pca_lstm_stateful

October 19, 2023

[1]:
```
import numpy as np
import pandas as pd
from lstm_functions import *
from lost_functions import *
from sklearn.preprocessing import MinMaxScaler, StandardScaler
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error, mean_squared_error
import yfinance as yf
from sklearn.decomposition import PCA
```

[2]:
```
xls = pd.ExcelFile('data/data_for_testing.xlsx')
all_data = {}
# This is too much data to load into memory at once
# for sheet in xls.sheet_names:
#     all_data[sheet] = pd.read_excel(xls, sheet_name=sheet)
for sheet in ["XOM", "SHW", "UPS", "DUK", "UNH", "JPM", "AMZN", "AAPL", "META",␣
 ↪"AMT"]:
    data = pd.read_excel(xls, sheet_name=sheet).set_index('Date')
    # Resample to monthly data as a simple way to reduce the number of data␣
 ↪points
    # Daily data is too much and take too long to train
    # monthly_data = data.resample('M').last().reset_index()
    original_data = data.reset_index()
    all_data[sheet] = original_data
```

[3]:
```
all_data['XOM'].head()
```

[3]:

	Date	Open	High	Low	Close	Adj Close	Volume	Sector	\
0	1962-01-02	0.0	1.589844	1.578125	1.578125	0.096660	902400	Energy	
1	1962-01-03	0.0	1.601563	1.578125	1.601563	0.098095	1200000	Energy	
2	1962-01-04	0.0	1.613281	1.597656	1.605469	0.098335	1088000	Energy	
3	1962-01-05	0.0	1.613281	1.566406	1.570313	0.096181	1222400	Energy	
4	1962-01-08	0.0	1.582031	1.546875	1.566406	0.095942	1388800	Energy	

	Ticker
0	XOM
1	XOM

```
2    XOM
3    XOM
4    XOM
```

[4]:
```
dfs = []  # List to hold individual DataFrames for each stock's transformed data

# Loop over each stock in the all_data dictionary
for stock, data in all_data.items():

    # Drop non-numeric columns, like 'Date' and 'Sector'
    numeric_data = data.drop(columns=['Date', 'Sector', "Ticker"])

    # Standardize the data
    scaler = StandardScaler()
    scaled_data = scaler.fit_transform(numeric_data)

    # Apply PCA
    pca = PCA(n_components=1)
    transformed_data = pca.fit_transform(scaled_data)

    # Create a new DataFrame for the transformed data and set its column name␣
 ↪to the stock
    df_stock = pd.DataFrame(transformed_data, columns=[stock],␣
 ↪index=data['Date'])
    dfs.append(df_stock)

# Concatenate all individual DataFrames to create the all_stock DataFrame
all_stock = pd.concat(dfs, axis=1)

all_stock = all_stock.dropna()
all_stock
```

[4]:

Date	XOM	SHW	UPS	DUK	UNH	JPM
2012-05-18	3.478038	-0.207014	-1.067590	1.640086	-0.460454	-0.045039
2012-05-21	3.205822	-0.181620	-1.059017	1.539188	-0.418354	0.033372
2012-05-22	3.229746	-0.165553	-1.045318	1.816959	-0.423939	-0.020945
2012-05-23	3.323495	-0.151068	-1.060424	1.680701	-0.408617	-0.231510
2012-05-24	3.259398	-0.116145	-1.022029	1.816028	-0.452707	-0.322557
…	…	…	…	…	…	…
2023-09-08	5.985778	6.998766	3.854588	3.919958	7.011402	5.974646
2023-09-11	5.966054	7.048274	3.837303	4.127691	7.004088	6.026697
2023-09-12	6.278726	6.946335	3.573316	4.107700	6.968067	6.108960
2023-09-13	6.064512	6.888667	3.617890	4.341208	7.008268	6.168327
2023-09-14	5.952563	6.910635	3.786476	4.075677	7.061056	6.265001

Date	AMZN	AAPL	META	AMT

```
Date
2012-05-18 -0.967054 -0.175595  6.954546 -0.631626
2012-05-21 -0.930354 -0.102414  3.996253 -0.604578
2012-05-22 -0.934165 -0.103473  3.564571 -0.640164
2012-05-23 -0.946742 -0.056809  3.348923 -0.619757
2012-05-24 -0.923368 -0.015167  3.146973 -0.595672
…                ...       ...       ...       ...
2023-09-08  4.818235  9.003848 -3.732795  2.809134
2023-09-11  4.962968  9.047471 -3.866347  2.776599
2023-09-12  4.988847  8.930482 -3.870835  2.737036
2023-09-13  5.042881  8.812317 -3.885291  2.699762
2023-09-14  5.139979  8.820390 -4.022763  2.803205

[2849 rows x 10 columns]
```

[5]:
```python
def evaluate_model(y_train, train_predictions, y_test, test_predictions,␣
↪ticker, feature):
    train_mae = mean_absolute_error(y_train, train_predictions)
    test_mae = mean_absolute_error(y_test, test_predictions)

    train_rmse = mean_squared_error(y_train, train_predictions, squared=False)
    test_rmse = mean_squared_error(y_test, test_predictions, squared=False)

    print(f"\nEvaluation for {ticker} on {feature}:")
    print(f"Training MAE: {train_mae}, Testing MAE: {test_mae}")
    print(f"Training RMSE: {train_rmse}, Testing RMSE: {test_rmse}\n")
    return train_mae, test_mae, train_rmse, test_rmse
```

[6]:
```python
def plot_predictions(y_train, train_predictions, y_test, test_predictions,␣
↪ticker, feature):
    plt.figure(figsize=(14,7))
    plt.plot(y_train, label="Actual Train Values", color='blue')
    plt.plot(train_predictions, label="Predicted Train Values", color='blue',␣
↪linestyle='dashed')
    plt.plot(np.arange(len(y_train), len(y_train) + len(y_test)), y_test,␣
↪label="Actual Test Values", color='red')
    plt.plot(np.arange(len(y_train), len(y_train) + len(y_test)),␣
↪test_predictions, label="Predicted Test Values", color='red',␣
↪linestyle='dashed')
    plt.title(f"{ticker} {feature} - Actual vs Predicted Values")
    plt.legend()
    plt.show()
```

[7]:
```python
final_importance_values = {}
final_predictions = {}
# 30 is not a good number of batches, but it's a start for testing
# 60 is a good number of batches, but it takes a long time to train
```

```
time_steps = 60
features = len(all_stock.columns)
features
batch_size_value = 4
```

[8]:
```
data = all_stock.copy().dropna()
lstm_model = LstmBuilder(time_step=time_steps, loss="mean_squared_error",␣
 ↪batch_size=batch_size_value)
model = lstm_model.create_stateful_model(features=features)
scaler = MinMaxScaler()
normalized_data = scaler.fit_transform(data)
X, y = lstm_model.create_sequences(normalized_data)
X_train, X_test, y_train, y_test = lstm_model.split_stateful_data(X,y, 0.9)
```

```
Remaining:  1
```

