QUERY 1-

4.1 Import the csv file of the stock which contained the Bollinger columns as well. Create a new column 'Call' , whose entries are - 'Buy' if the stock price is below the lower Bollinger band 'Hold Buy/ Liquidate Short' if the stock price is between the lower and middle Bollinger band 'Hold Short/ Liquidate Buy' if the stock price is between the middle and upper Bollinger band 'Short' if the stock price is above the upper Bollinger band Now train a classification model with the 3 bollinger columns and the stock price as inputs and 'Calls' as output. Check the accuracy on a test set. (There are many classifier models to choose from, try each one out and compare the accuracy for each) Import another stock data and create the bollinger columns. Using the already defined model, predict the daily calls for this new stock.

In [1]:
```
import numpy as np
import pandas as pd
import warnings
import matplotlib.pyplot as  plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
warnings.filterwarnings('ignore')
```

In [2]:
```
data=pd.read_csv('week3.csv')
```

In [3]:
```
data.isnull().sum()
```

Out[3]:
```
Date                       0
Symbol                     0
Series                     0
Prev Close                 0
Open Price                 0
High Price                 0
Low Price                  0
Last Price                 0
Close Price                0
Average Price              0
Total Traded Quantity      0
Turnover                   0
No. of Trades              0
Deliverable Qty            0
% Dly Qt to Traded Qty     0
month                      0
Day_Perc_Change            0
Trend                      0
ave                       13
upper                     13
lower                     13
dtype: int64
```

In [4]:
```
data.dropna(inplace=True)
data.reset_index(inplace=True)
```

In [5]:
```
data['Call']=0
for i in np.arange(data.ave.size):
    if data['Average Price'][i]>=data.upper[i] :
        print(i)
        data['Call'][i]='Short'
    elif data['Average Price'][i]<=data.lower[i] :
        print(i)
        data['Call'][i]='Buy'
    elif (data['Average Price'][i]>data.lower[i]) and (data['Average Price'][i]<=data.ave[i]) :
        print(i)
        data['Call'][i]='Hold Buy/Liquidate Short'
    else:
        print(i)
        data['Call'][i]='Hold Short/Liquidate Buy'
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```

In [6]:
```
data.Call.value_counts()
```

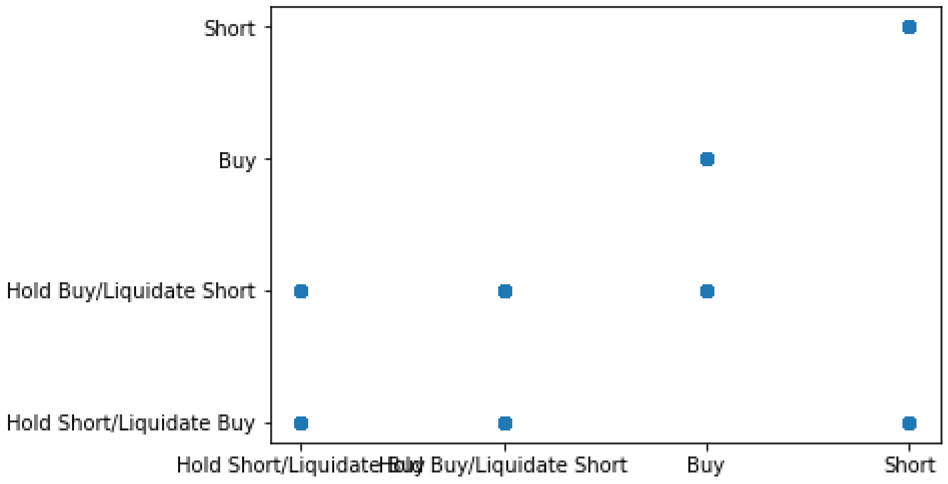
Out[6]:
```
Hold Buy/Liquidate Short    245
Hold Short/Liquidate Buy    204
Buy                          17
Short                        15
Name: Call, dtype: int64
```

In [7]:
```
RFX=data[['Average Price','upper','lower','ave']]
RFy=data['Call']
RFX_train, RFX_test, RFy_train, RFy_test= train_test_split(RFX, RFy, test_size=.25, random_state=42)
```

In [8]:
```
classifier=RandomForestClassifier(n_estimators =200, n_jobs=-1, criterion='entropy', random_state=23, max_depth=10)
classifier.fit(RFX_train, RFy_train)
```

Out[8]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
            max_depth=10, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=200, n_jobs=-1,
            oob_score=False, random_state=23, verbose=0, warm_start=False)
```

