In [1]:
```python
#import necessary libraries
```

In [1]:
```python
import pandas as pd
import numpy as np
import gensim
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

In [3]:
```python
#load the 'OHLC' data
csv_file_path = "C:/Users/narayan/Downloads/Reliance.csv"
df = pd.read_csv(csv_file_path)
```

In [7]:
```python
#display the first few row
print("OHLC Data (First 5 rows):")
print(df.head())
print("\nColumn Names:", df.columns)
```

```
OHLC Data (First 5 rows):
                        date    open    high     low   close  volume
0  2015-03-20 09:15:00+05:30  425.00  426.25  425.00  425.00    1186
1  2015-03-20 09:16:00+05:30  424.50  425.00  424.20  425.00     768
2  2015-03-20 09:17:00+05:30  425.00  425.90  425.00  425.15     425
3  2015-03-20 09:18:00+05:30  425.90  425.95  425.85  425.85     659
4  2015-03-20 09:19:00+05:30  426.25  426.40  425.15  425.15    1548

Column Names: Index(['date', 'open', 'high', 'low', 'close', 'volume'], dtype='object')
```

In [ ]:
```python
#load the vector embeddings using gensim
kv_file_path = "C:/Users/narayan/Downloads/Reliance_embeddings.kv"
kv_model = gensim.models.KeyedVectors.load(kv_file_path, mmap='r')
```

In [16]:
```python
#display embedding vectors for the first few keys
print(f"\nNumber of Keys in Embeddings Model: {len(kv_model.key_to_index)}")
print(f"Sample Keys: {list(kv_model.key_to_index.keys())[:5]}")
```

```
Number of Keys in Embeddings Model: 29979
Sample Keys: ['reliance_1', 'reliance_2', 'reliance_3', 'reliance_4', 'reliance_5']
```

In [17]:
```python
#cheking key for specific vector
print("\nVector for 'reliance_1':")
print(kv_model['reliance_1'])
```

