           0.000000
     25%    2017.000000           6.934524
     50%    2018.000000          11.222222
     75%    2019.000000          18.085442
     max    2019.000000          87.138889
```

```
[ ]: sns.set_theme(style="whitegrid")
     g = sns.catplot(
         data=df_rank_district_avg[df_rank_district_avg['stock_distributed']
                                    > 18], kind="bar",
```

```
    x="year", y="stock_distributed", hue="district",
    ci="sd", palette="dark", alpha=.6, height=10, aspect=2
)
g.despine(left=True)
plt.title(
    'Average of Distict Contraceptive Distribution above (IQR)Q3 for the Period␣
↪2016-2019', fontsize=20)
plt.show()
```




### 0.0.5 Ranking of District Total Contraceptive Consumption Across the Period(2016-2019)

[ ]:
```
df_rank_district = df.groupby(['district']).sum().reset_index()[
    ['district', 'stock_distributed']].sort_values(['stock_distributed'],␣
↪ascending=False)
```

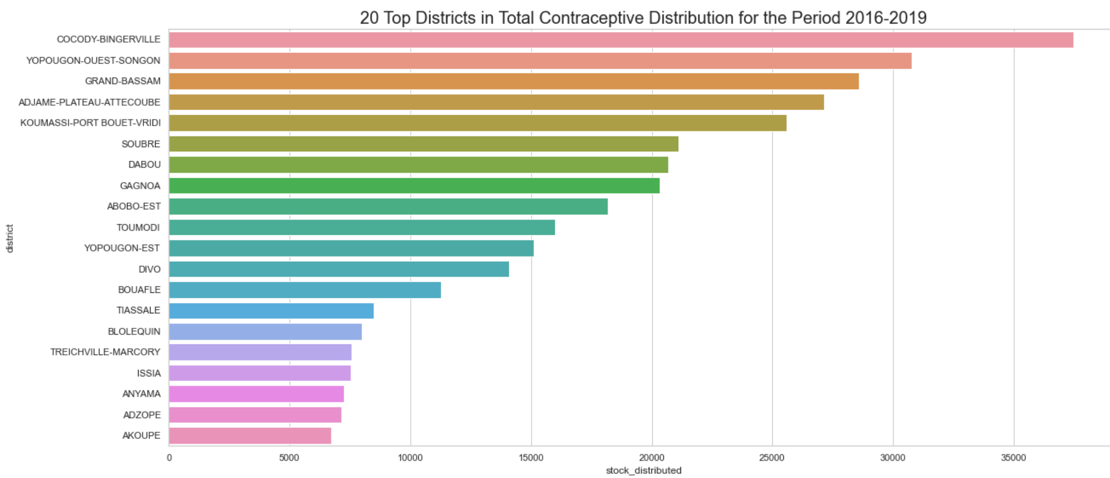
[ ]:
```
df_rank_district
```

[ ]:

| | district | stock_distributed |
|---|---|---|
| 23 | COCODY-BINGERVILLE | 37476 |
| 77 | YOPOUGON-OUEST-SONGON | 30781 |
| 36 | GRAND-BASSAM | 28599 |
| 4 | ADJAME-PLATEAU-ATTECOUBE | 27139 |
| 46 | KOUMASSI-PORT BOUET-VRIDI | 25598 |
| .. | … | … |
| 71 | TOULEPLEU | 1287 |
| 13 | BETTIE | 1188 |
| 62 | SIKENSI | 682 |
| 51 | MINIGNAN | 333 |
| 52 | NASSIAN | 300 |

[80 rows x 2 columns]

[ ]:
```
plt.figure(figsize=(20, 9))
sns.set_theme(style="whitegrid")
sns.barplot(data=df_rank_district[:20], orient='h',
            x='stock_distributed', y='district')
plt.title(
    '20 Top Districts in Total Contraceptive Distribution for the Period␣
 ↪2016-2019', fontsize=20)
plt.show()
```




### 0.0.6 Ranking of Site Total Contraceptive Consumption Across the Period(2016-2019)

[ ]:
```
df_rank_site = df.groupby(['site_code']).sum().reset_index()[
    ['site_code', 'stock_distributed']].sort_values(['stock_distributed'],␣
 ↪ascending=False)
```

[ ]:
```
df_rank_site
```

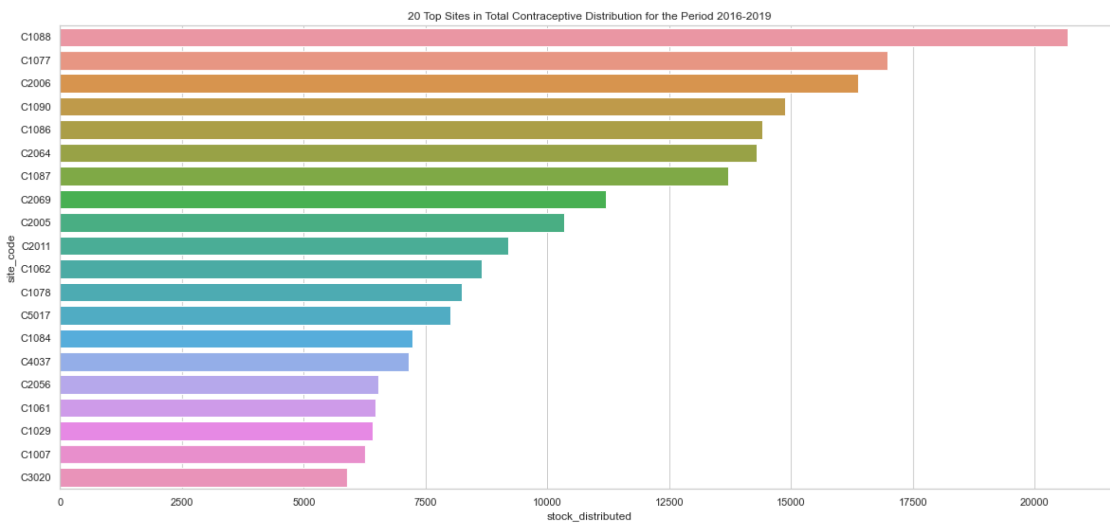
[ ]:

| | site_code | stock_distributed |
|---|---|---|
| 42 | C1088 | 20687 |
| 34 | C1077 | 16981 |
| 67 | C2006 | 16382 |
| 44 | C1090 | 14878 |
| 40 | C1086 | 14421 |
| .. | … | … |
| 110 | C3016 | 184 |
| 26 | C1063 | 159 |