[9]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

[9]:
```
((2508, 60, 10), (280, 60, 10), (2508, 10), (280, 10))
```

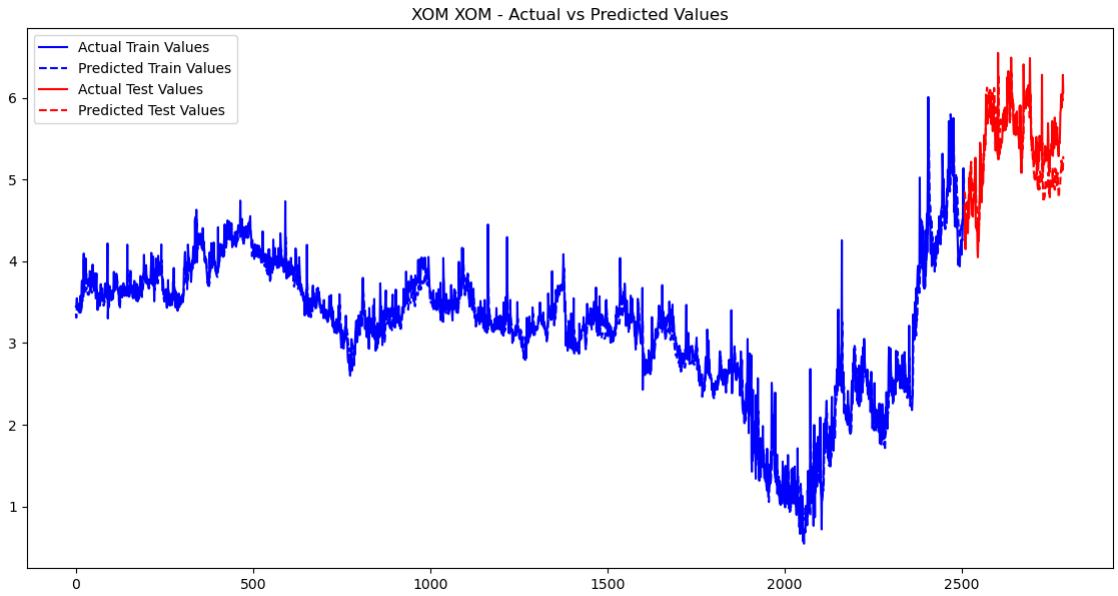
[10]:
```
model.fit(X_train, y_train, epochs=200, batch_size=batch_size_value, verbose=0)

# Extracting importance
dense_weights = model.layers[-1].get_weights()[0]

# Think about to use sum or mean and to use abs() or not
feature_weights = dense_weights.sum(axis=0)
print(feature_weights)
```

```
[1.6722538  1.5061065  1.2308213  1.6874094  1.3948823  1.233079
 0.93171537 2.149877   1.1192611  2.7269022 ]
```

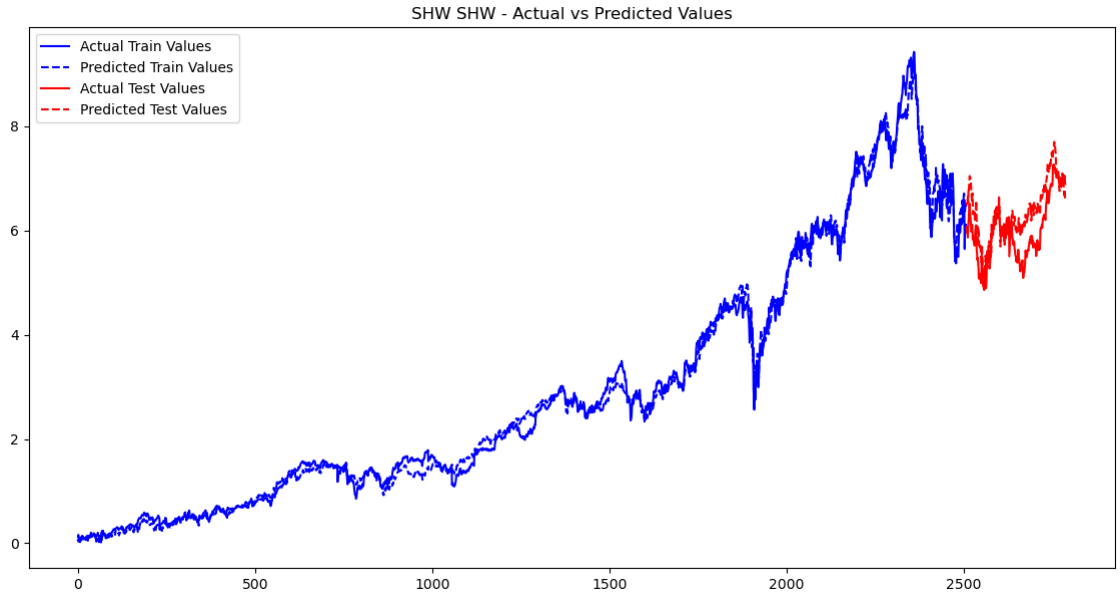
[11]:
```
# Predict for both training and testing data
train_predictions = scaler.inverse_transform(model.predict(X_train,␣
 ↪batch_size=batch_size_value))
test_predictions = scaler.inverse_transform(model.predict(X_test,␣
 ↪batch_size=batch_size_value))
y_train = scaler.inverse_transform(y_train)
y_test = scaler.inverse_transform(y_test)
features_list = ['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume']
for feature_index, feature_name in enumerate(data.columns):
    # Extracting data for the specific feature
    y_train_feature = y_train[:, feature_index]
    y_test_feature = y_test[:, feature_index]
    train_predictions_feature = train_predictions[:, feature_index]
    test_predictions_feature = test_predictions[:, feature_index]
```

```
    # Evaluating the model for this feature
    evaluate_model(y_train_feature, train_predictions_feature, y_test_feature,␣
↪test_predictions_feature, feature_name, feature_name)

    # Plotting the results for this feature
    plot_predictions(y_train_feature, train_predictions_feature,␣
↪y_test_feature, test_predictions_feature, feature_name, feature_name)
```

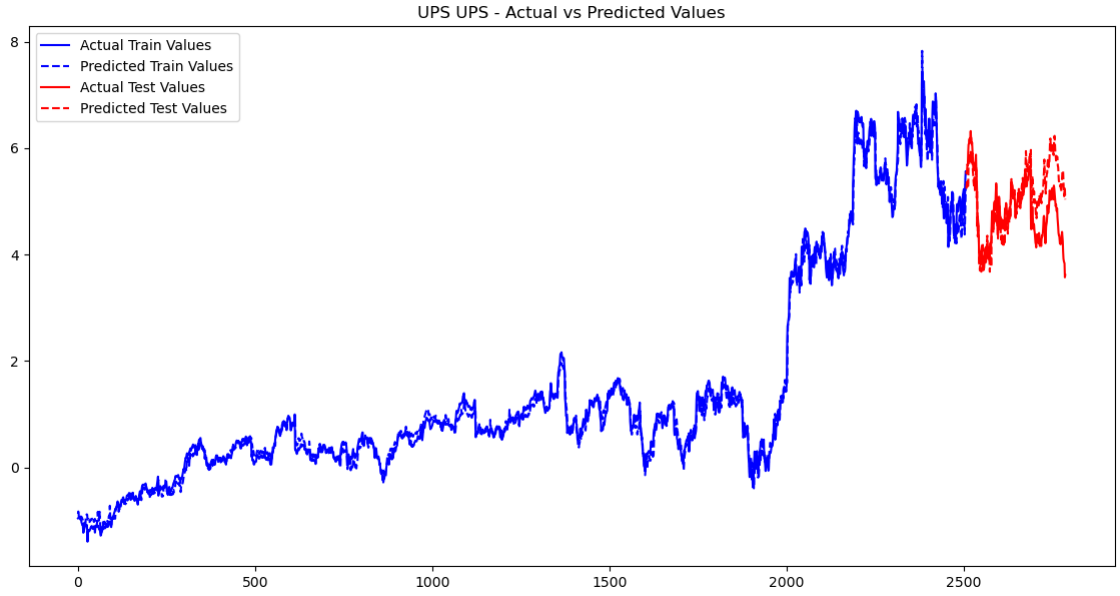
```
627/627 [==============================] - 1s 1ms/step
70/70 [==============================] - 0s 1ms/step

Evaluation for XOM on XOM:
Training MAE: 0.13255750287827858, Testing MAE: 0.27923698296888516
Training RMSE: 0.1960238900890741, Testing RMSE: 0.358164463929054
```




```
Evaluation for SHW on SHW:
Training MAE: 0.13258260727092536, Testing MAE: 0.41428915334166083
Training RMSE: 0.17892339862539705, Testing RMSE: 0.4878041220594257
```

