Load Required Library

```
globals().clear
import numpy as np
import pandas as pd
import holidays
from sklearn import preprocessing, metrics
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import seaborn as sns
sns.set()
import matplotlib.pyplot as plt
import openpyxl
%matplotlib inline
from sklearn.svm import SVR
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import mean_squared_error
import time
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
```

Load Dataset from Excel

Get statistics from unscale data

```
stock = pd.read_excel('merged_onehot_test.xlsx')
```

load scaled dataset

```
stock.info()
print('\n#################################\n## Null values verification ##\n####################
stock.head(5)
```

Saved successfully!

```
 55  day_31            13564 non-null  int64
 56  Hour_9            13564 non-null  int64
 57  Hour_10           13564 non-null  int64
 58  Hour_11           13564 non-null  int64
 59  Hour_12           13564 non-null  int64
 60  Hour_13           13564 non-null  int64
 61  Hour_14           13564 non-null  int64
 62  Hour_15           13564 non-null  int64
 63  Hour_16           13564 non-null  int64
 64  Timeslot          13564 non-null  int64
 65  week_label        13564 non-null  int64
dtypes: datetime64[ns](1), float64(9), int64(56)
memory usage: 6.8 MB

##############################
## Null values verification ##
##############################

 Unnamed: 0         0
Date                0
TSLA_close          0
TSLA_vol_4_ave      0
TSLA_vwap_4_ave     0
                   ..
Hour_14             0
Hour_15             0
Hour_16             0
Timeslot            0
week_label          0
Length: 66, dtype: int64
```

	Unnamed: 0	Date	TSLA_close	TSLA_vol_4_ave	TSLA_vwap_4_ave	TSLA_trans_4_ave
0	5	2020-06-01 10:30:00	176.600	6531560.00	174.371825	29927.25
1	6	2020-06-01 10:45:00	176.748	4872685.00	175.236475	22062.00
2	7	2020-06-01 11:00:00	176.560	3717613.75	175.730850	17452.00
3	8	2020-06-01 11:15:00	175.474	2821491.25	175.958200	13504.00
		2020-				
				2607445.00	176.151450	12176.75

Saved successfully!

5 rows × 66 columns

Saved successfully!

Load normalized dataset

```
t1=stock
t1.index=t1['Date']
t1.drop(columns=t1.columns[0:2],
        axis=1,
        inplace=True)
t1.head()
```

Date	TSLA_close	TSLA_vol_4_ave	TSLA_vwap_4_ave	TSLA_trans_4_ave	nasx_close_-1
2020-06-01 10:30:00	176.600	6531560.00	174.371825	29927.25	9526.87
2020-06-01 10:45:00	176.748	4872685.00	175.236475	22062.00	9535.28
2020-06-01 11:00:00	176.560	3717613.75	175.730850	17452.00	9532.38
2020-06-01 11:15:00	175.474	2821491.25	175.958200	13504.00	9521.55
2020-06-01 11:30:00	175.400	2607445.00	176.151450	12176.75	9512.11

5 rows × 64 columns

Defining Functions for Metrics Calculations

MPE

```
#Ref: https://stackoverflow.com/questions/47648133/mape-calculation-in-python
#define function to calculate the MPE
                                    _pred):
                                    e), np.array(y_pred)
                                    ) / y_true) * 100
```

Saved successfully!

MAPE

```
#Ref: https://stackoverflow.com/questions/47648133/mape-calculation-in-python
#define function to calculate the MAPE
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
```

Split the Train and Test Dataset

```
train_df = t1.loc['2020-06-01 10:30:00':'2021-12-31 16:00:00']
test_df = t1.loc['2022-01-01 09:30:00':'2022-05-27 16:00:00']


start = time.time()
predictions = list()
a=1
count_time=list()

for i in test_df['week_label'].unique():

    st = time.time()
    scale_X = MinMaxScaler()

    test_subset = test_df[test_df['week_label']==i]
    print(train_df.index[0])
    print(train_df.index[-1])
    print(test_subset.index[0])
    print(test_subset.index[-1])
    train_stand = train_df.copy()
    test_stand = test_subset.copy()


    X_train, y_train = train_stand.iloc[:,1:65], train_stand.iloc[:,0]
    X_train = scale_X.fit_transform(X_train)
    svr = SVR(kernel = 'rbf', C = 10, epsilon = 0.1, gamma = 0.1, shrinking = True)
    svr.fit(X_train,y_train)
    X_test, y_test = test_stand.iloc[:,1:65], test_stand.iloc[:,0]
    X_test = scale_X.transform(X_test)
    y_hat=svr.predict(X_test)
    predictions.append(y_hat)
    et = time.time()
```