```
61      C1701                  0
118     C3043                  0
115     C3021                  0

[155 rows x 2 columns]
```

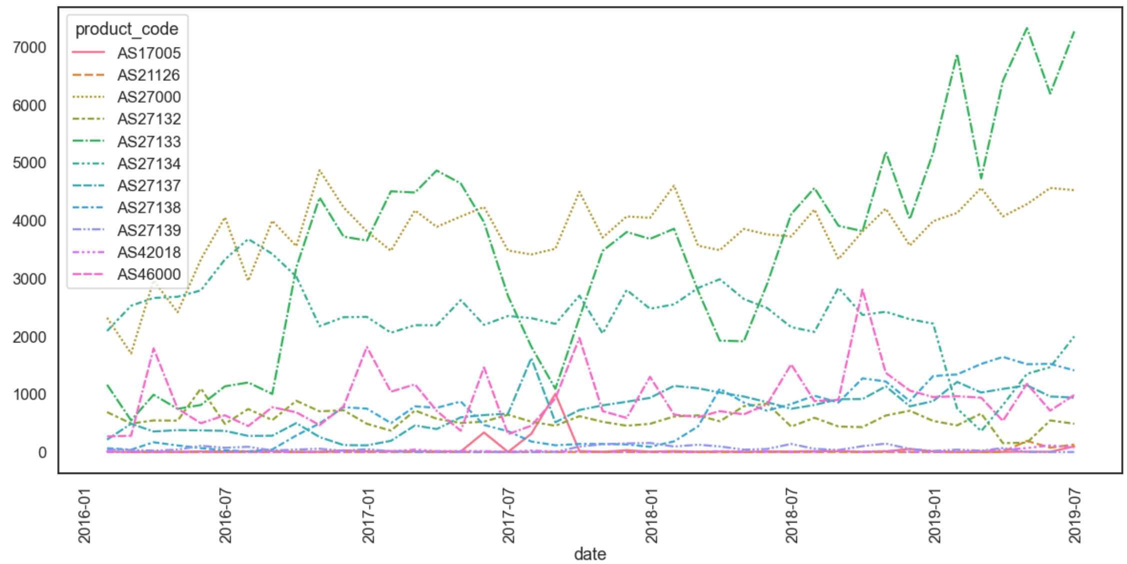
[ ]:
```
plt.figure(figsize=(20, 9))
sns.set_theme(style="whitegrid")
sns.barplot(data=df_rank_site[:20], orient='h', x='stock_distributed',␣
↪y='site_code')
plt.title('20 Top Sites in Total Contraceptive Distribution for the Period␣
↪2016-2019')
plt.show()
```




### 0.0.7 Monthly Trend of Contraceptive Consumption Across the Period(2016-2019)

[ ]:
```
df_monthly = df.groupby(['date', 'month', 'product_code']).sum().
↪reset_index()[['date', 'product_code', 'stock_distributed']].pivot(
    index='date', columns='product_code', values='stock_distributed').
↪reset_index()
df_monthly.set_index('date', inplace=True)
```

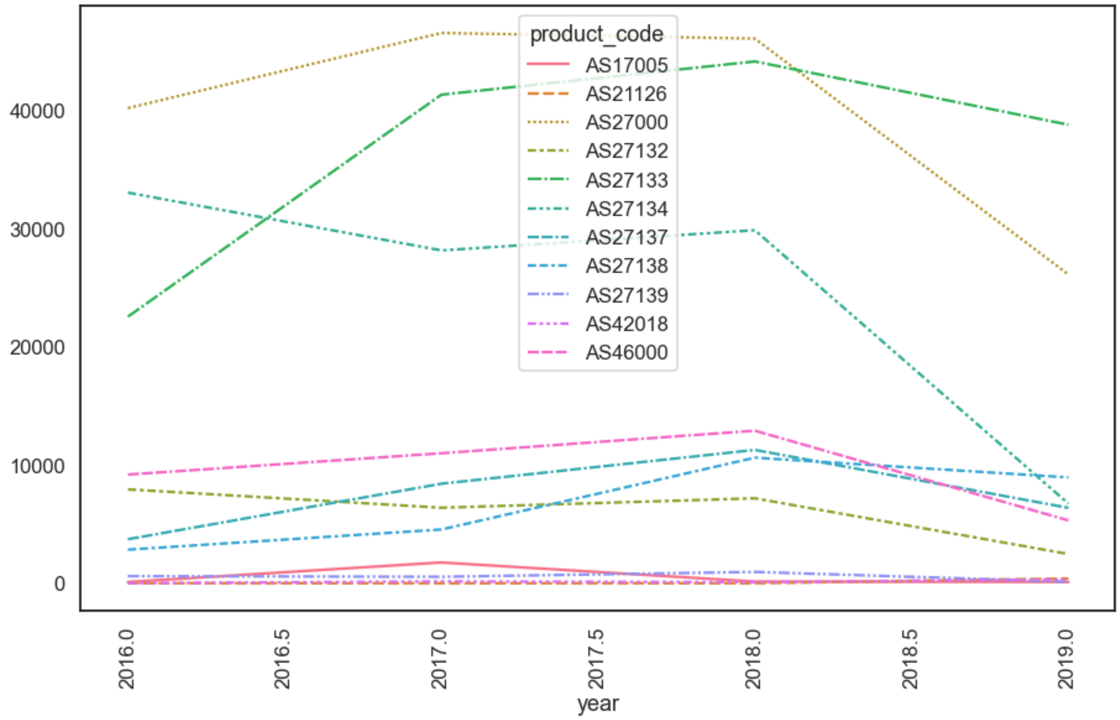
[ ]:
```
plt.figure(figsize=(20,9))
sns.set_theme(style= 'white', palette='dark', context='talk')
sns.lineplot(data=df_monthly)
plt.xticks(rotation=90)
plt.show()
```




### 0.0.8 Yearly Trend of Contraceptive Cumulative Consumption Across the Period(2016-2019)

[ ]:
```
df_yearly = df.groupby(['year', 'product_code']).sum().reset_index()[['year',␣
↪'product_code', 'stock_distributed']].pivot(
    index='year', columns='product_code', values='stock_distributed').
↪reset_index()
df_yearly.set_index('year', inplace=True)
```

[ ]:
```
plt.figure(figsize=(15, 9))
sns.set_theme(style='white', palette='dark', context='talk')
sns.lineplot(data=df_yearly)
plt.xticks(rotation=90)
plt.show()
```




From the two graphs, product AS21126 has data for a couple of months in 2019 and lacks for the rest. Therefore we will drop the site.