In [9]:
```
RFy_pred=classifier.predict(RFX)
plt.scatter(RFy, RFy_pred)
plt.show()
```



In [10]:
```
check=[RFy.values, RFy_pred]
check=pd.DataFrame(check)
check=check.T
check.columns=['Call','Prediction']
check
```

Out[10]:

| | Call | Prediction |
|---|---|---|
| 0 | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| 1 | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| 2 | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| 3 | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| 4 | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| 5 | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| 6 | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| 7 | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| 8 | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| 9 | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| 10 | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| 11 | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| 12 | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| 13 | Buy | Buy |
| 14 | Buy | Buy |
| 15 | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| 16 | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| 17 | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| 18 | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| 19 | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| 20 | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| 21 | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| 22 | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| 23 | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| 24 | Hold Short/Liquidate Buy | Hold Buy/Liquidate Short |

| | Call | Prediction |
|---|---|---|
| **25** | Hold Short/Liquidate Buy | Hold Buy/Liquidate Short |
| **26** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **27** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **28** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **29** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **...** | ... | ... |
| **451** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **452** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **453** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **454** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **455** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **456** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **457** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **458** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **459** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **460** | Hold Buy/Liquidate Short | Hold Short/Liquidate Buy |
| **461** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **462** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **463** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **464** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **465** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **466** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **467** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **468** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **469** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **470** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **471** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **472** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **473** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **474** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |

|  | Call | Prediction |
|---|---|---|
| **475** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **476** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **477** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **478** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **479** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **480** | Buy | Buy |

481 rows × 2 columns

In [11]:
```
f=0
for i in np.arange(len(data.ave)):
    if check.iloc[i,0]!=check.iloc[i,1]:
        f=f+1
print(f)
accuracy=(RFy_test.size-f)/RFy_test.size
accuracy
```

```
28
```

Out[11]:
```
0.768595041322314
```

In [12]:
```
data=pd.read_csv('lt.csv')
```

```
In [13]: data.isnull().sum()
```

```
Out[13]: Date                          0
         Symbol                        0
         Series                        0
         Prev Close                    0
         Open Price                    0
         High Price                    0
         Low Price                     0
         Last Price                    0
         Close Price                   0
         Average Price                 0
         Total Traded Quantity         0
         Turnover                      0
         No. of Trades                 0
         Deliverable Qty               0
         % Dly Qt to Traded Qty        0
         month                         0
         Day_Perc_Change               0
         Trend                         0
         ave                          13
         upper                        13
         lower                        13
         dtype: int64
```

```
In [14]: data.dropna(inplace=True)
         data.reset_index(inplace=True)
```

In [15]:

```python
data['Call']=0
for i in np.arange(data.ave.size):
    if data['Average Price'][i]>=data.upper[i] :
        print(i)
        data['Call'][i]='Short'
    elif data['Average Price'][i]<=data.lower[i] :
        print(i)
        data['Call'][i]='Buy'
    elif (data['Average Price'][i]>data.lower[i]) and (data['Average Price'][i]<=data.ave[i]):
        print(i)
        data['Call'][i]='Hold Buy/Liquidate Short'
    else:
        print(i)
        data['Call'][i]='Hold Short/Liquidate Buy'
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```

In [16]:

```python
data.Call.value_counts()
```

Out[16]:

```
Hold Short/Liquidate Buy    225
Hold Buy/Liquidate Short    225
Buy                          16
Short                        15
Name: Call, dtype: int64
```

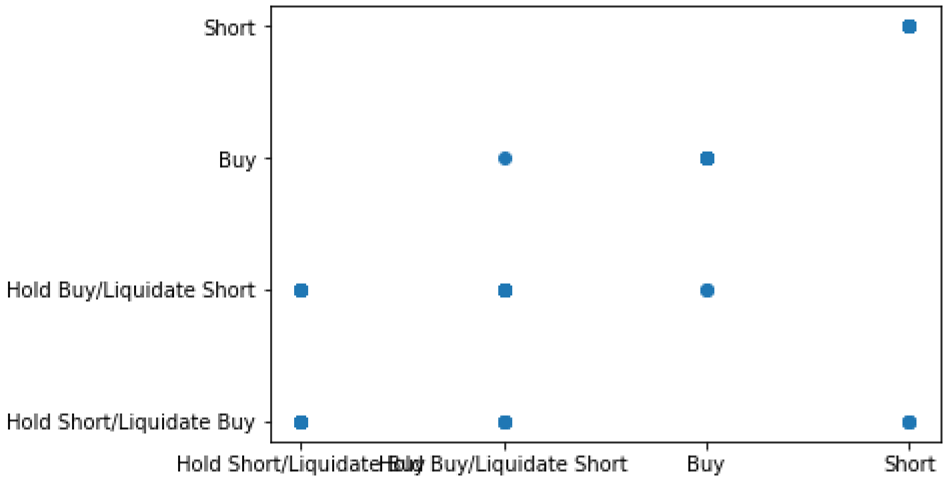
In [17]:

```python
A=data[['Average Price','upper','lower','ave']]
B=data['Call']
A_train, A_test, B_train, B_test= train_test_split(A, B, test_size=.25, random_state=42)
```

```
In [18]: classifier=RandomForestClassifier(n_estimators =200, n_jobs=-1, criterion='entropy', random_state=23, max_depth=10)
         classifier.fit(A_train, B_train)
```

```
Out[18]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
                     max_depth=10, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=200, n_jobs=-1,
                     oob_score=False, random_state=23, verbose=0, warm_start=False)
```

```
In [19]: B_pred=classifier.predict(A)
         plt.scatter(B, B_pred)
         plt.show()
```



In [20]:
```
check=[B.values, B_pred]
check=pd.DataFrame(check)
check=check.T
check.columns=['Call','Prediction']
check
```

Out[20]:

| | Call | Prediction |
|---|---|---|
| **0** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **1** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **2** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **3** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **4** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **5** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **6** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **7** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **8** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **9** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **10** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **11** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **12** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **13** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **14** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **15** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **16** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **17** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **18** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **19** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **20** | Buy | Buy |
| **21** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **22** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **23** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **24** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |

| | Call | Prediction |
|---|---|---|
| **25** | Hold Buy/Liquidate Short | Hold Short/Liquidate Buy |
| **26** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **27** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **28** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **29** | Buy | Buy |
| **...** | ... | ... |
| **451** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **452** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **453** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **454** | Short | Short |
| **455** | Short | Hold Short/Liquidate Buy |
| **456** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **457** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **458** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **459** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **460** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **461** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **462** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **463** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **464** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **465** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **466** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **467** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **468** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **469** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **470** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **471** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **472** | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| **473** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| **474** | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |

| | Call | Prediction |
|---|---|---|
| 475 | Hold Buy/Liquidate Short | Hold Short/Liquidate Buy |
| 476 | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| 477 | Hold Short/Liquidate Buy | Hold Short/Liquidate Buy |
| 478 | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| 479 | Hold Buy/Liquidate Short | Hold Buy/Liquidate Short |
| 480 | Buy | Buy |

481 rows × 2 columns

In [21]:
```
f=0
for i in np.arange(len(data.ave)):
    if check.iloc[i,0]!=check.iloc[i,1]:
        f=f+1
print(f)
accuracy=(B_test.size-f)/B_test.size
accuracy
```