```
Vector for 'reliance_1':
[0.00000000e+00 0.00000000e+00 4.16077115e-03 0.00000000e+00
 6.06304407e-01 0.00000000e+00 4.13331809e-03 0.00000000e+00
 0.00000000e+00 5.66237932e-03 0.00000000e+00 0.00000000e+00
 1.27049908e-02 0.00000000e+00 0.00000000e+00 1.59788623e-01
 0.00000000e+00 1.38400078e-01 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 2.45041423e-03
 2.30452009e-02 0.00000000e+00 8.23342577e-02 0.00000000e+00
 1.00663379e-01 0.00000000e+00 1.47060500e-02 7.20283191e-04
 3.97689193e-02 0.00000000e+00 2.01767147e-01 0.00000000e+00
 2.00435370e-02 2.75550671e-02 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 3.48734379e-01 0.00000000e+00
 1.39904320e-01 5.35318911e-01 9.57547594e-03 1.54781586e-03
 3.55014764e-03 5.44607520e-01 2.63779998e-01 0.00000000e+00
 1.48544076e-03 0.00000000e+00 4.72142436e-02 9.41457093e-01
 3.91822517e-01 9.51567572e-03 8.25552046e-01 6.12963259e-01
 0.00000000e+00 0.00000000e+00 2.76277168e-03 4.92841611e-03
 0.00000000e+00 0.00000000e+00 3.59666836e-03 0.00000000e+00
 0.00000000e+00 3.60150933e-02 0.00000000e+00 3.70119900e-01
 4.74092960e-01 0.00000000e+00 3.22024571e-04 0.00000000e+00
 9.39241350e-01 2.92293169e-02 0.00000000e+00 0.00000000e+00
 0.00000000e+00 4.26380746e-02 6.39318898e-02 8.23190715e-03
 0.00000000e+00 0.00000000e+00 1.00730193e+00 0.00000000e+00
 6.55241758e-02 1.68288314e+00 3.51182040e-04 2.81449128e-03
 0.00000000e+00 2.46222466e-02 0.00000000e+00 2.48476975e-02
 0.00000000e+00 0.00000000e+00 0.00000000e+00 4.51480240e-01
 0.00000000e+00 9.36042368e-01 1.85616940e-01 0.00000000e+00
 1.01621561e-02 0.00000000e+00 0.00000000e+00 0.00000000e+00
 2.98743099e-01 1.15873031e-02 0.00000000e+00 0.00000000e+00
 0.00000000e+00 1.77908492e+00 0.00000000e+00 0.00000000e+00
 1.95487309e-03 6.27002239e-01 1.05097897e-01 1.88223213e-01
 2.10856069e-02 3.75918336e-02 0.00000000e+00 3.90055738e-02
 6.94270134e-01 0.00000000e+00 0.00000000e+00 0.00000000e+00
 1.73526742e-02 2.13796258e-01 1.57990884e-02 2.52822950e-03
 0.00000000e+00 3.07520419e-01 2.30204955e-01 0.00000000e+00
 1.65130049e-02 0.00000000e+00 2.61859829e-03 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 2.77057588e-01
 0.00000000e+00 0.00000000e+00 3.46266538e-01 1.53096449e-02
 7.66901433e-01 0.00000000e+00 4.57348108e-01 0.00000000e+00
 4.47551608e-02 0.00000000e+00 1.29001811e-02 9.60844517e+00
 1.83263212e-01 0.00000000e+00 0.00000000e+00 2.20592483e-03
 1.92206912e-02 0.00000000e+00 0.00000000e+00 1.69307023e-01
 0.00000000e+00 5.36234751e-02 4.14906979e-01 0.00000000e+00
 2.16509521e-01 0.00000000e+00 3.31642979e-04 2.21846998e-02
 3.79920542e-01 4.34444286e-02 6.45953864e-02 2.20743590e-03
 1.08092114e-04 4.73324172e-02 1.29397601e-01 7.50416666e-02
 1.76223174e-01 0.00000000e+00 3.32093821e-03 1.69874787e-01
 1.46595649e-02 6.53675377e-01 3.89797002e-01 1.04904175e-01
 0.00000000e+00 0.00000000e+00 0.00000000e+00 4.29831678e-03
 3.87948332e-03 2.91599263e-03 3.26905074e-03 0.00000000e+00
 2.90803492e-01 0.00000000e+00 0.00000000e+00 5.06584644e-01
 0.00000000e+00 5.63492417e-01 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 1.84580944e-02
 0.00000000e+00 0.00000000e+00 4.36294498e-03 0.00000000e+00
 2.54044100e-03 4.57144417e-02 0.00000000e+00 1.54448688e-01
 0.00000000e+00 4.85616969e-04 8.60927254e-02 0.00000000e+00
 0.00000000e+00 2.07666936e-03 0.00000000e+00 0.00000000e+00
 0.00000000e+00 1.50027208e-03 9.54792649e-02 4.27831672e-02
 0.00000000e+00 6.30120873e-01 0.00000000e+00 3.97839509e-02
 7.61842057e-02 0.00000000e+00 0.00000000e+00 2.26972878e-01
 1.12283370e-03 6.20151758e-01 1.42719946e-03 0.00000000e+00
 1.00419767e-01 2.81603709e-02 0.00000000e+00 0.00000000e+00
 9.86023573e-04 1.03108816e-01 0.00000000e+00 2.15150602e-02
 4.03263509e-01 2.05507502e-03 3.26754898e-01 0.00000000e+00
 2.59594875e-04 2.69558579e-02 8.53820622e-01 0.00000000e+00
 3.48454836e-04 2.09822673e-02 1.37537271e-01 3.52560669e-01
 0.00000000e+00 7.01022102e-03 9.55130458e-02 0.00000000e+00
 8.24151337e-02 0.00000000e+00 1.76009443e-02 0.00000000e+00
 0.00000000e+00 1.08500555e-01 0.00000000e+00 3.22033912e-01
 0.00000000e+00 0.00000000e+00 0.00000000e+00 7.81232864e-02
 1.58517912e-01 7.40836933e-03 0.00000000e+00 3.71387601e-03