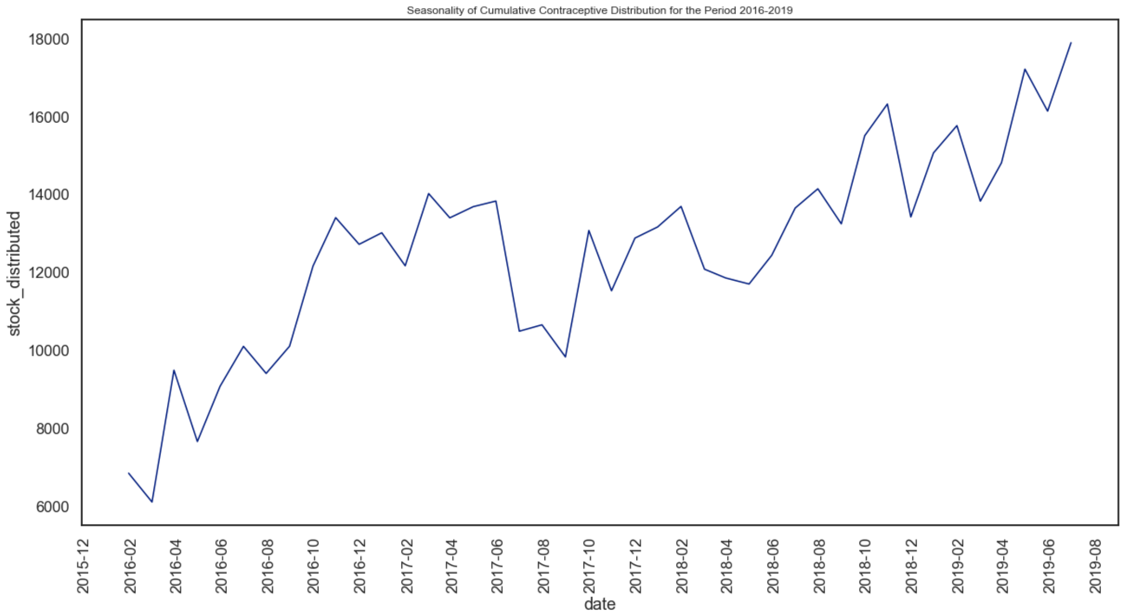
#### 0.0.9 Seasonality Accross Months for Cumulative Contraceptive Consumption Across the Period(2016-2019)

[ ]:
```
df_m_seasons = df.groupby(['date']).sum().reset_index()[
    ['date', 'stock_distributed']]
```

[ ]:
```
#df_m_seasons['month'] = pd.to_datetime(df_m_seasons['month'], format='%m').dt.
  ↪month_name()
```

[ ]:
```
import matplotlib.dates as mdates
```

[ ]:
```
fig = plt.figure(figsize=(20, 10))
ax = fig.add_subplot(1, 1, 1)
sns.set_theme(style="whitegrid")
sns.lineplot(data=df_m_seasons, x='date', y='stock_distributed')
plt.title('Seasonality of Cumulative Contraceptive Distribution for the Period␣
  ↪2016-2019')
ax.xaxis.set_major_locator(mdates.MonthLocator(interval=2))
plt.xticks(rotation=90)
plt.show()
```




#### 0.0.10 Average Distribution of Contraceptives Per Region Per Year

[ ]:
```
df_region = df.groupby(['region', 'year']).mean()[['stock_distributed']].
↪reset_index()
```

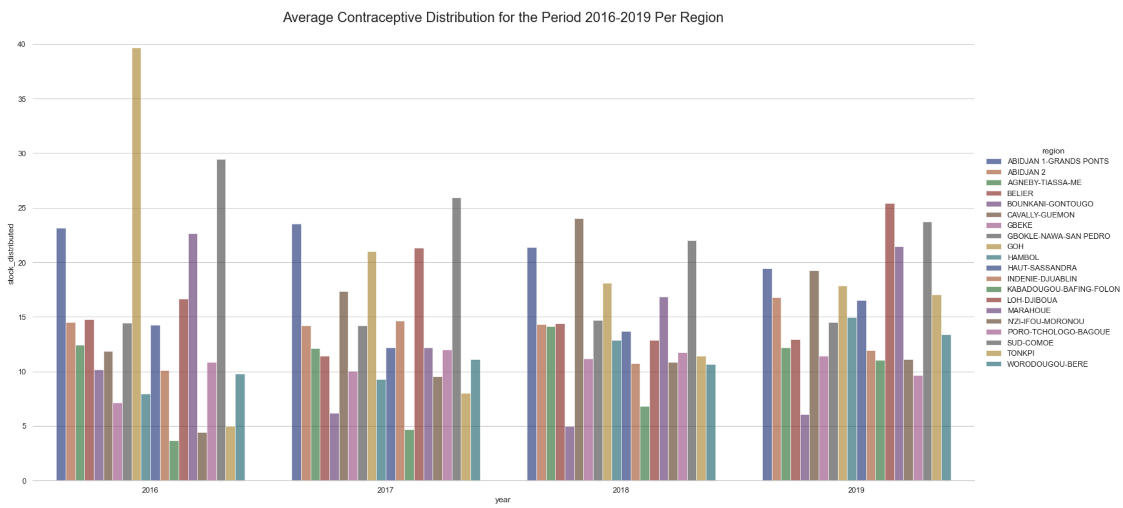
[ ]:
```
df_region
```

[ ]:
```
                    region  year  stock_distributed
0   ABIDJAN 1-GRANDS PONTS  2016          23.139018
1   ABIDJAN 1-GRANDS PONTS  2017          23.520583
2   ABIDJAN 1-GRANDS PONTS  2018          21.372521
3   ABIDJAN 1-GRANDS PONTS  2019          19.422340
4                ABIDJAN 2  2016          14.543696
..                     …   …                  …
75                  TONKPI  2019          17.051852
76         WORODOUGOU-BERE  2016           9.789474
77         WORODOUGOU-BERE  2017          11.144681
78         WORODOUGOU-BERE  2018          10.651639
79         WORODOUGOU-BERE  2019          13.373984