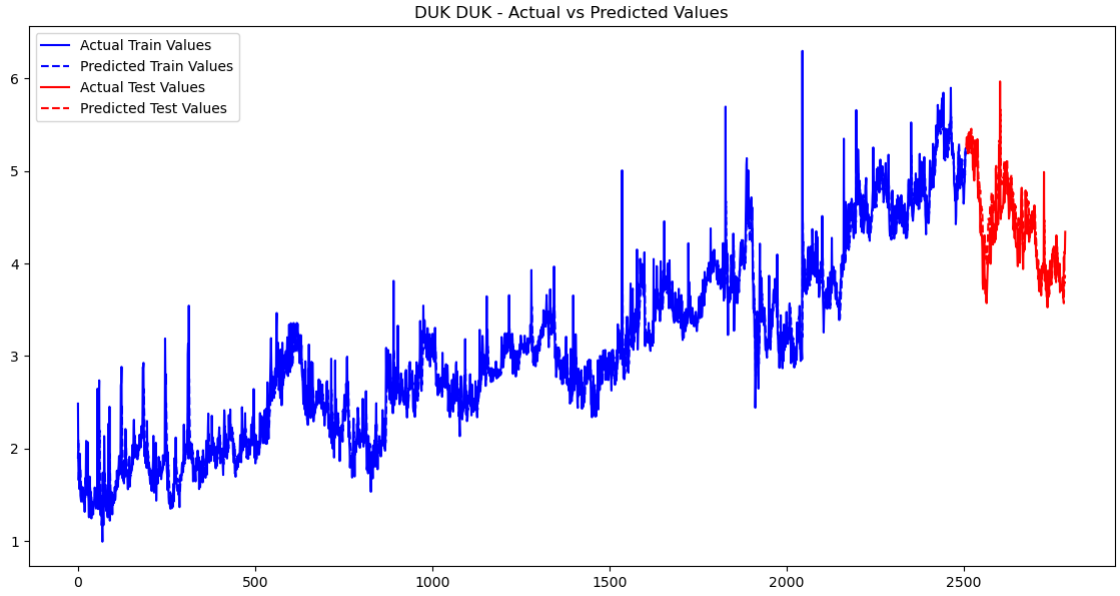


Evaluation for UPS on UPS:
Training MAE: 0.11289883081459635, Testing MAE: 0.43271060915199755
Training RMSE: 0.15144251258394173, Testing RMSE: 0.5706704862220177

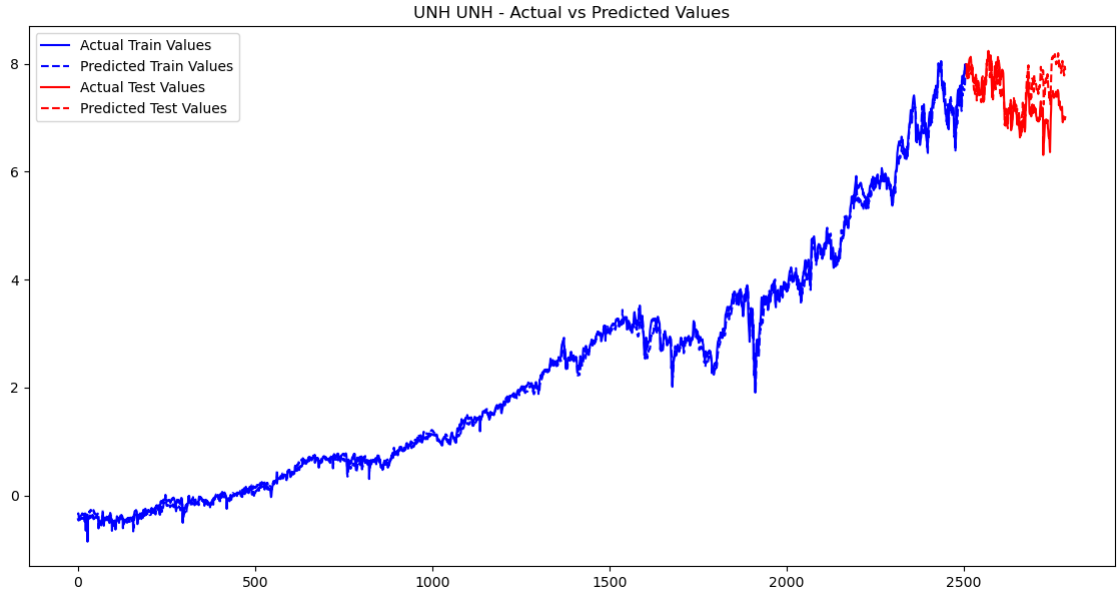


Evaluation for DUK on DUK:

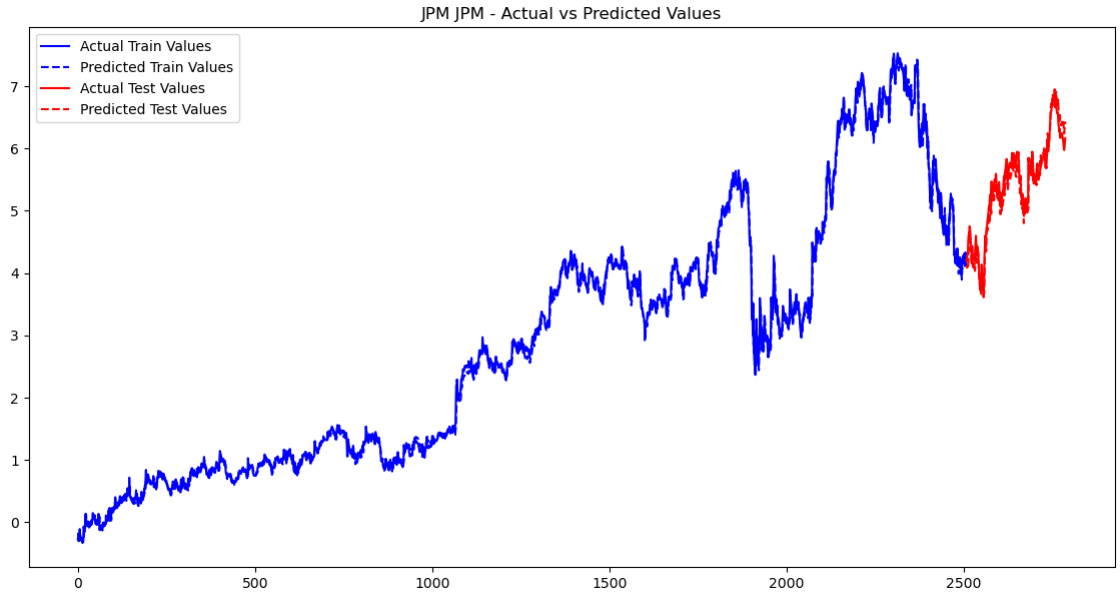
Training MAE: 0.1454755057177447, Testing MAE: 0.19396846964838918
Training RMSE: 0.21838351976513815, Testing RMSE: 0.2530138002937088