```
19
```

Out[21]:
```
0.8429752066115702
```

QUERY 2-

4.2 Now, we'll again utilize classification to make a trade call, and measure the efficiency of our trading algorithm over the past two years. For this assignment , we will use RandomForest classifier. Import the stock data file of your choice Define 4 new columns , whose values are: % change between Open and Close price for the day % change between Low and High price for the day 5 day rolling mean of the day to day % change in Close Price 5 day rolling std of the day to day % change in Close Price Create a new column 'Action' whose values are: 1 if next day's price(Close) is greater than present day's. (-1) if next day's price(Close) is less than present day's. i.e. Action [ i ] = 1 if Close[ i+1 ] > Close[ i ] i.e. Action [ i ] = (-1) if Close[ i+1 ] < Close[ i ] Construct a classification model with the 4 new inputs and 'Action' as target Check the accuracy of this model , also , plot the net cumulative returns (in %) if we were to follow this algorithmic model

In [22]:
```
data = pd.read_csv('Nifty50.csv')
data['Close Price']=data['Close']
data['Open Price']=data['Open']
data['High Price']=data['High']
data['Low Price']=data['Low']
data['Day_Perc_Change'] = 100*data['Close Price'].pct_change()
data.iloc[0,-1]=0
data['Open-Close'] = (data['Open Price'] - data['Close Price'])/data['Open Price']
data['High-Low'] = (data['High Price'] - data['Low Price'])/data['Low Price']
data['std_5'] = data['Day_Perc_Change'].rolling(5).std()
data['ret_5'] = data['Day_Perc_Change'].rolling(5).mean()
data.dropna(inplace=True)
```

In [23]:
```
data['Action']=np.where(data['Close Price'].shift(-1) > data['Close Price'], 1, -1)
```

In [24]:
```
X = data[['Open-Close','High-Low','std_5','ret_5']]
y = data['Action']
```

In [25]:
```
result = pd.concat([data['Close Price'], y], axis =1, join='inner')
result
```

Out[25]:

| | Close Price | Action |
|---|---|---|
| **4** | 9427.90 | 1 |
| **5** | 9438.25 | -1 |
| **6** | 9386.15 | -1 |
| **7** | 9360.55 | 1 |
| **8** | 9509.75 | 1 |
| **9** | 9595.10 | 1 |
| **10** | 9604.90 | 1 |
| **11** | 9624.55 | -1 |
| **12** | 9621.25 | -1 |
| **13** | 9616.10 | 1 |
| **14** | 9653.50 | 1 |
| **15** | 9675.10 | -1 |
| **16** | 9637.15 | 1 |
| **17** | 9663.90 | -1 |
| **18** | 9647.25 | 1 |
| **19** | 9668.25 | -1 |
| **20** | 9616.40 | -1 |
| **21** | 9606.90 | 1 |
| **22** | 9618.15 | -1 |
| **23** | 9578.05 | 1 |
| **24** | 9588.05 | 1 |
| **25** | 9657.55 | -1 |
| **26** | 9653.50 | -1 |
| **27** | 9633.60 | -1 |
| **28** | 9630.00 | -1 |
| **29** | 9574.95 | -1 |
| **30** | 9511.40 | -1 |

|  | Close Price | Action |
|---|---|---|
| **31** | 9491.25 | 1 |
| **32** | 9504.10 | 1 |
| **33** | 9520.90 | 1 |
| **...** | ... | ... |
| **464** | 11445.05 | 1 |
| **465** | 11570.00 | 1 |
| **466** | 11623.90 | 1 |
| **467** | 11669.15 | 1 |
| **468** | 11713.20 | -1 |
| **469** | 11643.95 | -1 |
| **470** | 11598.00 | 1 |
| **471** | 11665.95 | -1 |
| **472** | 11604.50 | 1 |
| **473** | 11671.95 | -1 |
| **474** | 11584.30 | 1 |
| **475** | 11596.70 | 1 |
| **476** | 11643.45 | 1 |
| **477** | 11690.35 | 1 |
| **478** | 11787.15 | -1 |
| **479** | 11752.80 | -1 |
| **480** | 11594.45 | -1 |
| **481** | 11575.95 | 1 |
| **482** | 11726.15 | -1 |
| **483** | 11641.80 | 1 |
| **484** | 11754.65 | -1 |
| **485** | 11748.15 | -1 |
| **486** | 11724.75 | -1 |
| **487** | 11712.25 | -1 |
| **488** | 11598.25 | -1 |
| **489** | 11497.90 | -1 |

|     | Close Price | Action |
|-----|-------------|--------|
| **490** | 11359.45 | -1 |
| **491** | 11301.80 | -1 |
| **492** | 11278.90 | -1 |
| **493** | 11148.20 | -1 |

490 rows × 2 columns

In [26]:
```
X_train, X_test, y_train, y_test= train_test_split(X, y, test_size=.25, random_state=42)
print(X_train.shape, X_test.shape)
print(y_train.shape, y_test.shape)
```