Saved successfully!

```
    train_df = train_df.append(test_df[test_df['week_label']==i])
    train_df=train_df.drop(train_df[train_df['week_label']==a].index)
```

```
        a+=1

        print(train_df.index[0])
        print(train_df.index[-1])
        print('Time taken:'+str(used_time))
        print('--------------------------')

    end = time.time()
    print("total used time"+str(end-start))
```

```
     Time taken:13.57542109489441
     --------------------------
     2020-09-08 09:30:00
     2022-04-08 16:00:00
     2022-04-11 09:30:00
     2022-04-14 16:00:00
     2020-09-14 09:30:00
     2022-04-14 16:00:00
     Time taken:13.45042896270752
     --------------------------
     2020-09-14 09:30:00
     2022-04-14 16:00:00
     2022-04-18 09:30:00
     2022-04-22 16:00:00
     2020-09-21 09:30:00
     2022-04-22 16:00:00
     Time taken:13.665416479110718
     --------------------------
     2020-09-21 09:30:00
     2022-04-22 16:00:00
     2022-04-25 09:30:00
     2022-04-29 16:00:00
     2020-09-28 09:30:00
     2022-04-29 16:00:00
     Time taken:13.26797890663147
     --------------------------
     2020-09-28 09:30:00
     2022-04-29 16:00:00
     2022-05-02 09:30:00
     2022-05-06 16:00:00
     2020-10-05 09:30:00
     2022-05-06 16:00:00
     Time taken:14.532151699066162
     --------------------------
     2020-10-05 09:30:00
     2022-05-06 16:00:00
     2022-05-09 09:30:00
     2022-05-13 16:00:00
     2020-10-12 09:30:00
     2022-05-13 16:00:00
```

Saved successfully!

```
     2022-05-13 16:00:00
     2022-05-16 09:30:00
     2022-05-20 16:00:00
```

```
2020-10-19 09:30:00
2022-05-20 16:00:00
Time taken:13.595523595809937
---------------------------
2020-10-19 09:30:00
2022-05-20 16:00:00
2022-05-23 09:30:00
2022-05-27 16:00:00
2020-10-26 09:30:00
2022-05-27 16:00:00
Time taken:13.390478610992432
---------------------------
```

```
print("total used time"+str(end-start))
```

```
total used time288.5623903274536
```

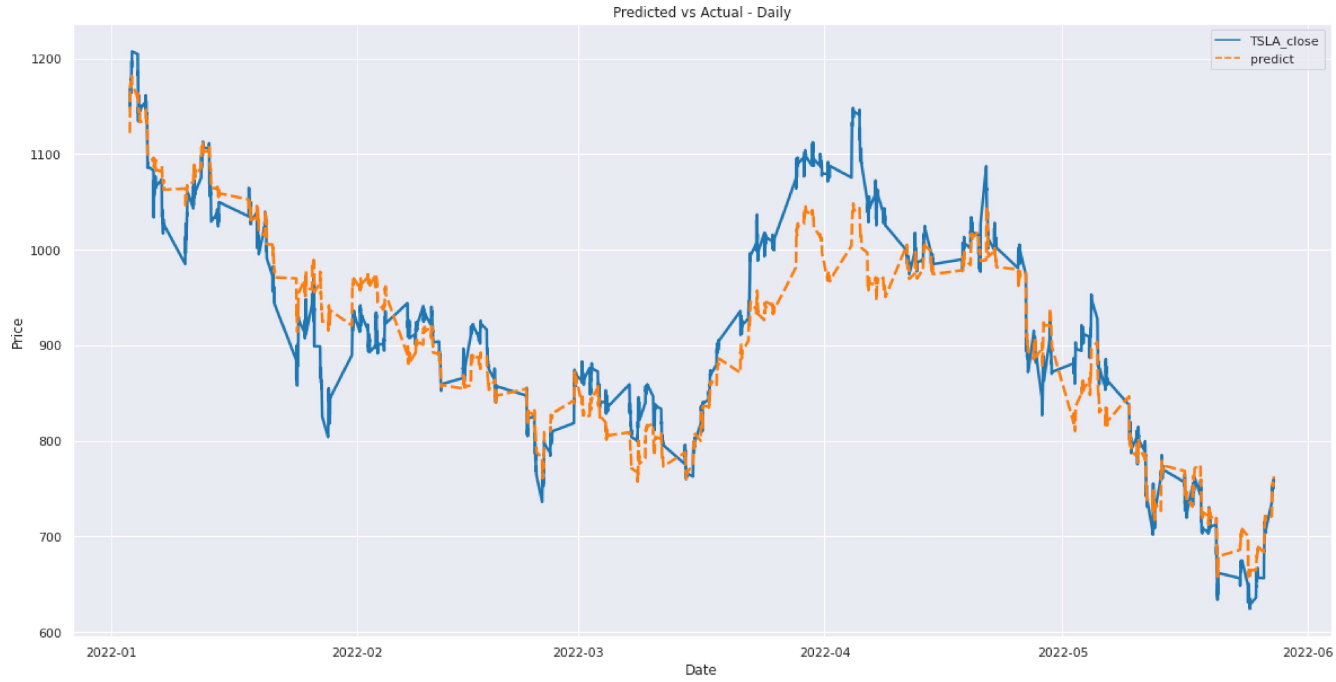
Model Evaluation

```
#Storing actual and predicted values in dataframe
df_expe = pd.DataFrame(test_df.iloc[:,0])
pred_list= list()
for i in range(len(predictions)):
  pred_list=pred_list+predictions[i].tolist()


df_pred = pd.DataFrame(pred_list,index=test_df.index,columns= ['predict'])

df_Result = pd.concat([df_expe,df_pred],axis=1)

df_Result
```

Saved successfully!

Date	TSLA_close	predict
2022-01-03 09:30:00	1165.198	1122.093769
2022-01-03 09:45:00	1149.000	1140.518604

```
plt.figure(figsize=(20,10))
linep = sns.lineplot(data=df_Result, palette="tab10", linewidth=2.5)
linep.set(xlabel='Date', ylabel='Price', title='Predicted vs Actual - Daily')
plt.show()
```



Saved successfully!

```
esult['TSLA_close'],df_Result['predict'])
```