[80 rows x 3 columns]
```

[ ]:
```
sns.set_theme(style="whitegrid")
g = sns.catplot(
    data=df_region, kind="bar",
    x="year", y="stock_distributed", hue="region",
```

```
    ci="sd", palette="dark", alpha=.6, height=10, aspect=2
)
g.despine(left=True)
plt.title(
    'Average Contraceptive Distribution for the Period 2016-2019 Per Region',␣
  ↪fontsize=20)
plt.show()
```




### 0.0.11 Average Distribution of Contraceptives Per Product

[ ]:
```
df_product = df.groupby(['product_code', 'year']).mean()[
    ['stock_distributed']].reset_index()
```

[ ]:
```
df_product
```

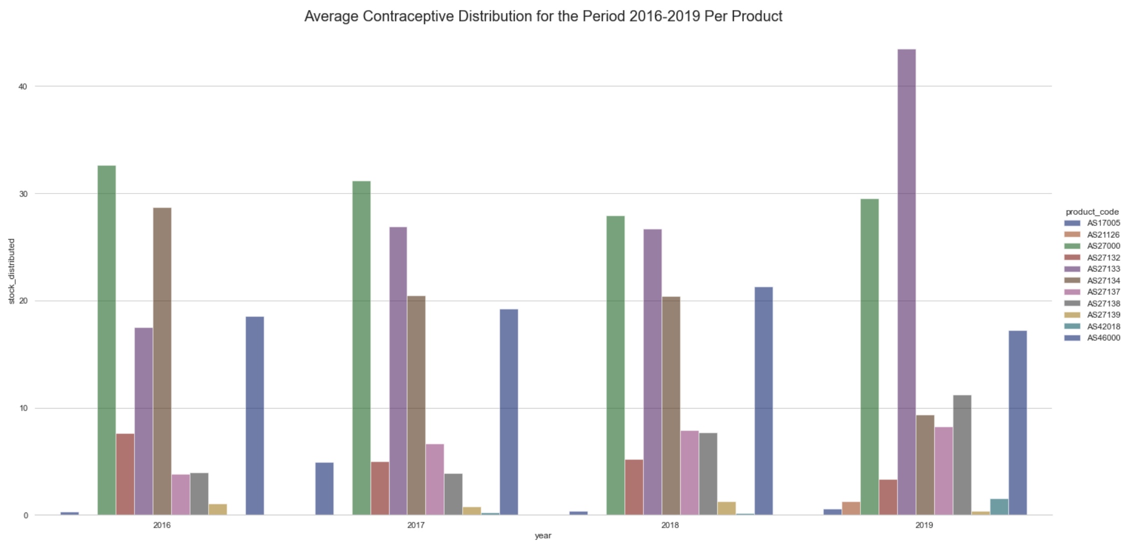
[ ]:

| | product_code | year | stock_distributed |
|---|---|---|---|
| 0 | AS17005 | 2016 | 0.284916 |
| 1 | AS17005 | 2017 | 4.918768 |
| 2 | AS17005 | 2018 | 0.396011 |
| 3 | AS17005 | 2019 | 0.587912 |
| 4 | AS21126 | 2016 | 0.000000 |
| 5 | AS21126 | 2018 | 0.000000 |
| 6 | AS21126 | 2019 | 1.255521 |
| 7 | AS27000 | 2016 | 32.616883 |
| 8 | AS27000 | 2017 | 31.174146 |
| 9 | AS27000 | 2018 | 27.932686 |
| 10 | AS27000 | 2019 | 29.511864 |
| 11 | AS27132 | 2016 | 7.595215 |
| 12 | AS27132 | 2017 | 5.024390 |
| 13 | AS27132 | 2018 | 5.192336 |