```
(367, 4) (123, 4)
(367,) (123,)
```

In [27]:
```
clf = RandomForestClassifier(random_state=5)
```

In [28]:
```
model = clf.fit(X_train, y_train)
```

In [29]:
```
from sklearn.metrics import accuracy_score
print('Correct Prediction (%)', accuracy_score(y_test, model.predict(X_test), normalize=True)*100.0)
```

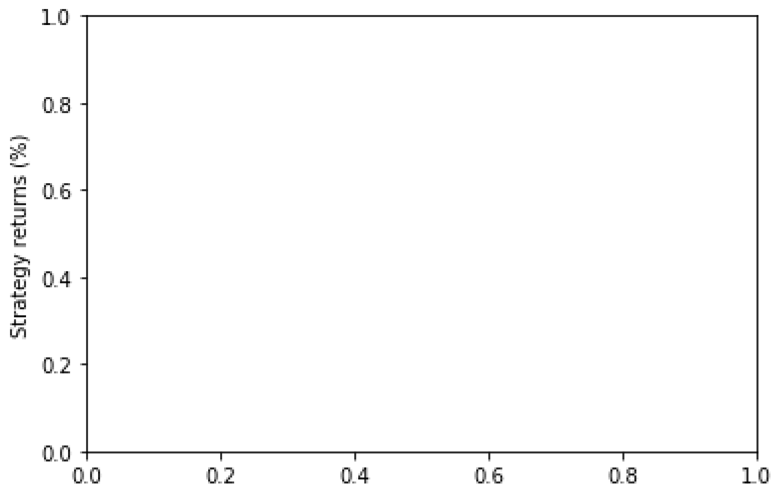
```
Correct Prediction (%) 51.21951219512195
```

In [30]:
```
data['strategy_returns'] = data.Day_Perc_Change * model.predict(X)
```

In [31]:
```
data
```
Out[31]:

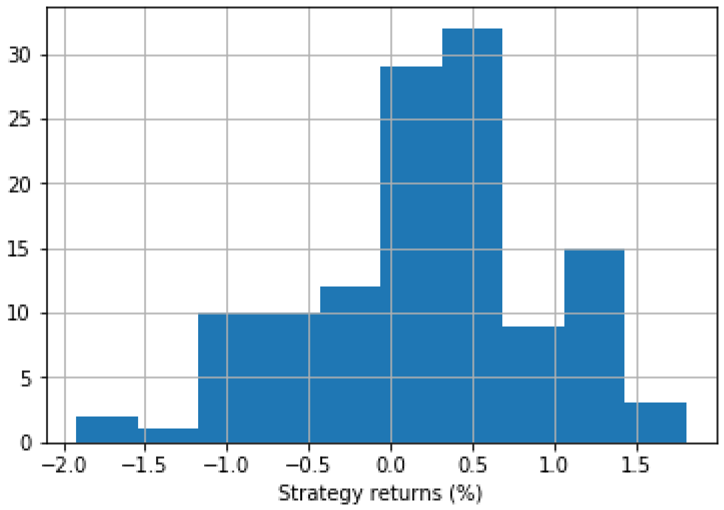
| | Date | Open | High | Low | Close | Shares Traded | Turnover (Rs. Cr) | Close Price | Open Price | High Price | Low Price | Day_Perc_Change | Open-Close | I |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **4** | 19-May-2017 | 9469.90 | 9505.75 | 9390.75 | 9427.90 | 259861396 | 11544.77 | 9427.90 | 9469.90 | 9505.75 | 9390.75 | -0.016438 | 0.004435 | 0.01 |
| **5** | 22-May-2017 | 9480.25 | 9498.65 | 9427.90 | 9438.25 | 202874757 | 9432.97 | 9438.25 | 9480.25 | 9498.65 | 9427.90 | 0.109781 | 0.004430 | 0.00 |
| **6** | 23-May-2017 | 9445.05 | 9448.05 | 9370.00 | 9386.15 | 231345629 | 11553.27 | 9386.15 | 9445.05 | 9448.05 | 9370.00 | -0.552009 | 0.006236 | 0.00 |
| **7** | 24-May-2017 | 9410.90 | 9431.90 | 9341.65 | 9360.55 | 218265181 | 11045.95 | 9360.55 | 9410.90 | 9431.90 | 9341.65 | -0.272742 | 0.005350 | 0.00 |
| **8** | 25-May-2017 | 9384.05 | 9523.30 | 9379.20 | 9509.75 | 298147347 | 16964.26 | 9509.75 | 9384.05 | 9523.30 | 9379.20 | 1.593923 | -0.013395 | 0.01 |

In [32]:
```
((data.strategy_returns[y_train.size:]+100)/100).cumprod().plot
plt.ylabel('Strategy returns (%)')
plt.show()
```



I could not resolve the issue above of graph I tried same with other stocks also graph is still blank can you plz help where did I go wrong

In [33]:
```
data.strategy_returns[y_train.size:].hist()
plt.xlabel('Strategy returns (%)')
plt.show()
```


Strategy returns (%)