```
import numpy as np
import pandas as pd
from datetime import datetime
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, mean_absolute_error


# read and merge datasets
data_ohlc = pd.read_csv('sp500.csv')
# data_ohlc['Date'] = pd.to_datetime(data_ohlc['Date'])
# data_ohlc['Date'] = data_ohlc['Date'].dt.strftime("%Y-%m-%d")

data_index = pd.read_csv('sp500_index.csv')

merged = pd.merge(data_ohlc, data_index, on='Date', how='inner')

merged[['Open', 'High', 'Low', 'Close', 'Volume', 'Adj Close']] = merged[['Open', 'High', 'Low', 'Close*', 'Volume', 'Adj Close**']].replace(
# merged[['Close', 'Volume']] = merged[['Close*', 'Volume']].astype(float)
# merged = merged.drop(['Close*', 'S&P500', 'Adj Close**'], axis='columns')
merged['Date'] = [int(datetime.strptime(date, '%Y-%m-%d').timestamp()) for date in merged['Date']]

# merged.dropna(inplace=True)
merged.to_csv('form1.csv',index=False)
# pd.read_csv('form1.csv').info()
set = pd.read_csv('form1.csv')
set.head(10)
```

	Date	Open	High	Low	Close*	Adj Close**	Volu
0	1672704000	3853.29	3878.46	3794.33	3,824.14	3,824.14	3.959140e+
1	1672358400	3829.06	3839.85	3800.34	3,839.50	3,839.50	2.979870e+
2	1672272000	3805.45	3858.19	3805.45	3,849.28	3,849.28	3.003680e+
3	1672185600	3829.56	3848.32	3780.78	3,783.22	3,783.22	3.083520e+
4	1672099200	3843.34	3846.65	3813.22	3,829.25	3,829.25	3.030300e+
5	1671753600	3815.11	3845.80	3797.01	3,844.82	3,844.82	2.819280e+
6	1671667200	3853.26	3853.26	3764.49	3,822.39	3,822.39	3.956950e+

```
set.corr()['Close']
```

```
<ipython-input-6-9f83690982b6>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version,
  set.corr()['Close']
Date         0.865539
Open         0.998632
High         0.999240
Low          0.999337
Volume       0.157841
Return       0.027479
S&P500       1.000000
Close        1.000000
Adj Close    1.000000
Name: Close, dtype: float64
```

```
X = set[['Date', 'Open', 'High', 'Low', 'Volume', 'Close', 'Adj Close']]
# X = set[['Volume', 'S&P500', 'Close', 'Day', 'Month', 'Year']]
y = set['Return']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# svm_model = SVR(kernel='poly', degree=2, C=100)# best parameter
svm_model = SVR(kernel='linear')
```

```
# svm_model = SVR(kernel='rbf', C=1.0, epsilon=0.1)
svm_model.fit(X_train_scaled, y_train)
```

```
SVR
SVR(kernel='linear')
```

```
y_pred = svm_model.predict(X_test_scaled)
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"mean Absolute Error (MAE): {mae:.2f}")
```

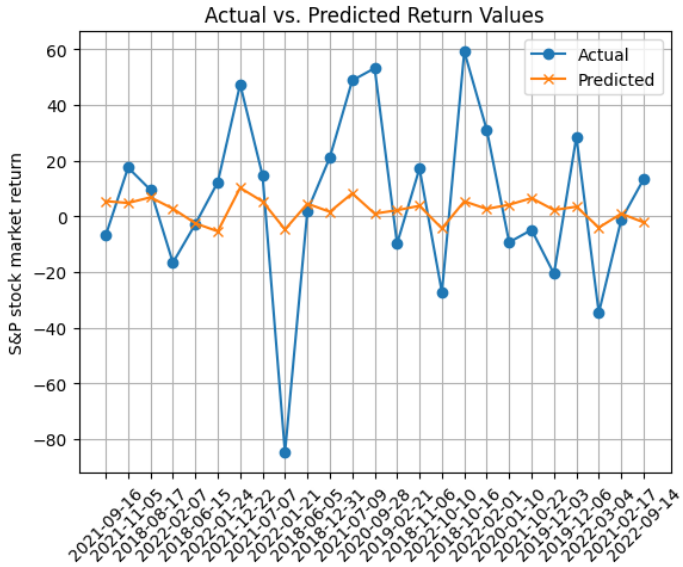
```
Mean Squared Error (MSE): 1790.62
mean Absolute Error (MAE): 28.78
```

```
X_test['Date'] = [datetime.fromtimestamp(timestamp).strftime('%Y-%m-%d') for timestamp in X_test['Date']]
```

```
import matplotlib.pyplot as plt
quick_test = pd.DataFrame({'Actual': y_test, 'Predicted' : y_pred})
quick_test.head(20)
```