 0.00000000e+00 0.00000000e+00 1.59629971e-01 2.65542537e-01
 0.00000000e+00 8.70115831e-02 0.00000000e+00 0.00000000e+00
 0.00000000e+00 6.05189689e-02 3.29940505e-02 2.14093909e-01
 1.14243489e-03 0.00000000e+00 0.00000000e+00 0.00000000e+00
 1.02942371e+00 0.00000000e+00 2.09552914e-01 2.71797739e-03
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 4.40212229e-04 5.45027247e-03 5.26772976e-01
 0.00000000e+00 1.14633166e-03 0.00000000e+00 0.00000000e+00
 2.87284225e-01 0.00000000e+00 1.63603167e-03 1.45751500e+00
 3.32789612e-03 2.14510217e-01 0.00000000e+00 4.59172159e-01
 3.64167895e-03 2.70983785e-01 4.74257320e-02 0.00000000e+00
 0.00000000e+00 1.24228196e-02 2.62936912e-02 0.00000000e+00
 3.59746575e-01 7.43420571e-02 3.23249474e-02 0.00000000e+00
 0.00000000e+00 0.00000000e+00 5.03963754e-02 1.95186436e-02
 1.88063100e-01 0.00000000e+00 1.20775901e-01 0.00000000e+00
 1.51033734e-03 3.76193613e-01 2.53435951e-02 4.63781655e-01
 0.00000000e+00 2.27322485e-02 1.61327139e-01 3.68296541e-02
 0.00000000e+00 7.24332482e-02 1.12533540e-01 0.00000000e+00
 1.47662091e+00 2.31809795e-01 1.10240225e-02 0.00000000e+00
 9.22594313e-03 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 3.93609330e-03
 0.00000000e+00 4.16016132e-02 0.00000000e+00 1.39067611e-02
 4.20212606e-03 6.82298720e-01 0.00000000e+00 5.77424606e-03
 3.18909109e-01 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 3.97320166e-02 0.00000000e+00 9.66454763e-03
 1.68706791e-03 9.37338948e-01 5.82365334e-01 0.00000000e+00
 2.71626227e-02 9.19015054e-03 0.00000000e+00 5.78620434e-01
 0.00000000e+00 0.00000000e+00 4.98304874e-01 0.00000000e+00
 1.53324246e-01 0.00000000e+00 9.36354324e-03 7.59849732e-04
 4.09154454e-03 0.00000000e+00 0.00000000e+00 0.00000000e+00
 4.72321846e-02 1.22074913e-02 1.72334030e-01 1.43685574e-02
 0.00000000e+00 1.72083646e-01 0.00000000e+00 0.00000000e+00
 5.25263604e-03 3.82415615e-02 2.67155468e-02 4.69895273e-01
 8.12048384e-04 0.00000000e+00 2.35244378e-01 0.00000000e+00
 2.02912450e-01 7.60719597e-01 0.00000000e+00 7.81607628e-03
 0.00000000e+00 0.00000000e+00 0.00000000e+00 2.35119201e-02
 1.60810798e-01 3.29564675e-03 0.00000000e+00 0.00000000e+00
 8.54936913e-02 0.00000000e+00 0.00000000e+00 0.00000000e+00
 1.09886944e+00 2.69365194e-03 6.28775120e-01 0.00000000e+00
 6.47291588e-03 0.00000000e+00 4.37758297e-01 1.72344130e-03
 6.12932503e-01 2.79752994e+00 0.00000000e+00 0.00000000e+00
 5.92835665e-01 3.56558623e-04 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 5.87493777e-01 0.00000000e+00
 0.00000000e+00 0.00000000e+00 1.78457517e-03 8.73932168e-02
 0.00000000e+00 0.00000000e+00 3.72092724e-01 1.71930075e-01
 0.00000000e+00 0.00000000e+00 2.36916775e-03 4.34784824e-03
 1.44674852e-01 0.00000000e+00 0.00000000e+00 2.22914037e-03
 1.44575330e-04 4.66828644e-02 1.36799866e-03 0.00000000e+00
 0.00000000e+00 0.00000000e+00 1.36370867e-01 1.12261556e-01
 5.76316357e-01 8.58622909e-01 1.30526507e-02 1.13718258e-03
 5.73361106e-02 0.00000000e+00 6.23522885e-02 6.39145151e-02
 0.00000000e+00 0.00000000e+00 2.13912219e-01 0.00000000e+00
 3.46761271e-02 0.00000000e+00 0.00000000e+00 1.70124456e-01
 1.28269657e-01 0.00000000e+00 9.97253358e-02 1.33464068e-01
 0.00000000e+00 4.34132246e-03 5.00533469e-02 1.37850523e-01
 8.44503939e-02 0.00000000e+00 1.26904202e+00 2.30286643e-01
 1.70187443e-01 2.65877880e-02 1.00804027e-03 2.93543369e-01
 0.00000000e+00 0.00000000e+00 9.06009138e-01 0.00000000e+00]
```

In [18]:
```python
#create a NumPy array for the embeddings aligned with the first 29,979 rows in the CSV
embedding_keys = [f'reliance_{i+1}' for i in range(29979)]
embeddings = np.array([kv_model[key] for key in embedding_keys])
```

In [19]:
```python
#align these embeddings with the OHLC data
ohlc_data = df.iloc[:29979].copy()
```

In [21]:
```python
#use 'Close' prices to calculate returns for labeling (buy/sell signals)
ohlc_data['Return'] = ohlc_data['close'].pct_change().shift(-1)
ohlc_data['Signal'] = np.where(ohlc_data['Return'] > 0, 1, 0)
```

In [22]:
```python
#cheking the structure of the ohlc data with new columns
print("\nOHLC Data with Returns and Signals (First 5 rows):")
print(ohlc_data.head())
```