```
14      AS27132  2019          3.352782
15      AS27133  2016         17.487975
16      AS27133  2017         26.901693
17      AS27133  2018         26.730466
18      AS27133  2019         43.477578
19      AS27134  2016         28.702867
20      AS27134  2017         20.488355
21      AS27134  2018         20.380887
22      AS27134  2019          9.378830
23      AS27137  2016          3.816410
24      AS27137  2017          6.631994
25      AS27137  2018          7.878407
26      AS27137  2019          8.237726
27      AS27138  2016          3.980365
28      AS27138  2017          3.882906
29      AS27138  2018          7.700508
30      AS27138  2019         11.215539
31      AS27139  2016          1.086331
32      AS27139  2017          0.785303
33      AS27139  2018          1.282258
34      AS27139  2019          0.371105
35      AS42018  2016          0.023256
36      AS42018  2017          0.243156
37      AS42018  2018          0.172277
38      AS42018  2019          1.516746
39      AS46000  2016         18.528226
40      AS46000  2017         19.252189
41      AS46000  2018         21.323967
42      AS46000  2019         17.197411
```

[ ]:

```
sns.set_theme(style="whitegrid")
g = sns.catplot(
    data=df_product, kind="bar",
    x="year", y="stock_distributed", hue="product_code",
    ci="sd", palette="dark", alpha=.6, height=10, aspect=2
)
g.despine(left=True)
plt.title(
    'Average Contraceptive Distribution for the Period 2016-2019 Per Product',␣
 ↪fontsize=20)
plt.show()
```




[ ]:
```
df
```

[ ]:
```
        year  month                region              district site_code  \
35279   2016      1   ABIDJAN 1-GRANDS PONTS  YOPOUGON-OUEST-SONGON     C1028
35278   2016      1   ABIDJAN 1-GRANDS PONTS  YOPOUGON-OUEST-SONGON     C1028
35277   2016      1   ABIDJAN 1-GRANDS PONTS  YOPOUGON-OUEST-SONGON     C1028
35276   2016      1   ABIDJAN 1-GRANDS PONTS  YOPOUGON-OUEST-SONGON     C1028
35275   2016      1   ABIDJAN 1-GRANDS PONTS  YOPOUGON-OUEST-SONGON     C1028
…        …     …                      …                      …        …
47      2019      6         INDENIE-DJUABLIN             ABENGOUROU     C4001
46      2019      6         INDENIE-DJUABLIN             ABENGOUROU     C4001
45      2019      6         INDENIE-DJUABLIN             ABENGOUROU     C4001
44      2019      6         INDENIE-DJUABLIN             ABENGOUROU     C4001
43      2019      6         INDENIE-DJUABLIN             ABENGOUROU     C4001

      product_code  stock_initial  stock_received  stock_distributed  \
35279      AS46000              0               0                  0
35278      AS17005             10               0                  0
35277      AS27134              0               0                  0
35276      AS27132             45               0                  3
35275      AS27000             80               0                  9
…              …             …              …                 …
47         AS27137             18               0                  0
46         AS27000             19              10                  8
45         AS27132              4               5                  0
44         AS27134             67               0                 11
43         AS21126              0               0                  0

        stock_adjustment  stock_end  average_monthly_consumption  \
```

```
35279                      0          0                            0
35278                      0         10                            0
35277                      0          0                           18
35276                      0         42                            8
35275                      0         71                           33
…                     …         …                           …
47                         0         18                            1
46                         0         21                            8
45                         0          9                            0
44                         0         56                           15
43                         0          0                            0

       stock_stockout_days  stock_ordered       date
35279                    0            0.0 2016-01-28
35278                    0            0.0 2016-01-28
35277                    0            0.0 2016-01-28
35276                    0            0.0 2016-01-28
35275                    0            0.0 2016-01-28
…                      …            …        …
47                       0            0.0 2019-06-28
46                       0            0.0 2019-06-28
45                       0            0.0 2019-06-28
44                       0            0.0 2019-06-28
43                       0            0.0 2019-06-28