	Actual	Predicted
326	-6.95	5.397390
290	17.47	4.769143
1101	9.44	6.794012
227	-16.66	2.703139
1145	-2.83	-2.534731
237	12.19	-5.416264
258	47.33	10.202859
376	14.59	5.290810
238	-84.79	-4.735621
1153	1.93	4.509782
1009	21.11	1.577971
374	48.73	8.213565
570	53.14	0.934381
974	-9.82	2.155136
1045	17.14	3.749649
58	-27.27	-4.379720
1060	59.13	5.350856

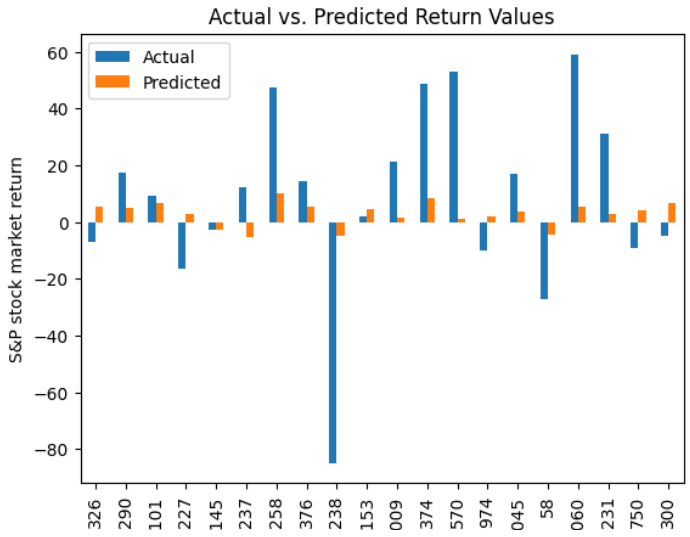
```
merged_df = quick_test.join(X_test)
merged_df = merged_df.head(25)
plt.plot(merged_df['Date'], merged_df['Actual'], label='Actual', marker='o')
plt.plot(merged_df['Date'], merged_df['Predicted'], label='Predicted', marker='x')
plt.xlabel('Date')
plt.ylabel('S&P stock market return')
plt.title('Actual vs. Predicted Return Values')
plt.legend()
plt.xticks(rotation=45)
plt.grid()
plt.show()
```




```
# X_test['Date'] = [datetime.fromtimestamp(timestamp).strftime('%Y-%m-%d') for timestamp in X_test['Date']]
# merged_df = quick_test.join(X_test)
# merged_df.set_index('Date', inplace=True)
# graph = merged_df.head(25)
# graph.plot(kind='bar')
# plt.xlabel('Date')
# plt.ylabel('S&P stock market return')
# plt.title('Actual vs. Predicted Return Values')
# plt.legend()
# plt.xticks(rotation=45)
# plt.grid()
# plt.show()
```

```
graph = quick_test.head(20)
graph.plot(kind='bar')
plt.xlabel('Date')
plt.ylabel('S&P stock market return')
plt.title('Actual vs. Predicted Return Values')
```

```
Text(0.5, 1.0, 'Actual vs. Predicted Return Values')
```