```
OHLC Data with Returns and Signals (First 5 rows):
                        date    open    high     low   close  volume  \
0  2015-03-20 09:15:00+05:30  425.00  426.25  425.00  425.00    1186
1  2015-03-20 09:16:00+05:30  424.50  425.00  424.20  425.00     768
2  2015-03-20 09:17:00+05:30  425.00  425.90  425.00  425.15     425
3  2015-03-20 09:18:00+05:30  425.90  425.95  425.85  425.85     659
4  2015-03-20 09:19:00+05:30  426.25  426.40  425.15  425.15    1548

     Return  Signal
0  0.000000       0
1  0.000353       1
2  0.001646       1
3 -0.001644       0
4 -0.002587       0
```

In [23]:
```python
#split the data into training and test sets (80% training, 20% test)
X_train, X_test, y_train, y_test = train_test_split(embeddings, ohlc_data['Signal'].dropna(), test_size=0.2, random_state=42)
```

In [24]:
```python
#Use a simple classifier (Random Forest) to predict price movements
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)
```

Out[24]:
```
RandomForestClassifier(random_state=42)
```

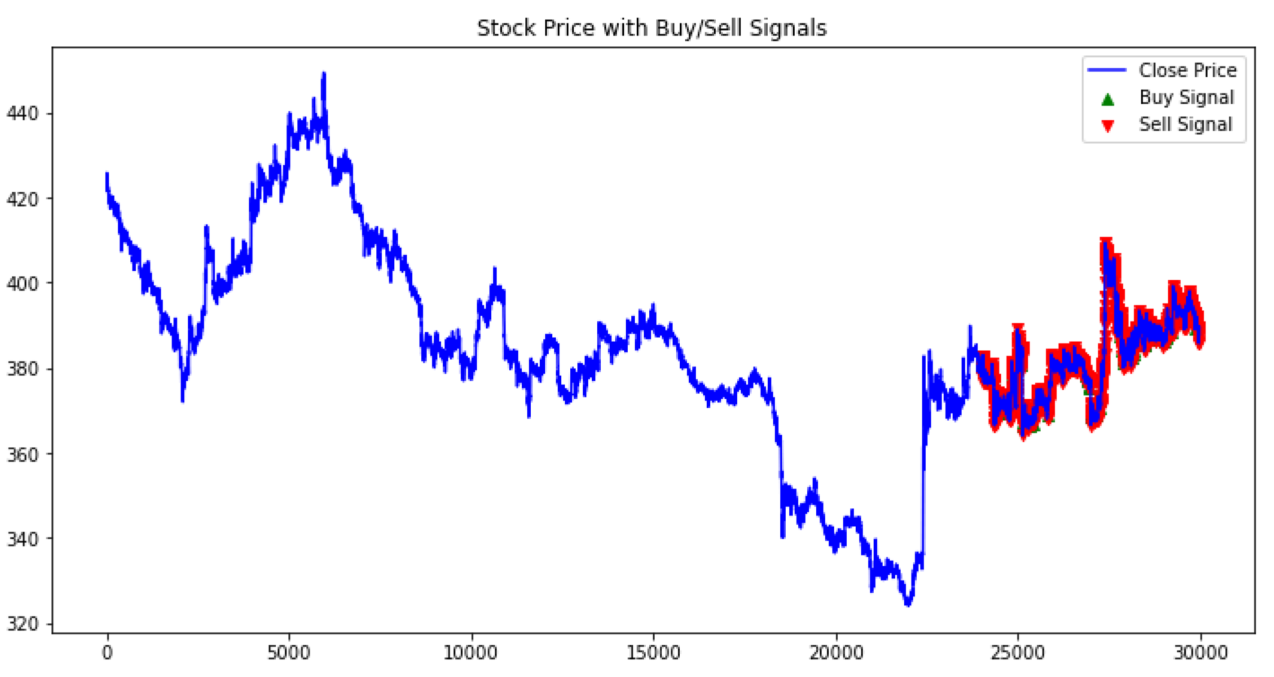
In [25]:
```python
#predict on the test set
y_pred = clf.predict(X_test)
```

In [26]:
```python
#evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"\nAccuracy of the model: {accuracy * 100:.2f}%")
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy of the model: 62.58%