[35753 rows x 15 columns]
```

[ ]:
```python
# Removing the rows for the product code with only 2019 data
df_clean = df.drop(df[df['product_code'] == 'AS21126'].index, axis=0)
```

[ ]:
```python
# Knowing the distribution of value counts of the products across the years
df_clean['product_code'].value_counts()
```

[ ]:
```
AS27133    5368
AS27000    5259
AS27134    4708
AS27137    4449
AS27132    4436
AS27138    4060
AS27139    2347
AS46000    1981
AS42018    1550
AS17005    1248
Name: product_code, dtype: int64
```

[ ]:
```python
df_clean = df_clean.groupby(['date', 'year', 'month', 'site_code',␣
↪'product_code']).sum().reset_index()[[
```

```
    'date', 'year', 'month', 'site_code', 'product_code', 'stock_distributed']]
```

[ ]:
```
def create_df(data: pd.DataFrame):
    dates = data.date.unique()
    sites = data.site_code.unique()
    products = data.product_code.unique()
    missn = pd.DataFrame(
        columns=['date', 'site_code', 'product_code', 'stock_distributed'])

    for date in dates:
        for site in sites:
            temp = data[(data.date == date) & (data.site_code == site)]
            temp_prod = temp.product_code.unique()
            miss = list(set(products).difference(temp_prod))
            if len(miss) > 0:
                for val in miss:
                    missn = missn.append(
                        {'date': date, 'site_code': site, 'product_code': val,␣
↪'stock_distributed': 0}, ignore_index=True)

    return missn
```

With an assumption of the missing data: 1. o(zero) value data was created where certain products were missing for the various sites hence the function create_df

[ ]:
```
# Creating of x Dataframe holding the data where certain products missed in␣
↪specific sites
x = create_df(df_clean)
```

[ ]:
```
# Merging the original dataframe with the data with certain products missing in␣
↪specific sites
df_clean = pd.concat([df_clean.reset_index()[
                     ['date', 'site_code', 'product_code',␣
↪'stock_distributed']], x])
```

[ ]:
```
df_clean.sort_values(['date', 'site_code', 'product_code'],
                     ascending=True, inplace=True)
```

[ ]:
```
df_clean.set_index('date', inplace=True)
```

[ ]:
```
df_clean.head()
```

[ ]:
```
           site_code product_code stock_distributed
date
2016-01-28     C1004      AS17005                 0
2016-01-28     C1004      AS27000                 0
2016-01-28     C1004      AS27132                 0
```

```
2016-01-28     C1004      AS27133                  0
2016-01-28     C1004      AS27134                  0
```

[ ]:
```
df_clean['product_code'].value_counts()
```

[ ]:
```
AS17005    6510
AS27000    6510
AS27132    6510
AS27133    6510
AS27134    6510
AS27137    6510
AS27138    6510
AS27139    6510
AS42018    6510
AS46000    6510
Name: product_code, dtype: int64
```

The value count distribution is now even accross all products

[ ]:
```
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import RobustScaler, MinMaxScaler
from sklearn.pipeline import Pipeline
```

[ ]:
```
cat_pipe = Pipeline([
    ('encoder', OrdinalEncoder(handle_unknown='error'))
])

num_pipe = Pipeline([
    ('scaler_1', RobustScaler()),
    ('scaler_2', MinMaxScaler())
])
```

[ ]:
```
# Ordinal encoding for the site_code and product_code features
site_pro_trsm = cat_pipe.fit_transform(df_clean.drop(['stock_distributed'],␣
 ↪axis=1))
```

[ ]:
```
# Appending the transformed site_code and product_code features as new features
df_clean['site'] = site_pro_trsm[:, 0]
df_clean['product'] = site_pro_trsm[:, 1]
```

[ ]:
```
# Robust Scaling to remove outliers in the features considering␣
 ↪stock_distributed has large outlier figures
all_trsm = num_pipe.fit_transform(df_clean[['site',␣
 ↪'product','stock_distributed']])
```

[ ]:
```
# Appending the robust scaled features
df_clean['site'] = all_trsm[:, 0]
```

```
df_clean['product'] = all_trsm[:, 1]
df_clean['stock'] = all_trsm[:, 2]
```

[ ]:
```
df_clean
```

[ ]:
```
            site_code product_code stock_distributed  site   product     stock
date
2016-01-28      C1004      AS17005                 0   0.0  0.000000  0.000000
2016-01-28      C1004      AS27000                 0   0.0  0.111111  0.000000
2016-01-28      C1004      AS27132                 0   0.0  0.222222  0.000000
2016-01-28      C1004      AS27133                 0   0.0  0.333333  0.000000
2016-01-28      C1004      AS27134                 0   0.0  0.444444  0.000000
…             …         …                  …     …      …         …
2019-06-28      C5066      AS27137                 0   1.0  0.555556  0.000000
2019-06-28      C5066      AS27138                 3   1.0  0.666667  0.001736
2019-06-28      C5066      AS27139                 0   1.0  0.777778  0.000000
2019-06-28      C5066      AS42018                 0   1.0  0.888889  0.000000
2019-06-28      C5066      AS46000                 0   1.0  1.000000  0.000000