****WITH OTHER DATA****

```
# read and merge datasets
data_ohlc = pd.read_csv('sp500.csv')
# data_ohlc['Date'] = pd.to_datetime(data_ohlc['Date'])
```

```
# data_ohlc['Date'] = data_ohlc['Date'].dt.strftime("%Y-%m-%d")

data_index = pd.read_csv('sp500_index.csv')
merged = pd.merge(data_ohlc, data_index, on='Date', how='inner')

merged[['Open','High', 'Low', 'Adj Close', 'Close', 'Volume']] = merged[['Open','High', 'Low', 'Adj Close**', 'Close*', 'Volume']].replace({'
# merged[['Open','High', 'Low', 'Adj Close**', 'Close', 'Volume']] = merged[['Close*', 'Volume']].astype(float)
merged = merged.drop(['Adj Close**', 'Close*', 'S&P500'], axis='columns')

# merged.dropna(inplace=True)
print(merged)
```

```
                Date     Open     High      Low        Volume  Return  Adj Close  \
0     2023-01-03  3853.29  3878.46  3794.33  3.959140e+09  -15.36    3824.14
1     2022-12-30  3829.06  3839.85  3800.34  2.979870e+09   -9.78    3839.50
2     2022-12-29  3805.45  3858.19  3805.45  3.003680e+09   66.06    3849.28
3     2022-12-28  3829.56  3848.32  3780.78  3.083520e+09  -46.03    3783.22
4     2022-12-27  3843.34  3846.65  3813.22  3.030300e+09  -15.57    3829.25
...          ...      ...      ...      ...           ...     ...        ...
1255  2018-01-08  2742.67  2748.51  2737.60  3.246160e+09    4.56    2747.71
1256  2018-01-05  2731.33  2743.45  2727.92  3.239280e+09   19.16    2743.15
1257  2018-01-04  2719.31  2729.29  2719.07  3.697340e+09   10.93    2723.99
1258  2018-01-03  2697.85  2714.37  2697.77  3.544030e+09   17.25    2713.06
1259  2018-01-02  2683.73  2695.89  2682.36  3.397430e+09   22.20    2695.81

        Close
0     3824.14
1     3839.50
2     3849.28
3     3783.22
4     3829.25
...       ...
1255  2747.71
1256  2743.15
1257  2723.99
1258  2713.06
1259  2695.81

[1260 rows x 8 columns]
```

```
data_news = pd.read_csv('sp500_news.csv')
merged_data = pd.merge(merged, data_news, on='Date', how='inner')
# merged_data['Date'] = pd.to_datetime(merged_data['Date'])
# merged_data['Day'] = merged_data['Date'].dt.day
# merged_data['Month'] = merged_data['Date'].dt.month
# merged_data['Year'] = merged_data['Date'].dt.year
merged_data['Date'] = [int(datetime.strptime(date, '%Y-%m-%d').timestamp()) for date in merged_data['Date']]
print(merged_data)

merged_data.info()
merged_data.isnull().sum()
```

```
                Date     Open     High      Low        Volume  Return  Adj Close  \
0    1591833600  3123.53  3123.53  2999.49  7.037320e+09 -188.04    3002.10
1    1591747200  3213.42  3223.27  3181.49  6.598870e+09  -17.04    3190.14
2    1591660800  3213.32  3222.71  3193.11  6.410930e+09  -25.21    3207.18
3    1591574400  3199.92  3233.13  3196.00  8.498300e+09   38.46    3232.39
4    1591315200  3163.84  3211.72  3163.84  8.639300e+09   81.58    3193.93
..          ...      ...      ...      ...           ...     ...        ...
610  1515369600  2742.67  2748.51  2737.60  3.246160e+09    4.56    2747.71
611  1515110400  2731.33  2743.45  2727.92  3.239280e+09   19.16    2743.15
612  1515024000  2719.31  2729.29  2719.07  3.697340e+09   10.93    2723.99
613  1514937600  2697.85  2714.37  2697.77  3.544030e+09   17.25    2713.06
614  1514851200  2683.73  2695.89  2682.36  3.397430e+09   22.20    2695.81

       Close  Positive  Negative   Neutral  Price_change
0    3002.10  0.067369  0.743126  0.189504            -1
1    3190.14  0.390972  0.240399  0.368628            -1
2    3207.18  0.296441  0.337487  0.366072            -1
3    3232.39  0.523481  0.124646  0.351873             1
4    3193.93  0.567891  0.070503  0.361606             1
..       ...       ...       ...       ...           ...
610  2747.71  0.281969  0.176601  0.541430             1
611  2743.15  0.134174  0.150020  0.715806             1
612  2723.99  0.201664  0.272806  0.525530             1
613  2713.06  0.203532  0.295476  0.500992             1
614  2695.81  0.255685  0.267352  0.476963             1

[615 rows x 12 columns]
<class 'pandas.core.frame.DataFrame'>
Int64Index: 615 entries, 0 to 614
```

```
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Date          615 non-null    int64
 1   Open          615 non-null    float64
 2   High          615 non-null    float64
 3   Low           615 non-null    float64
 4   Volume        615 non-null    float64
 5   Return        615 non-null    float64
 6   Adj Close     615 non-null    float64
 7   Close         615 non-null    float64
 8   Positive      615 non-null    float64
 9   Negative      615 non-null    float64
 10  Neutral       615 non-null    float64
 11  Price_change  615 non-null    int64
dtypes: float64(10), int64(2)
memory usage: 62.5 KB
Date            0
Open            0
High            0
Low             0
Volume          0
Return          0
Adj Close       0
Close           0
Positive        0
Negative        0
Neutral         0
Price_change    0
```

```
merged_data.loc[:, 'ma20'] = merged_data.Close.rolling(20).mean()
merged_data.dropna(inplace=True)

# merged_data.loc[:, "rsi"] = talib.RSI(merged_data.Close, 14)
# print(merged_data)

merged_data.to_csv('form1.csv',index=False)
# pd.read_csv('form1.csv').info()
set = pd.read_csv('form1.csv')
# set = set.drop(['Price_change'], axis='columns')
set.head(10)
```