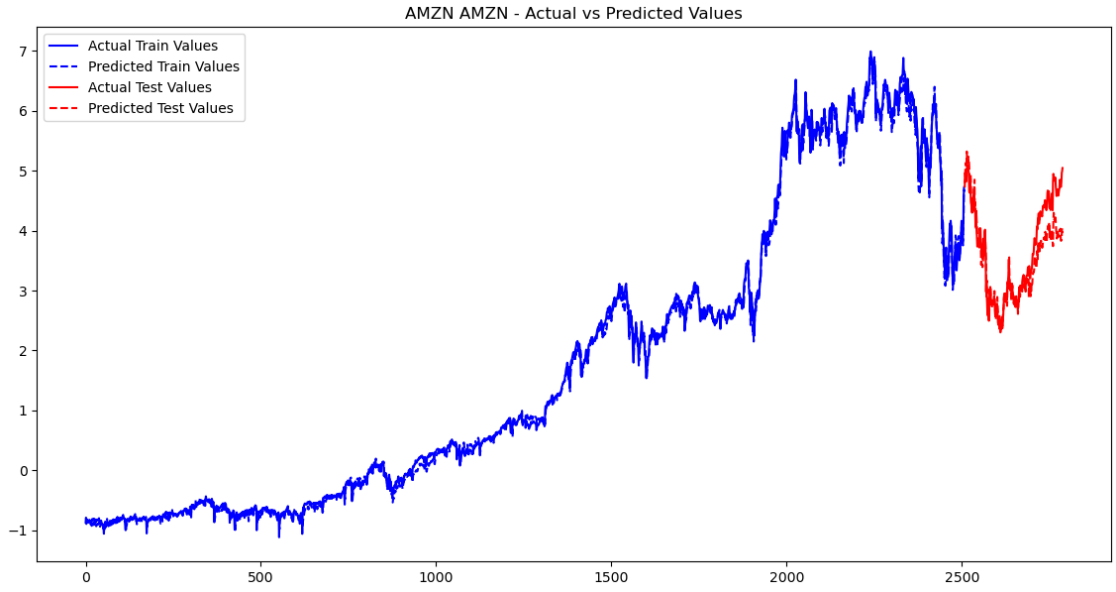
Evaluation for UNH on UNH:
Training MAE: 0.08181138140441975, Testing MAE: 0.3448757715336238
Training RMSE: 0.1109818600946433, Testing RMSE: 0.4458599778601504



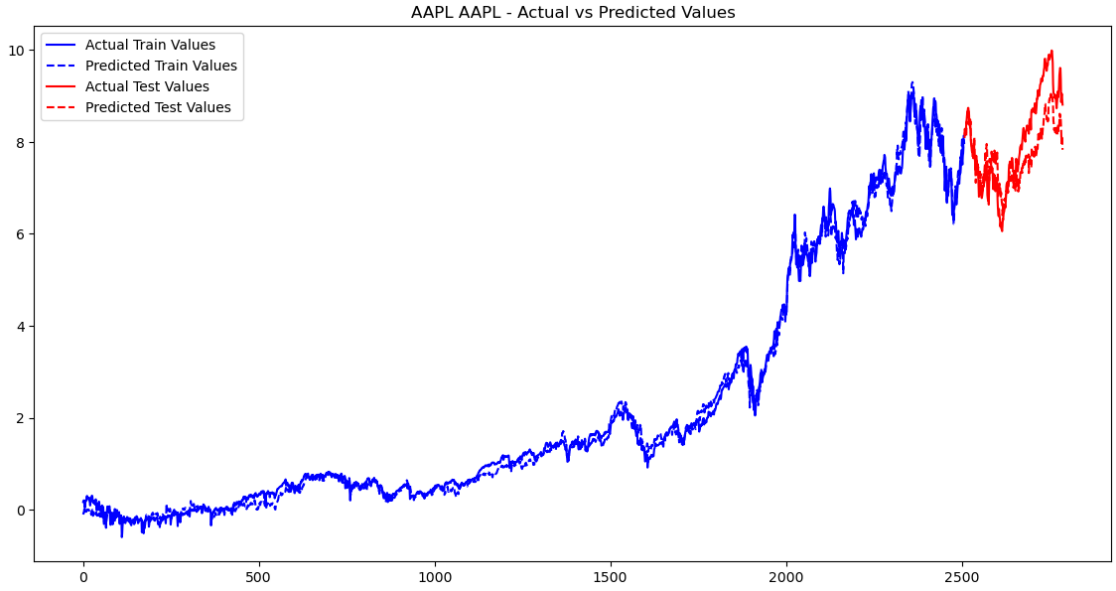
Evaluation for JPM on JPM:
Training MAE: 0.0706217669028374, Testing MAE: 0.14500820977332032
Training RMSE: 0.10134259634373674, Testing RMSE: 0.17545070222284753



Evaluation for AMZN on AMZN:
Training MAE: 0.07782010885771444, Testing MAE: 0.2623258567174893
Training RMSE: 0.10794589396598049, Testing RMSE: 0.3609639347542028

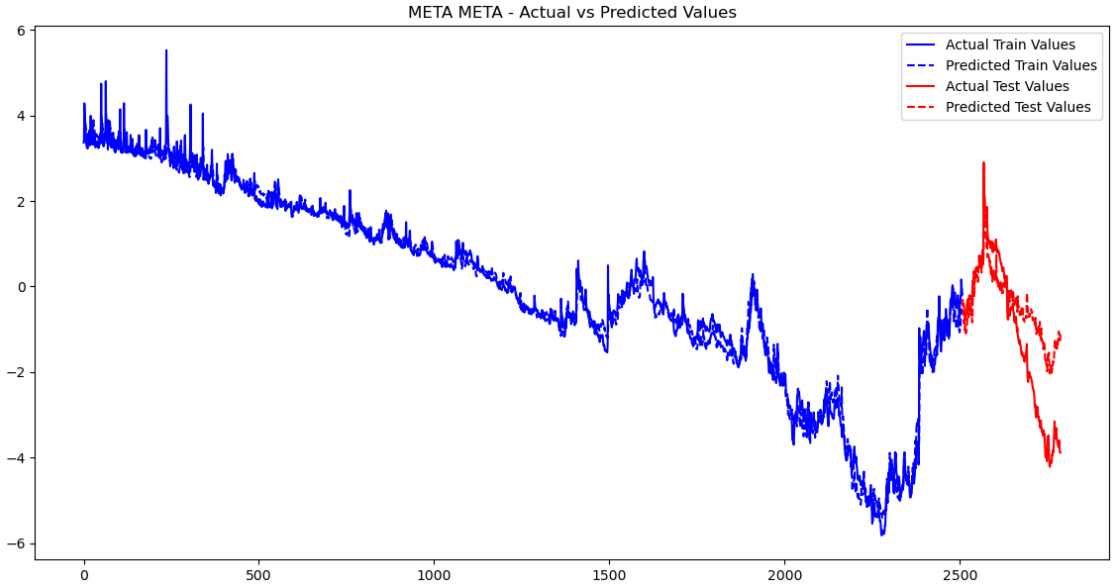


Evaluation for AAPL on AAPL:
Training MAE: 0.13441788189405504, Testing MAE: 0.5376823241552982
Training RMSE: 0.17271958993667796, Testing RMSE: 0.6447484456704441

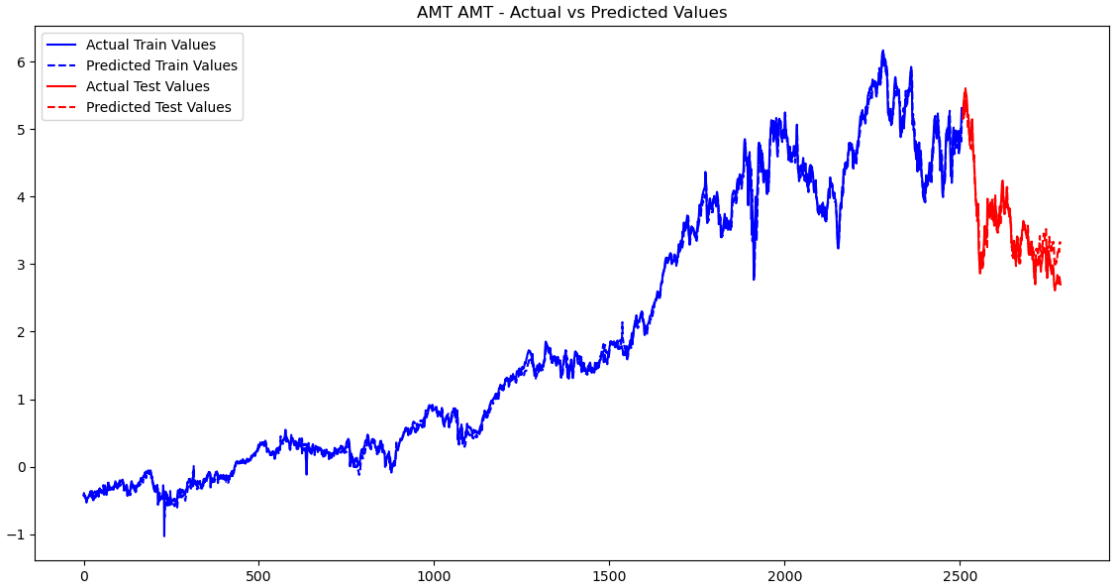


Evaluation for META on META:

Training MAE: 0.18496576996811606, Testing MAE: 0.9931223090585263
Training RMSE: 0.2514881603544181, Testing RMSE: 1.2715745174174493



Evaluation for AMT on AMT:
Training MAE: 0.06538467024065234, Testing MAE: 0.1481521772734754
Training RMSE: 0.09131565170164016, Testing RMSE: 0.19158925141715077



[12]:
```
final_importance_values = dict(zip(data.columns, feature_weights))
final_importance_values
```

[12]:
```
{'XOM': 1.6722538,
 'SHW': 1.5061065,
 'UPS': 1.2308213,
 'DUK': 1.6874094,
 'UNH': 1.3948823,
 'JPM': 1.233079,
 'AMZN': 0.93171537,
 'AAPL': 2.149877,
 'META': 1.1192611,
 'AMT': 2.7269022}
```

[13]:
```
importance_values = np.array(list(final_importance_values.values()))
importance_values
```

[13]:
```
array([1.6722538 , 1.5061065 , 1.2308213 , 1.6874094 , 1.3948823 ,
       1.233079  , 0.93171537, 2.149877  , 1.1192611 , 2.7269022 ],
      dtype=float32)
```

1 Run this if we want a arbitrage strategy

Each weight will be -1 to 1, the sum is 0

[14]:
```
# 1. Scale the values to [-1, 1]
arbitrage_scaled_importance = 2 * (importance_values - np.
 ↪min(importance_values)) / (np.max(importance_values) - np.
 ↪min(importance_values)) - 1

# 2. Ensure the sum is zero
arbitrage_normalized_importance = arbitrage_scaled_importance - np.
 ↪mean(arbitrage_scaled_importance)

# Convert back to dictionary
arbitrage_ticker_to_importance = dict(zip(final_importance_values.keys(),␣
 ↪arbitrage_normalized_importance))

print(arbitrage_ticker_to_importance)
```

```
{'XOM': 0.11923331, 'SHW': -0.06586981, 'UPS': -0.3725624, 'DUK': 0.13611794,
'UNH': -0.18978357, 'JPM': -0.37004715, 'AMZN': -0.7057933, 'AAPL': 0.65134865,
'META': -0.49685043, 'AMT': 1.2942066}
```

2 Run this instead if we want a normal strategy

Each weight will be 0 to 1, the sum is 1

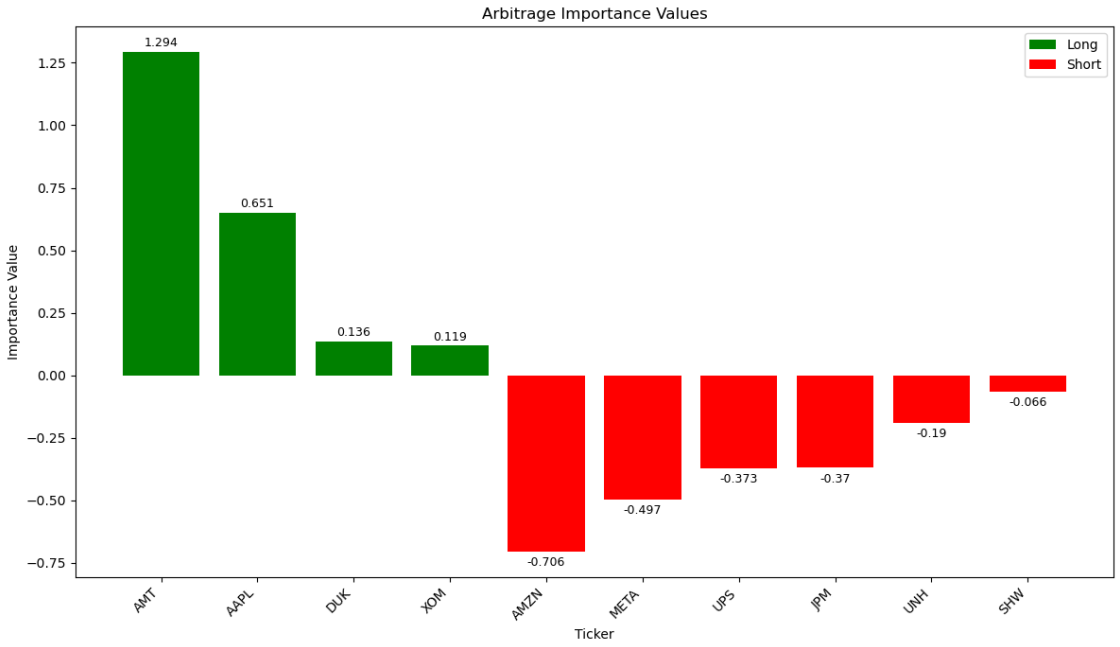
[15]:
```
def softmax(x):
    """Compute softmax values for each sets of scores in x."""
    e_x = np.exp(x - np.max(x))  # subtract max to avoid potential overflow
    return e_x / e_x.sum(axis=0)

# Convert the importance values to probabilities using softmax
probabilities = softmax(importance_values)

# Convert back to dictionary
normalized_ticker_to_importance = dict(zip(final_importance_values.keys(),␣
 ↪probabilities))

print(normalized_ticker_to_importance)
```

```
{'XOM': 0.095758535, 'SHW': 0.08109996, 'UPS': 0.061583698, 'DUK': 0.09722086,
'UNH': 0.07256322, 'JPM': 0.061722893, 'AMZN': 0.045663133, 'AAPL': 0.15438555,
'META': 0.055082776, 'AMT': 0.27491945}
```