[65100 rows x 6 columns]
```

[ ]:
```
def create_seq(dataset):
    seq = []
    labels = []
    start_idx = 0

    for stop_idx in range(1550, len(dataset)):
        seq.append(dataset.iloc[start_idx:stop_idx][['site', 'product',␣
↪'stock']])
        labels.append(dataset['stock'][stop_idx])
        start_idx += 1
    return np.array(seq), np.array(labels)
```

[ ]:
```
train_size = round(42*0.80)*1550 #Number of months - 42, Proportion of test␣
↪size=0.80, # Records per month - 1550
```

[ ]:
```
train_data = df_clean[:train_size+1]
test_data = df_clean[train_size+1:]
```

[ ]:
```
train_seq, train_label = create_seq(train_data)
test_seq, test_label = create_seq(test_data)
```

[ ]:
```
test_seq.shape
```

[ ]:
```
(10849, 1550, 3)
```

**Storing some Elements for Later Use in Testing**

[ ]:
```python
import joblib
```

[ ]:
```python
data_dic = {'train_seq': train_seq, 'train_label': train_label, 'test_seq':␣
↪test_seq, 'test_label': test_label}
```

[ ]:
```python
joblib.dump(data_dic, './elements/procssd_data.joblib')
```

[ ]:
```
['./elements/procssd_data.joblib']
```

[ ]:
```python
joblib.dump(num_pipe, './elements/num_pipe.joblib')
```

[ ]:
```
['./elements/num_pipe.joblib']
```

[ ]:
```python
df_clean.to_csv('./elements/df_clean.csv')
```

### Modelling to Forecast

[ ]:
```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, InputLayer, Conv1D,␣
↪MaxPooling1D
```

```
2022-01-24 11:29:14.126606: W
tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load
dynamic library 'libcudart.so.11.0'; dlerror: libcudart.so.11.0: cannot open
shared object file: No such file or directory
2022-01-24 11:29:14.126796: I tensorflow/stream_executor/cuda/cudart_stub.cc:29]
Ignore above cudart dlerror if you do not have a GPU set up on your machine.
```

[ ]:
```python
model = Sequential()
model.add(InputLayer(input_shape=(None, 3,),
          batch_size=16, name='input_layer'))
model.add(Conv1D(64, kernel_size=2, padding='same',
          activation='relu', name='conv1d_1'))
model.add(MaxPooling1D(1, padding='same', name='maxpool_1'))
model.add(LSTM(units=512, name='lstm_1', return_sequences=True))
model.add(LSTM(units=256, name='lstm_2', return_sequences=False))
model.add(Dense(64, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(1, activation='linear'))
```

[ ]:
```python
model.compile(loss='mean_squared_error', optimizer='adam',
              metrics=['mean_absolute_error'])
```

[ ]:
```python
tf.config.run_functions_eagerly(True)
tf.data.experimental.enable_debug_mode()
```

```
[ ]: checkpoint_cb = tf.keras.callbacks.ModelCheckpoint(
         "./model.h5", save_best_only=True, monitor='val_mean_absolute_error')
     early_stopping_cb = tf.keras.callbacks.EarlyStopping(
         patience=10, restore_best_weights=True)
```

```
[ ]: history = model.fit(train_seq, train_label, epochs=100,␣
      ↪validation_data=(test_seq, test_label),
                         callbacks=[checkpoint_cb, early_stopping_cb])
```

```
[ ]: import json
     json.dump(history.history, open('./model/history.json', 'w'), indent=4)
```

# Testing and Predicting

January 24, 2022

```
[ ]: import tensorflow as tf
     import numpy as np
     from sklearn.metrics import mean_squared_error
     import joblib
     import matplotlib.pyplot as plt
     import seaborn as sns
     import pandas as pd
```

**Loading the Necessary Elements**

```
[ ]: df_clean = pd.DataFrame('./elements/df_clean.csv', parse_dates=['date'])
```

```
[ ]: num_pipe = joblib.load('./elements/num_pipe.joblib')
```

```
[ ]: data_dic = joblib.load('./elements/procssd.joblib')
```

```
[ ]: tf.config.run_functions_eagerly(True)
     tf.data.experimental.enable_debug_mode()
```

```
[ ]: model = tf.keras.models.load_model('./model/model.h5')
```

```
[ ]: train_pred = model.predict(data_dic['train_seq'])
```

```
[ ]: test_pred = model.predict(data_dic['test_seq'])
```

```
[ ]: train_i_pred= num_pipe.inverse_transform(np.repeat(train_pred, 3, axis=1))
```

### 0.0.1 Testing the Model Performance

```
[ ]: train_rmse = np.sqrt(mean_squared_error(data_dic['train_label'], train_i_pred[:
     ↪, -1].reshape(-1, 1)))
```

```
[ ]: test_i_pred= num_pipe.inverse_transform(np.repeat(test_pred, 3, axis=1))
```

```
[ ]: test_rmse = np.sqrt(mean_squared_error(data_dic['test_label'], test_i_pred[:,␣
     ↪-1].reshape(-1, 1)))
```

```
[ ]: rmse_vals = pd.DataFrame({'Train RMSE': train_rmse, 'Test RMSE': test_rmse},␣
      ↪index=['RMSE']).T.rename(columns={0:'RMSE'})
```