	Date	Open	High	Low	Volume	Return	Adj Close
0	1589414400	2794.54	2852.80	2766.64	5.651130e+09	32.50	2852.5(
1	1589328000	2865.86	2874.14	2793.15	6.151650e+09	-50.12	2820.0(
2	1589241600	2939.50	2945.82	2869.59	5.119630e+09	-60.07	2870.1:
3	1589155200	2915.46	2944.25	2903.44	4.819730e+09	0.39	2930.1!
4	1588896000	2908.83	2932.16	2902.88	4.876030e+09	48.61	2929.8(
5	1588809600	2878.26	2901.92	2876.48	5.178790e+09	32.77	2881.1!
6	1588723200	2883.14	2891.11	2847.65	4.892570e+09	-20.02	2848.4:

```
# merged['Date'] = pd.to_datetime(merged['Date'])
# merged['Day'] = merged['Date'].dt.day
# merged['Month'] = merged['Date'].dt.month
# merged['Year'] = merged['Date'].dt.year
# print(merged)

# merged.to_csv('form1.csv',index=False)
# # pd.read_csv('form1.csv').info()
# set = pd.read_csv('form1.csv')
# set.head()


X = set[['Date', 'Open','High', 'Low', 'Volume', 'Adj Close', 'Positive', 'Negative', 'Price_change', 'Neutral', 'ma20', 'Close']]
# X = set[['Volume', 'S&P500', 'Close', 'Day', 'Month', 'Year']]
y = set['Return']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4)

scaler = StandardScaler()
```

```
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# svm_model = SVR(kernel='poly', degree=3, C=100)# best parameter
svm_model = SVR(kernel='linear')
# svm_model = SVR(kernel='rbf', C=1.0, epsilon=0.1)
svm_model.fit(X_train_scaled, y_train)
```

```
SVR
SVR(kernel='linear')
```

```
y_pred = svm_model.predict(X_test_scaled)
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
```

```
Mean Squared Error (MSE): 415.98
Mean Absolute Error (MAE): 13.93
```

```
quick_test = pd.DataFrame({'Actual': y_test, 'Predicted' : y_pred})
quick_test.head(20)
```