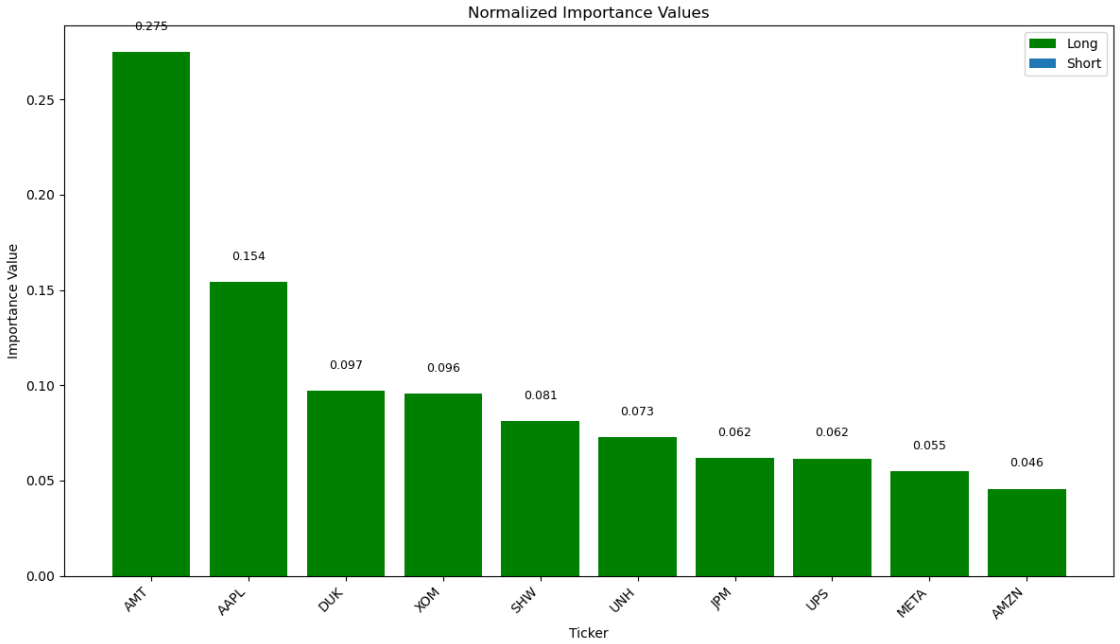
[16]:
```
def plot_importance(normalized_ticker_to_importance =␣
 ↪normalized_ticker_to_importance, title='Normalized Importance Values'):
# Split the tickers and importance values based on positive and negative values
    long_positions = {k: v for k, v in normalized_ticker_to_importance.items()␣
 ↪if v > 0}
    short_positions = {k: v for k, v in normalized_ticker_to_importance.items()␣
 ↪if v <= 0}

    # Sort the positions for better visualization
    sorted_long = dict(sorted(long_positions.items(), key=lambda item: item[1],␣
 ↪reverse=True))
    sorted_short = dict(sorted(short_positions.items(), key=lambda item:␣
 ↪item[1]))

    # Create bar charts
    fig, ax = plt.subplots(figsize=(12, 7))

    # Positive cluster
    bars_long = ax.bar(sorted_long.keys(), sorted_long.values(), color='g',␣
 ↪label='Long')

    # Negative cluster
    bars_short = ax.bar(sorted_short.keys(), sorted_short.values(), color='r',␣
 ↪label='Short')
```

```
    # Rotate x-tick labels for better readability
    plt.xticks(rotation=45, ha='right')

    # Annotate the bars
    for bar in bars_long:
        yval = bar.get_height()
        ax.text(bar.get_x() + bar.get_width()/2, yval + 0.01, round(yval, 3),␣
↪ha='center', va='bottom', fontsize=9)

    for bar in bars_short:
        yval = bar.get_height()
        ax.text(bar.get_x() + bar.get_width()/2, yval - 0.02, round(yval, 3),␣
↪ha='center', va='top', fontsize=9)

    ax.set_title(title)
    ax.set_ylabel('Importance Value')
    ax.set_xlabel('Ticker')
    ax.legend()

    plt.tight_layout()
    plt.show()
```

[17]:
```
# Plot the arbitrage importance values
plot_importance(arbitrage_ticker_to_importance, title='Arbitrage Importance␣
↪Values')
```



[18]:
```
# Plot the importance values
plot_importance(normalized_ticker_to_importance, title='Normalized Importance␣
↪Values')
```



[19]:
```
spy_data = yf.download('SPY')
spy_monthly = spy_data.resample('M').last()
spy_monthly
```

```
[**********************100%%**********************]  1 of 1 completed
```

[19]:

Date	Open	High	Low	Close	Adj Close	Volume
1993-01-31	43.968750	43.968750	43.750000	43.937500	24.941383	
1993-02-28	44.437500	44.437500	44.187500	44.406250	25.207481	
1993-03-31	45.343750	45.468750	45.187500	45.187500	25.772114	
1993-04-30	44.125000	44.281250	44.031250	44.031250	25.112648	
1993-05-31	45.406250	45.406250	45.000000	45.218750	25.789919	
…	…	…	…	…	…	
2023-06-30	441.440002	444.299988	441.109985	443.279999	441.721893	
2023-07-31	457.410004	458.160004	456.049988	457.790009	456.180908	
2023-08-31	451.649994	452.829987	450.160004	450.350006	448.767059	
2023-09-30	431.670013	431.850006	425.910004	427.480011	427.480011	
2023-10-31	434.190002	435.179993	429.089996	430.209991	430.209991	

```
1993-01-31    1003200
1993-02-28      66200
1993-03-31     111600
1993-04-30      88500
1993-05-31      79100
…                 …
2023-06-30  104921500
2023-07-31   62040400
2023-08-31   66084600
2023-09-30  115078500
2023-10-31   93409900

[370 rows x 6 columns]
```