```
[ ]: plt.figure(figsize=(4, 12))
     sns.set_theme(style="whitegrid")
     ax = sns.barplot(data=rmse_vals, x=rmse_vals.index, y='RMSE')
     plt.title('RMSE for Test and Train Data')
     plt.ylim(top=15)
     for p in ax.patches:
         # get the height of each bar
         height = p.get_height()
         # adding text to each bar
         ax.text(x=p.get_x()+(p.get_width()/2),  # x-coordinate position of data␣
      ↪label, padded to be in the middle of the bar
                 y=height+0.2,  # y-coordinate position of data label, padded 100␣
      ↪above bar
                 # data label, formatted to have 4 decimals
                 s='{: .2f}'.format(height),
                 ha='center')  # sets horizontal alignment (ha) to center
     plt.savefig('./rmse.png', dpi=300, format=None, metadata=None,
                 bbox_inches=None, pad_inches=0.1
                 )
     plt.show()
```

### 0.0.2 Predicting Into the Future(July, August, September)

```
[ ]: # 3months * 1550 = 4650
     # Predicting 3 months into the future
     jul_sep = model.predict(data_dic['test_seq'][-4650:])
```

```
[ ]: jul_sep = num_pipe.inverse_transform(np.repeat(jul_sep, 3, axis=1))[
         :, -1].reshape(-1, 1)
```

```
[ ]: f_dates = np.concatenate([np.repeat(pd.to_datetime('2019-7-28'), 1550, axis=0),␣
      ↪np.repeat(
         pd.to_datetime('2019-8-28'), 1550, axis=0), np.repeat(pd.
      ↪to_datetime('2019-9-28'), 1550, axis=0)], axis=0)
     f_site = df_clean[-4650:]['site_code'].values.tolist()
     f_product = df_clean[-4650:]['product_code'].values.tolist()
```

```
[ ]: submit = pd.DataFrame({'date': f_dates.tolist(), 'site_code': f_site,␣
      ↪'product_code': f_product, 'prediction':jul_sep.reshape(1, -1)[0]})
```

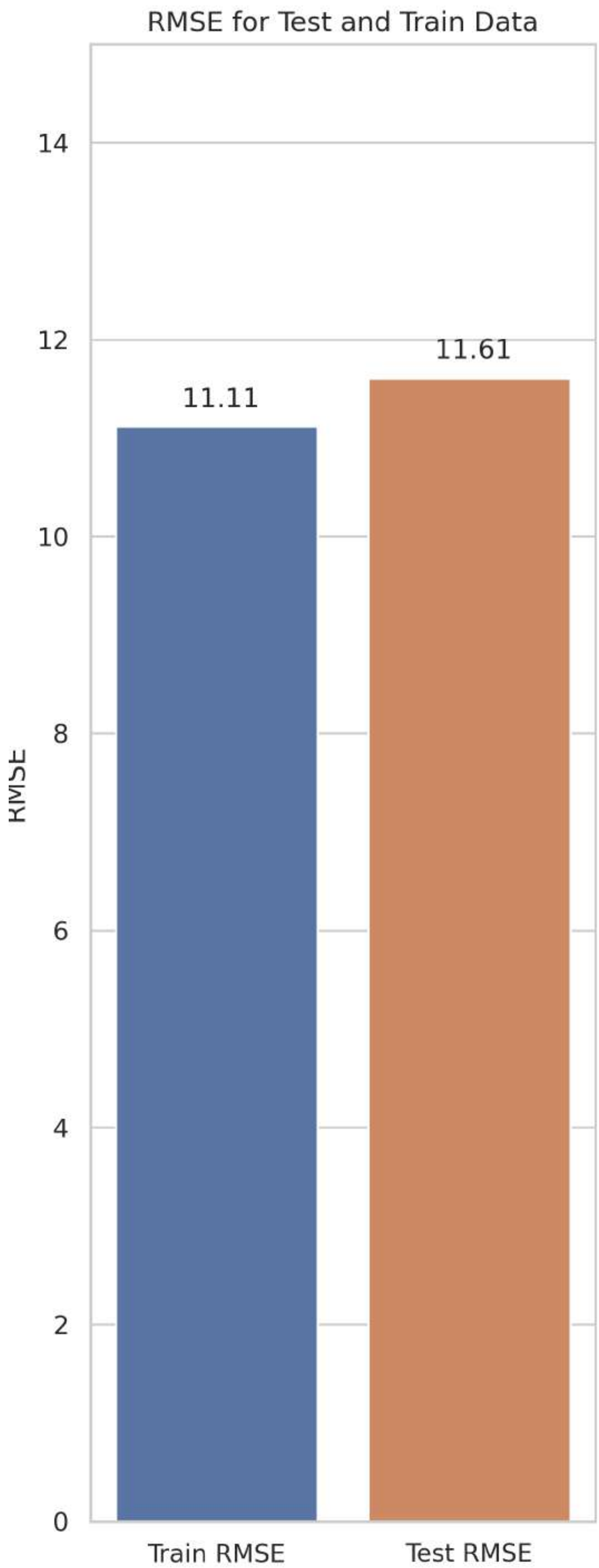
```
[ ]: def create_id(data: pd.DataFrame):
         id = []
```

```
    for i in range(len(data)):
        id_ = str(data['date'][i].year) + ' X ' + \
            str(data['date'][i].month) + ' X ' + \
            data['site_code'][i] + ' X ' + data['product_code'][i]
        id.append(id_)

    return id
```

[ ]:
```
submit['ID'] = create_id(submit)
```

[ ]:
```
submit[['ID', 'prediction']].to_csv('./submit.csv', index=False)
```


RMSE for Test and Train Data
14
12
10
8
6
4
2
0
RMSE
11.11
11.61
Train RMSE
Test RMSE