In [5]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
from dateutil import parser
```

Team

- Group Name: Lone Wolf
- Name: Gao Mo
- Email: david113mo@gmail.com (mailto:david113mo@gmail.com)
- Country: United States
- College: Carnegie Mellon University
- Specialization: Data Science

Project Objective:

1. Build atleast 4-5 multivariate forecasting model which included ML or Deep Learning based Model in PySpark leveraging parallel computing techniques(You can develop models without Pyspark if you are not comfortable with pyspark and parallel computing).
2. Demonstrate best in class forecast accuracy (Forecast Accuracy = 1 - Wt. MAPE where Wt. MAPE = sum(Error)/sum(Actual)
3. Write a code in such a way you run the model in least time
4. Demonstrate explainability in the form of contribution of each variables

Note:

- Leveage Feature Engineering concepts to derive more variables to gain accuracy improvement
- You can build model and demostrate accuracy for Q3-Q4 of 2020

Pre-Processing and Feature Engineering

In [34]:
```
data = pd.read_csv('forecasting.csv')
data['Price Discount (%)'] = data['Price Discount (%)'].apply(lambda
data.head()
```

Out[34]:

	Product	date	Sales	Price Discount (%)	In-Store Promo	Catalogue Promo	Store End Promo	Google_Mobility	Covid_Flag	V
0	SKU1	2/5/2017	27750	0	0	0	0	0.0	0	
1	SKU1	2/12/2017	29023	0	1	0	1	0.0	0	
2	SKU1	2/19/2017	45630	17	0	0	0	0.0	0	
3	SKU1	2/26/2017	26789	0	1	0	1	0.0	0	
4	SKU1	3/5/2017	41999	17	0	0	0	0.0	0	

In [35]:
```
data['Google_Mobility'] = data['Google_Mobility'].apply(lambda x: in
data.head()
```

Out[35]:

	Product	date	Sales	Price Discount (%)	In-Store Promo	Catalogue Promo	Store End Promo	Google_Mobility	Covid_Flag	V
0	SKU1	2/5/2017	27750	0	0	0	0	0	0	
1	SKU1	2/12/2017	29023	0	1	0	1	0	0	
2	SKU1	2/19/2017	45630	17	0	0	0	0	0	
3	SKU1	2/26/2017	26789	0	1	0	1	0	0	
4	SKU1	3/5/2017	41999	17	0	0	0	0	0	

In [36]:
```
data.date = data.date.apply(lambda x: parser.parse(x))
data.head()
```

Out[36]:

	Product	date	Sales	Price Discount (%)	In-Store Promo	Catalogue Promo	Store End Promo	Google_Mobility	Covid_Flag	V_D
0	SKU1	2017-02-05	27750	0	0	0	0	0	0	
1	SKU1	2017-02-12	29023	0	1	0	1	0	0	
2	SKU1	2017-02-19	45630	17	0	0	0	0	0	
3	SKU1	2017-02-26	26789	0	1	0	1	0	0	
4	SKU1	2017-03-05	41999	17	0	0	0	0	0	

In [22]:

```
mean_sales = data.Sales.describe().mean()
std_sales = data.Sales.describe().std()
outliers = mean_sales + 1.5*std_sales
data = data[data.Sales <= outliers]
data.head()
```

Out[22]:

	Product	date	Sales	Price Discount (%)	In-Store Promo	Catalogue Promo	Store End Promo	Google_Mobility	Covid_Flag	V_D
0	SKU1	2017-02-05	27750	0	0	0	0	0	0	
1	SKU1	2017-02-12	29023	0	1	0	1	0	0	
2	SKU1	2017-02-19	45630	17	0	0	0	0	0	
3	SKU1	2017-02-26	26789	0	1	0	1	0	0	
4	SKU1	2017-03-05	41999	17	0	0	0	0	0	

In [44]:

```
q3q4_2020 = parser.parse('2020-07-01')
train_df = data[data.date <= q3q4_2020]
test_df = data[data.date > q3q4_2020]
```

In [50]:

```
train_prod1 = train_df[train_df.Product == 'SKU1']
train_prod2 = train_df[train_df.Product == 'SKU2']
train_prod3 = train_df[train_df.Product == 'SKU3']
train_prod4 = train_df[train_df.Product == 'SKU4']
train_prod5 = train_df[train_df.Product == 'SKU5']
train_prod6 = train_df[train_df.Product == 'SKU6']

test_prod1 = test_df[test_df.Product == 'SKU1']
test_prod2 = test_df[test_df.Product == 'SKU2']
test_prod3 = test_df[test_df.Product == 'SKU3']
test_prod4 = test_df[test_df.Product == 'SKU4']
test_prod5 = test_df[test_df.Product == 'SKU5']
test_prod6 = test_df[test_df.Product == 'SKU6']
```

In [51]:

```
train_prod6.head()
```

Out[51]:

	Product	date	Sales	Price Discount (%)	In-Store Promo	Catalogue Promo	Store End Promo	Google_Mobility	Covid_Flag	V
1020	SKU6	2017-02-05	32138	28	0	0	0	0	0	
1021	SKU6	2017-02-12	11659	5	0	0	0	0	0	
1022	SKU6	2017-02-19	12140	5	1	0	1	0	0	
1023	SKU6	2017-02-26	29635	28	0	0	0	0	0	
1024	SKU6	2017-03-05	11666	5	0	1	1	0	0	

Model building requirements:

Select your base model and then explore 1 model of each family if its classification problem then 1 model for Linear models, 1- Model for Ensemble, 1-Model for boosting and other models if you have time (like stacking)

In [48]:

```
train_df.columns
```

Out[48]:

```
Index(['Product', 'date', 'Sales', 'Price Discount (%)', 'In-Store Prom
o',
       'Catalogue Promo', 'Store End Promo', 'Google_Mobility', 'Covid_
Flag',
       'V_DAY', 'EASTER', 'CHRISTMAS'],
      dtype='object')
```

In [53]:

```
feat_lst = ['Price Discount (%)','In-Store Promo',
       'Catalogue Promo', 'Store End Promo', 'Google_Mobility', 'Cov
       'V_DAY', 'EASTER', 'CHRISTMAS']
```

In [112]:

```
X = train_df[feat_lst].to_numpy()

y = train_df['Sales'].to_numpy()
```

In [62]:

```
test_x1 = test_prod1[feat_lst].to_numpy()
test_x2 = test_prod2[feat_lst].to_numpy()
test_x3 = test_prod3[feat_lst].to_numpy()
test_x4 = test_prod4[feat_lst].to_numpy()
test_x5 = test_prod5[feat_lst].to_numpy()
test_x6 = test_prod6[feat_lst].to_numpy()

test_y1 = test_prod1['Sales'].to_numpy()
test_y2 = test_prod2['Sales'].to_numpy()
test_y3 = test_prod3['Sales'].to_numpy()
test_y4 = test_prod4['Sales'].to_numpy()
test_y5 = test_prod5['Sales'].to_numpy()
test_y6 = test_prod6['Sales'].to_numpy()
```

1. Base Model

In [119]:

```
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier

base_model = Pipeline([('lin', LinearRegression())])
base_model.fit(X,y)
```

Out[119]:

```
Pipeline(steps=[('lin', LinearRegression())])
```

In [180]:

```
def forecast_accuracy(pred, label):
    error = [i - j for i,j in zip(pred,label)]
    mape = np.absolute(sum(error)) / sum(label)
    acc = 1 - mape
    return np.absolute(acc)
```

In [154]:

```python
feat_vals = base_model.named_steps['lin'].coef_
df_importances = pd.DataFrame(data = list(zip(feat_lst,feat_vals)),
df_importances.importance = df_importances.importance.apply(lambda x
sum_val = sum(list(df_importances.importance))
df_importances.importance = df_importances.importance.apply(lambda x
df_importances
```

Out[154]:

	feat	importance
0	Price Discount (%)	0.018384
1	In-Store Promo	0.102057
2	Catalogue Promo	0.178485
3	Store End Promo	0.243340
4	Google_Mobility	0.002008
5	Covid_Flag	0.346518
6	V_DAY	0.011078
7	EASTER	0.041167
8	CHRISTMAS	0.056963

In [175]:

```python
prediction = base_model.predict(test_x6)
forecast_accuracy(prediction, test_y6)
```

Out[175]:

```
0.8148778659709384
```

In [185]:

```python
test_Xs = [test_x1,test_x2,test_x3,test_x4,test_x5,test_x6]
test_ys = [test_y1,test_y2,test_y3,test_y4,test_y5,test_y6]

accs = []
for i,j in zip(test_Xs, test_ys):
    prediction = base_model.predict(i)
    accs.append(forecast_accuracy(prediction, j))

d = list(zip(['SKU1','SKU2','SKU3','SKU4','SKU5','SKU6'],accs))
df_acc = pd.DataFrame(data = d, columns=['Product', 'Forecast_Accura
df_acc
```

Out[185]:

	Product	Forecast_Accuracy
0	SKU1	0.365635
1	SKU2	0.162784
2	SKU3	0.607694
3	SKU4	0.817010
4	SKU5	0.956517
5	SKU6	0.814878

2. Linear Model

In [188]:

```
from sklearn.svm import SVC

lin_model = Pipeline([('svc', SVC(kernel='linear'))])
lin_model.fit(X,y)
```

Out[188]:

```
Pipeline(steps=[('svc', SVC(kernel='linear'))])
```

In [202]:

```
accs = []
for i,j in zip(test_Xs, test_ys):
    prediction = lin_model.predict(i)
    accs.append(forecast_accuracy(prediction, j))

d = list(zip(['SKU1','SKU2','SKU3','SKU4','SKU5','SKU6'],accs))
df_acc = pd.DataFrame(data = d, columns=['Product', 'Forecast_Accura
df_acc
```

Out[202]:

	Product	Forecast_Accuracy
0	SKU1	0.515940
1	SKU2	0.040987
2	SKU3	0.737935
3	SKU4	0.127312
4	SKU5	0.741711
5	SKU6	0.894752

3. Ensemble Model

In [203]:

```
from sklearn.ensemble import RandomForestClassifier

ensemble_model = Pipeline([('RF', RandomForestClassifier(n_estimator
                                                          max_depth=20,
                                                          random_state=10,
                                                          verbose=0))])
ensemble_model.fit(X,y)
```

Out[203]:

```
Pipeline(steps=[('RF',
                 RandomForestClassifier(max_depth=20, n_estimators=50,
                                        random_state=10))])
```

In [207]:

```
feat_vals = ensemble_model.named_steps['RF'].feature_importances_
df_importances = pd.DataFrame(data = list(zip(feat_lst,feat_vals)),
df_importances
```

Out[207]:

	feat	importance
0	Price Discount (%)	0.659229
1	In-Store Promo	0.040179
2	Catalogue Promo	0.024491
3	Store End Promo	0.047342
4	Google_Mobility	0.134086
5	Covid_Flag	0.028466
6	V_DAY	0.023624
7	EASTER	0.022418
8	CHRISTMAS	0.020166

In [208]:

```
accs = []
for i,j in zip(test_Xs, test_ys):
    prediction = ensemble_model.predict(i)
    accs.append(forecast_accuracy(prediction, j))

d = list(zip(['SKU1','SKU2','SKU3','SKU4','SKU5','SKU6'],accs))
df_acc = pd.DataFrame(data = d, columns=['Product', 'Forecast_Accura
df_acc
```

Out[208]:

	Product	Forecast_Accuracy
0	SKU1	0.488851
1	SKU2	0.101093
2	SKU3	0.998846
3	SKU4	0.896565
4	SKU5	0.833991
5	SKU6	0.371307

4. Boosting Model

In [212]:

```
from sklearn.ensemble import GradientBoostingRegressor

boost_model = Pipeline([('boost', GradientBoostingRegressor(random_s
boost_model.fit(X,y)
```

Out[212]:

```
Pipeline(steps=[('boost', GradientBoostingRegressor(random_state=10))])
```

In [214]:

```python
feat_vals = boost_model.named_steps['boost'].feature_importances_
df_importances = pd.DataFrame(data = list(zip(feat_lst,feat_vals)),
df_importances
```

Out[214]:

	feat	importance
0	Price Discount (%)	0.797550
1	In-Store Promo	0.006846
2	Catalogue Promo	0.015122
3	Store End Promo	0.040215
4	Google_Mobility	0.005193
5	Covid_Flag	0.127829
6	V_DAY	0.006970
7	EASTER	0.000274
8	CHRISTMAS	0.000000

In [215]:

```python
accs = []
for i,j in zip(test_Xs, test_ys):
    prediction = boost_model.predict(i)
    accs.append(forecast_accuracy(prediction, j))

d = list(zip(['SKU1','SKU2','SKU3','SKU4','SKU5','SKU6'],accs))
df_acc = pd.DataFrame(data = d, columns=['Product', 'Forecast_Accura
df_acc
```

Out[215]:

	Product	Forecast_Accuracy
0	SKU1	0.970139
1	SKU2	0.177440
2	SKU3	0.825018
3	SKU4	0.375845
4	SKU5	0.795043
5	SKU6	0.971005

In [ ]:

```python

```