[20]:
```
all_data
```

[20]:
```
{'XOM':             Date        Open        High         Low       Close   Adj
Close  \
 0     1962-01-02    0.000000    1.589844    1.578125    1.578125    0.096660
 1     1962-01-03    0.000000    1.601563    1.578125    1.601563    0.098095
 2     1962-01-04    0.000000    1.613281    1.597656    1.605469    0.098335
 3     1962-01-05    0.000000    1.613281    1.566406    1.570313    0.096181
 4     1962-01-08    0.000000    1.582031    1.546875    1.566406    0.095942
 …            …           …           …           …           …           …
 15527 2023-09-08  114.529999  116.050003  114.320000  115.610001  115.610001
 15528 2023-09-11  116.180000  116.680000  113.570000  114.160004  114.160004
 15529 2023-09-12  115.339996  117.669998  115.269997  117.489998  117.489998
 15530 2023-09-13  117.410004  117.959999  115.900002  116.440002  116.440002
 15531 2023-09-14  117.550003  118.660004  117.320000  118.345001  118.345001

          Volume  Sector Ticker
 0        902400  Energy    XOM
 1       1200000  Energy    XOM
 2       1088000  Energy    XOM
 3       1222400  Energy    XOM
 4       1388800  Energy    XOM
 …           …      …     …
 15527  14283200  Energy    XOM
 15528  14383600  Energy    XOM
 15529  20145800  Energy    XOM
 15530  13471600  Energy    XOM
 15531   7550806  Energy    XOM

 [15532 rows x 9 columns],
 'SHW':             Date        Open        High         Low       Close   Adj
Close  \
 0     1980-03-17    0.000000    0.299479    0.283854    0.283854    0.150063
```

```
1      1980-03-18    0.000000    0.276042    0.272135    0.273438    0.144557
2      1980-03-19    0.000000    0.281250    0.276042    0.281250    0.148687
3      1980-03-20    0.000000    0.278646    0.276042    0.276042    0.145933
4      1980-03-21    0.000000    0.276042    0.272135    0.276042    0.145933
…           …           …           …           …           …           …
10962 2023-09-08  270.549988  273.820007  270.410004  271.450012  271.450012
10963 2023-09-11  271.980011  274.640015  271.760010  273.649994  273.649994
10964 2023-09-12  272.059998  272.239990  267.609985  268.730011  268.730011
10965 2023-09-13  268.040009  269.230011  265.459991  268.459991  268.459991
10966 2023-09-14  269.459991  269.839996  266.519989  268.519989  268.519989

         Volume     Sector Ticker
0       1401600  Materials    SHW
1       1459200  Materials    SHW
2       1296000  Materials    SHW
3        422400  Materials    SHW
4        220800  Materials    SHW
…          …          …    …
10962   1088400  Materials    SHW
10963   1035800  Materials    SHW
10964   1358100  Materials    SHW
10965    910000  Materials    SHW
10966    376910  Materials    SHW

[10967 rows x 9 columns],
'UPS':            Date        Open        High         Low       Close   Adj
Close  \
0     1999-11-10   65.000000   70.312500   64.500000   68.250000   37.137554
1     1999-11-11   68.750000   76.937500   68.750000   75.000000   40.810482
2     1999-11-12   76.000000   76.625000   69.250000   70.500000   38.361855
3     1999-11-15   70.500000   70.875000   67.312500   69.000000   37.545658
4     1999-11-16   67.875000   68.000000   65.000000   66.000000   35.913219
…          …          …           …           …           …           …
5994 2023-09-08  162.419998  162.419998  160.600006  161.039993  161.039993
5995 2023-09-11  161.600006  162.279999  160.339996  160.889999  160.889999
5996 2023-09-12  157.369995  158.000000  155.100006  156.570007  156.570007
5997 2023-09-13  156.559998  158.050003  155.860001  157.839996  157.839996
5998 2023-09-14  159.759995  160.539993  159.029999  160.110001  160.110001

         Volume       Sector Ticker
0      80793700  Industrials    UPS
1      28309000  Industrials    UPS
2      14768100  Industrials    UPS
3       8675100  Industrials    UPS
4       6978600  Industrials    UPS
…          …           …    …
5994    3561500  Industrials    UPS
```

```
 5995   3638000  Industrials    UPS
 5996   6232500  Industrials    UPS
 5997   4071300  Industrials    UPS
 5998   1301783  Industrials    UPS

 [5999 rows x 9 columns],
 'DUK':             Date       Open       High        Low      Close  Adj Close
\
 0     1980-03-17   0.000000   6.646331   6.428419   6.428419   0.821596
 1     1980-03-18   0.000000   6.918722   6.428419   6.918722   0.884260
 2     1980-03-19   0.000000   6.973200   6.700809   6.700809   0.856409
 3     1980-03-20   0.000000   6.755288   6.646331   6.755288   0.863372
 4     1980-03-21   0.000000   6.755288   6.646331   6.700809   0.856409
 …          …         …         …         …         …         …
 10962 2023-09-08  89.570000  91.169998  89.279999  91.019997  91.019997
 10963 2023-09-11  90.849998  92.470001  90.730003  91.779999  91.779999
 10964 2023-09-12  91.910004  92.669998  90.900002  92.120003  92.120003
 10965 2023-09-13  92.430000  94.360001  92.320000  93.879997  93.879997
 10966 2023-09-14  94.650002  95.589996  94.360001  95.432999  95.432999

         Volume     Sector Ticker
 0       133540  Utilities    DUK
 1       224861  Utilities    DUK
 2       319394  Utilities    DUK
 3       118855  Utilities    DUK
 4       100270  Utilities    DUK
 …          …        …    …
 10962  3302700  Utilities    DUK
 10963  3992900  Utilities    DUK
 10964  3689300  Utilities    DUK
 10965  4382900  Utilities    DUK
 10966  2081490  Utilities    DUK

 [10967 rows x 9 columns],
 'UNH':             Date        Open        High         Low       Close   Adj
Close  \
 0    1984-10-17    0.000000    0.148438    0.144531    0.144531    0.116592
 1    1984-10-18    0.000000    0.156250    0.148438    0.148438    0.119744
 2    1984-10-19    0.000000    0.148438    0.144531    0.144531    0.116592
 3    1984-10-22    0.000000    0.148438    0.144531    0.144531    0.116592
 4    1984-10-23    0.000000    0.144531    0.140625    0.140625    0.113441
 …          …          …          …          …          …          …
 9801 2023-09-08  480.190002  482.970001  478.750000  480.769989  480.769989
 9802 2023-09-11  481.980011  483.839996  478.000000  479.380005  479.380005
 9803 2023-09-12  477.380005  483.640015  472.119995  479.899994  479.899994
 9804 2023-09-13  481.429993  484.040009  479.459991  479.839996  479.839996
 9805 2023-09-14  482.630005  484.230011  478.777496  483.459991  483.459991
```

```
        Volume      Sector Ticker
 0      9868800  Healthcare    UNH
 1      5324800  Healthcare    UNH
 2      3043200  Healthcare    UNH
 3      2326400  Healthcare    UNH
 4       787200  Healthcare    UNH
 …         …         …    …
 9801   1858400  Healthcare    UNH
 9802   2059200  Healthcare    UNH
 9803   2195200  Healthcare    UNH
 9804   2197800  Healthcare    UNH
 9805    976896  Healthcare    UNH

 [9806 rows x 9 columns],
 'JPM':             Date        Open        High         Low       Close   Adj
Close  \
 0      1980-03-17    0.000000    5.129630    5.018519    5.037037    1.098447
 1      1980-03-18    0.000000    5.111111    5.037037    5.074074    1.106524
 2      1980-03-19    0.000000    5.166667    5.111111    5.148148    1.122678
 3      1980-03-20    0.000000    5.148148    5.092593    5.111111    1.114601
 4      1980-03-21    0.000000    5.222222    5.111111    5.222222    1.138831
 …          …         …         …          …          …         …
 10962 2023-09-08  143.369995  144.119995  142.649994  143.830002  143.830002
 10963 2023-09-11  144.750000  145.050003  143.690002  144.460007  144.460007
 10964 2023-09-12  144.500000  147.320007  144.050003  146.339996  146.339996
 10965 2023-09-13  147.339996  147.699997  145.820007  146.410004  146.410004
 10966 2023-09-14  147.839996  149.889999  147.520004  149.330002  149.330002

         Volume      Sector Ticker
 0        62775  Financials    JPM
 1        64125  Financials    JPM
 2        40500  Financials    JPM
 3        18900  Financials    JPM
 4        97200  Financials    JPM
 …          …         …    …
 10962  7107900  Financials    JPM
 10963  6854200  Financials    JPM
 10964  8363200  Financials    JPM
 10965  8323000  Financials    JPM
 10966  4911704  Financials    JPM

 [10967 rows x 9 columns],
 'AMZN':             Date        Open        High         Low       Close   Adj
Close  \
 0    1997-05-15    0.121875    0.125000    0.096354    0.097917    0.097917
 1    1997-05-16    0.098438    0.098958    0.085417    0.086458    0.086458
```

```
2    1997-05-19    0.088021    0.088542    0.081250    0.085417    0.085417
3    1997-05-20    0.086458    0.087500    0.081771    0.081771    0.081771
4    1997-05-21    0.081771    0.082292    0.068750    0.071354    0.071354
…         …          …          …          …          …          …
6622 2023-09-08  136.860001  138.850006  136.750000  138.229996  138.229996
6623 2023-09-11  138.750000  143.619995  138.639999  143.100006  143.100006
6624 2023-09-12  142.320007  143.000000  140.610001  141.229996  141.229996
6625 2023-09-13  140.949997  144.979996  140.869995  144.850006  144.850006
6626 2023-09-14  145.080002  145.729996  142.960007  145.399994  145.399994

          Volume                  Sector Ticker
0     1443120000  Consumer Discretionary   AMZN
1      294000000  Consumer Discretionary   AMZN
2       122136000  Consumer Discretionary   AMZN
3       109344000  Consumer Discretionary   AMZN
4       377064000  Consumer Discretionary   AMZN
…          …                    …    …
6622    38348200  Consumer Discretionary   AMZN
6623    56764500  Consumer Discretionary   AMZN
6624    42668500  Consumer Discretionary   AMZN
6625    60338700  Consumer Discretionary   AMZN
6626    37457493  Consumer Discretionary   AMZN

[6627 rows x 9 columns],
'AAPL':             Date        Open        High         Low       Close   Adj
Close  \
0     1980-12-12    0.128348    0.128906    0.128348    0.128348    0.099450
1     1980-12-15    0.122210    0.122210    0.121652    0.121652    0.094261
2     1980-12-16    0.113281    0.113281    0.112723    0.112723    0.087343
3     1980-12-17    0.115513    0.116071    0.115513    0.115513    0.089504
4     1980-12-18    0.118862    0.119420    0.118862    0.118862    0.092099
…          …          …          …          …          …          …
10774 2023-09-08  178.350006  180.240005  177.789993  178.179993  178.179993
10775 2023-09-11  180.070007  180.300003  177.339996  179.360001  179.360001
10776 2023-09-12  179.490005  180.130005  174.820007  176.300003  176.300003
10777 2023-09-13  176.509995  177.300003  173.979996  174.210007  174.210007
10778 2023-09-14  174.000000  176.039993  173.580002  175.850006  175.850006

          Volume                  Sector Ticker
0      469033600  Information Technology   AAPL
1      175884800  Information Technology   AAPL
2      105728000  Information Technology   AAPL
3       86441600  Information Technology   AAPL
4       73449600  Information Technology   AAPL
…          …                    …    …
10774   65551300  Information Technology   AAPL
10775   58953100  Information Technology   AAPL
```

```
 10776   90370200  Information Technology   AAPL
 10777   84165800  Information Technology   AAPL
 10778   37303954  Information Technology   AAPL

 [10779 rows x 9 columns],
 'META':            Date        Open        High         Low       Close   Adj
Close  \
 0    2012-05-18   42.049999   45.000000   38.000000   38.230000   38.230000
 1    2012-05-21   36.529999   36.660000   33.000000   34.029999   34.029999
 2    2012-05-22   32.610001   33.590000   30.940001   31.000000   31.000000
 3    2012-05-23   31.370001   32.500000   31.360001   32.000000   32.000000
 4    2012-05-24   32.950001   33.209999   31.770000   33.029999   33.029999
 …         …         …          …          …          …          …
 2844 2023-09-08  299.220001  305.250000  296.779999  297.890015  297.890015
 2845 2023-09-11  301.410004  309.040009  301.279999  307.559998  307.559998
 2846 2023-09-12  306.329987  308.660004  300.230011  301.660004  301.660004
 2847 2023-09-13  302.359985  307.179993  301.320007  305.059998  305.059998
 2848 2023-09-14  306.739990  311.500000  305.029999  311.265015  311.265015

          Volume                  Sector Ticker
 0     573576400  Communication Services   META
 1     168192700  Communication Services   META
 2     101786600  Communication Services   META
 3      73600000  Communication Services   META
 4      50237200  Communication Services   META
 …         …                      …    …
 2844   17548000  Communication Services   META
 2845   19489300  Communication Services   META
 2846   13480400  Communication Services   META
 2847   13197400  Communication Services   META
 2848   11355695  Communication Services   META

 [2849 rows x 9 columns],
 'AMT':            Date        Open        High         Low       Close   Adj
Close  \
 0    1998-02-27   17.375000   18.000000   17.375000   17.375000   13.841517
 1    1998-03-02   17.250000   17.250000   17.000000   17.250000   13.741940
 2    1998-03-03   17.125000   17.125000   16.750000   17.125000   13.642358
 3    1998-03-04   15.750000   15.750000   15.500000   15.750000   12.546988
 4    1998-03-05   16.125000   16.125000   16.125000   16.125000   12.845725
 …         …         …          …          …          …          …
 6424 2023-09-08  182.889999  183.300003  178.860001  180.889999  180.889999
 6425 2023-09-11  180.759995  181.039993  177.550003  180.279999  180.279999
 6426 2023-09-12  179.350006  179.619995  175.399994  179.190002  179.190002
 6427 2023-09-13  178.479996  178.929993  176.369995  176.949997  176.949997
 6428 2023-09-14  178.630005  182.220001  178.660004  180.740005  180.740005
```

```
        Volume       Sector Ticker
0        50000  Real Estate    AMT
1        75000  Real Estate    AMT
2        35000  Real Estate    AMT
3        60000  Real Estate    AMT
4          400  Real Estate    AMT
…          …           …    …
6424   1645800  Real Estate    AMT
6425   1393100  Real Estate    AMT
6426   1341500  Real Estate    AMT
6427   1639000  Real Estate    AMT
6428    642821  Real Estate    AMT

[6429 rows x 9 columns]}
```