Classification Report:
              precision    recall  f1-score   support

           0       0.63      0.96      0.77      3801
           1       0.39      0.04      0.07      2195

    accuracy                           0.63      5996
   macro avg       0.51      0.50      0.42      5996
weighted avg       0.55      0.63      0.51      5996
```

In [27]:
```python
#visualize Buy/Sell signals on the Close price chart
ohlc_data['Predicted Signal'] = np.nan
ohlc_data.iloc[X_train.shape[0]:, ohlc_data.columns.get_loc('Predicted Signal')] = y_pred
```

In [30]:
```python
#plot the Close price and predicted buy/sell signals
plt.figure(figsize=(12, 6))
plt.plot(ohlc_data['close'], label='Close Price', color='b')
buy_signals = ohlc_data[(ohlc_data['Predicted Signal'] == 1)]
sell_signals = ohlc_data[(ohlc_data['Predicted Signal'] == 0)]
plt.scatter(buy_signals.index, buy_signals['close'], marker='^', color='g', label='Buy Signal', alpha=1)
plt.scatter(sell_signals.index, sell_signals['close'], marker='v', color='r', label='Sell Signal', alpha=1)
plt.title('Stock Price with Buy/Sell Signals')
plt.legend()
plt.show()
```



In [ ]:
```python

```

In [ ]:
```python
#Overview of the Strategy
1)Using the insights from the vector embeddings, a classification-based trading strategy is developed.
2)The vectors are treated as features, and we aim to predict whether the stock price will go up or down in the next time window.
3)Based on this prediction, we execute trades to maximize profits.

#Strategy Details
#Features
1)Pre-calculated Vectors: These vectors are used as features to predict future stock price movements.
2)OHLC Data: The "close" price from the OHLC data is used to calculate returns and determine buy/sell signals.
#Buy/Hold/Sell Rules
1)Buy: Enter a buy position if the model predicts that the stock price will rise in the next time window.
2)Hold: Hold the position if the prediction remains unchanged.
3)Sell: Exit (sell) if the model predicts a downward movement in the next time window.
4)Stop Loss/Take Profit: Incorporating risk management through stop-loss and
  take-profit levels ensures that the strategy minimizes potential losses and locks in gains.
```