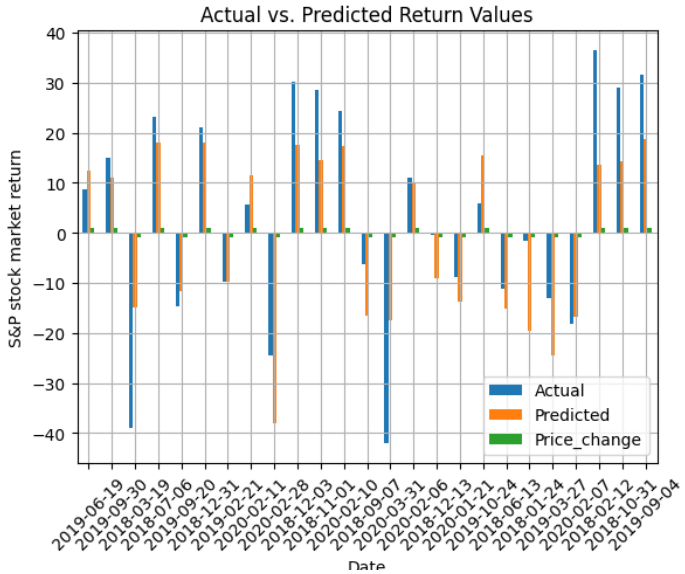
	Actual	Predicted
228	8.71	12.470199
157	14.95	10.933563
543	-39.09	-14.925421
467	23.21	18.101486
163	-14.72	-11.529882
345	21.11	17.920783
310	-9.82	-9.871308
65	5.66	11.605153
53	-24.54	-38.064555
363	30.20	17.454317
384	28.63	14.505847
66	24.38	17.287955
423	-6.37	-16.435005
31	-42.06	-17.389006
68	11.09	9.929528
356	-0.53	-9.092536
80	-8.83	-13.788133

```
merged_df = quick_test.join(X_test)
merged_df = merged_df.drop([ 'Open','High', 'Low', 'Volume', 'Adj Close', 'Positive', 'Negative', 'Neutral', 'ma20', 'Close'], axis='columns'
merged_df['Date'] = [datetime.fromtimestamp(timestamp).strftime('%Y-%m-%d') for timestamp in merged_df['Date']]
merged_df
```

	Actual	Predicted	Date	Price_change
228	8.71	12.470199	2019-06-19	1
157	14.95	10.933563	2019-09-30	1
543	-39.09	-14.925421	2018-03-19	-1
467	23.21	18.101486	2018-07-06	1
163	-14.72	-11.529882	2019-09-20	-1
...	...	...	...	...

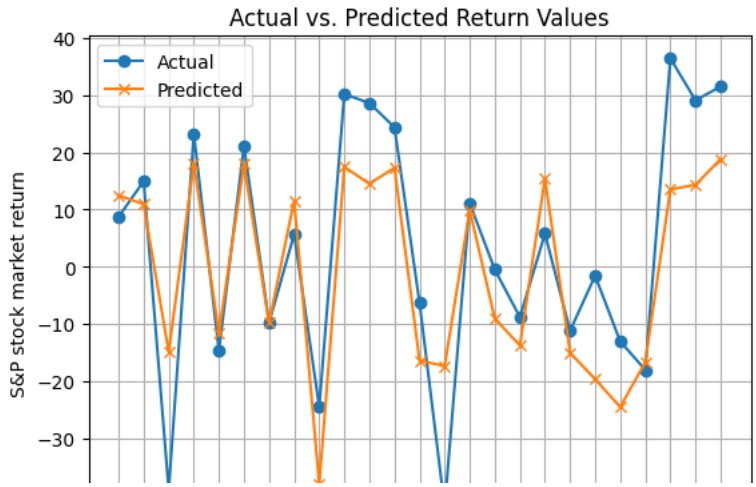
```
merged_df.set_index('Date', inplace=True)

# Create the bar chart
graph = merged_df.head(25)
graph.plot(kind='bar')
plt.xlabel('Date')
plt.ylabel('S&P stock market return')
plt.title('Actual vs. Predicted Return Values')
plt.legend()
plt.xticks(rotation=45)
plt.grid()
plt.show()
```




```
X_test['Date'] = [datetime.fromtimestamp(timestamp).strftime('%Y-%m-%d') for timestamp in X_test['Date']]
```

```
merged_df = quick_test.join(X_test)
merged_df = merged_df.head(25)
plt.plot(merged_df['Date'], merged_df['Actual'], label='Actual', marker='o')
plt.plot(merged_df['Date'], merged_df['Predicted'], label='Predicted', marker='x')
plt.xlabel('Date')
plt.ylabel('S&P stock market return')
plt.title('Actual vs. Predicted Return Values')
plt.legend()
plt.xticks(rotation=45)
plt.grid()
plt.show()
```



4/20/2020 volume-5228630000 close-2823.16 index-2823.16 positi-0.160252 negtive-0.284604 neutral-0.555145 pricechange-(-1) day-20 month-4 year-2020 ma20-2856.8213 expceted(-51.4)