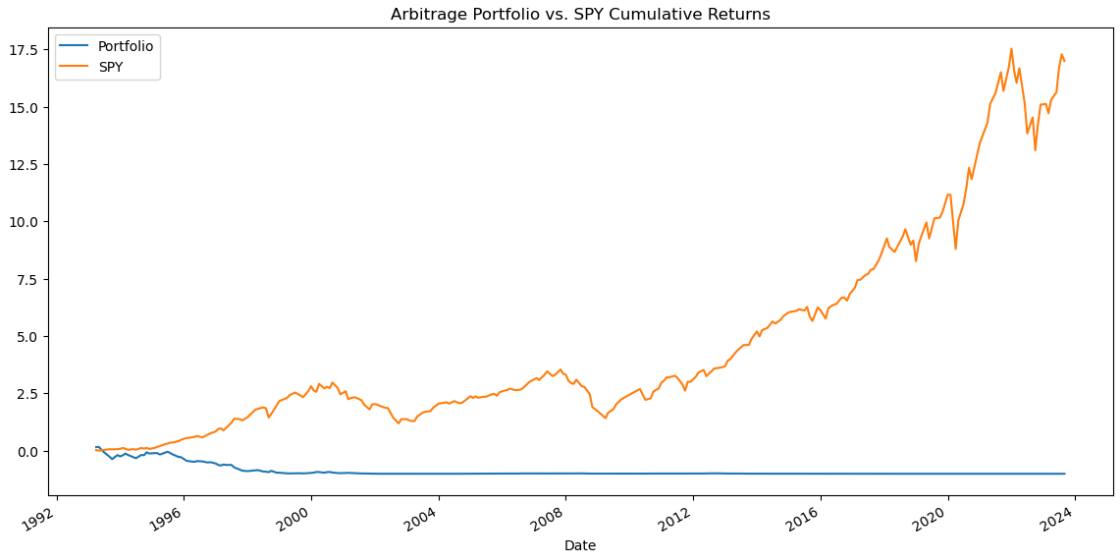
[21]:
```
# Construct the Portfolio and Backtest
def build_portfolio(normalized_ticker_to_importance=normalized_ticker_to_importance, strategy='Normal'):
    portfolio_returns = pd.DataFrame()
    for ticker, importance in normalized_ticker_to_importance.items():
        data = all_data[ticker].set_index('Date')
        data['Returns'] = data['Adj Close'].pct_change().fillna(0)
        portfolio_returns[ticker] = data['Returns'] * importance
    portfolio_returns['Portfolio'] = portfolio_returns.sum(axis=1)
    spy_monthly['SPY Returns'] = spy_monthly['Adj Close'].pct_change().fillna(0)
    # Cumulative Returns
    portfolio_returns['Cumulative Portfolio'] = (portfolio_returns['Portfolio'] + 1).cumprod() - 1
    spy_monthly['Cumulative SPY'] = (spy_monthly['SPY Returns'] + 1).cumprod() - 1
    combined = pd.concat([portfolio_returns['Cumulative Portfolio'], spy_monthly['Cumulative SPY']], axis=1).dropna()
    print(combined)
    # Plot
    plt.figure(figsize=(14,7))
    combined['Cumulative Portfolio'].plot(label="Portfolio")
    combined['Cumulative SPY'].plot(label="SPY")
    plt.legend()
    plt.title(strategy + " Portfolio vs. SPY Cumulative Returns")
    plt.show()
```

[22]:
```
# Build the portfolio for arbitrage strategy
build_portfolio(arbitrage_ticker_to_importance, strategy='Arbitrage')
```

```
            Cumulative Portfolio  Cumulative SPY
Date
```

1993-03-31	0.163276	0.033307
1993-04-30	0.171251	0.006867
1993-06-30	-0.066997	0.037752
1993-08-31	-0.255177	0.072295
1993-09-30	-0.361156	0.064494
…	…	…
2023-03-31	-0.998138	15.295858
2023-05-31	-0.998636	15.632611
2023-06-30	-0.998667	16.710401
2023-07-31	-0.998853	17.290121
2023-08-31	-0.998870	16.992870

[259 rows x 2 columns]



[23]:
```
# Build the portfolio for the Normal strategy
build_portfolio(normalized_ticker_to_importance, strategy='Normal')
```

Date	Cumulative Portfolio	Cumulative SPY
1993-03-31	5.102172	0.033307
1993-04-30	5.090873	0.006867
1993-06-30	5.011551	0.037752
1993-08-31	4.728857	0.072295
1993-09-30	4.768873	0.064494
…	…	…
2023-03-31	837.606962	15.295858
2023-05-31	832.067358	15.632611
2023-06-30	887.711092	16.710401
2023-07-31	910.083894	17.290121

```
2023-08-31          876.984497        16.992870

[259 rows x 2 columns]
```