stock_data_RandomForest

April 24, 2018

```
In [1]: %matplotlib inline
        import matplotlib.pyplot as plt
        import pandas as pd
        from sklearn.metrics import accuracy_score
        from sklearn.model_selection import train_test_split
        from sklearn.ensemble import RandomForestClassifier
        # from sklearn.linear_model import LogisticRegression
        # from sklearn.linear_model import LinearRegression

In [2]: df = pd.read_csv('stock_data.csv')

In [3]: df.shape

Out[3]: (3000, 101)

In [4]: Y = df['Y']
        X = df.drop(['Y'],axis=1)
        X.head()

0    1
1   -1
2    1
3    1
4    1
Name: Y, dtype: int64


Out[4]:            X1          X2          X3          X4          X5          X6  \
        0  100.000000  100.000000  100.000000  100.000000  100.000000  100.000000
        1   91.567862   96.621901   98.084599   96.543909   91.678881   96.337918
        2   92.908600   97.640753  103.112398   98.130312   78.041958   97.693843
        3   84.869178   94.369892  109.018217  102.209632   73.357203  100.676878
        4   88.888889   95.603221  108.459686  103.116147   77.832727  104.677398

                 X7          X8          X9         X10     ...             X91  \
        0  100.0000  100.000000  100.000000  100.000000     ...      100.000000
        1   96.1250   94.015957   94.409201   97.084183     ...       97.292714
        2   95.6250   95.744681   94.089712   96.655229     ...       95.093172
```

```
3   96.9166   93.085106   98.243067   98.113343     ...       97.461792
4  100.8750   93.617021  100.000000   99.828542     ...      102.537953

           X92         X93         X94         X95         X96         X97  \
0  100.000000  100.000000  100.000000  100.000000  100.000000  100.000000
1   96.888889   95.499022   97.457971   99.282297  101.296698   98.310678
2   97.333333   96.379804   93.397222  100.478469  101.179953   98.141624
3  101.925926   95.988415   94.412410  100.956938   99.287025  102.027024
4  102.074074  102.152642  105.579469  106.937799  101.651101  103.884995

           X98         X99        X100
0  100.000000  100.000000  100.000000
1  100.489134   99.602234   93.263158
2   96.084839   99.204467   86.421053
3  100.326263  103.985313   77.513179
4  106.688139  108.525205   85.736842

[5 rows x 100 columns]
```

In [5]:
```
x_train,x_test,y_train,y_test = train_test_split(X,Y)
```

In [7]:
```
model = RandomForestClassifier()
# model = LinearRegression()
# model = LogisticRegression()
model.fit(x_train,y_train)
```

Out[7]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, 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=10, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [8]:
```
# model.coef_
```

In [9]:
```
# model.intercept_
```

In [10]:
```
# model.predict(X)
model.predict(x_test)
```

Out[10]:
```
array([-1, -1, -1, -1, -1,  1, -1,  1,  1,  1,  1,  1, -1, -1, -1, -1, -1,
       -1,  1,  1, -1,  1, -1,  1, -1,  1,  1, -1, -1,  1,  1,  1,  1, -1,
        1, -1, -1,  1, -1,  1, -1, -1,  1, -1, -1,  1,  1, -1, -1, -1, -1,
        1,  1,  1,  1,  1, -1, -1, -1, -1,  1, -1, -1, -1,  1, -1,  1, -1,
        1, -1,  1,  1, -1, -1, -1,  1, -1, -1,  1, -1,  1, -1,  1,  1,  1,
       -1, -1, -1,  1,  1,  1,  1,  1, -1,  1, -1,  1,  1,  1, -1, -1, -1,
        1,  1,  1, -1, -1,  1, -1, -1, -1, -1, -1,  1, -1, -1,  1, -1,  1,
       -1,  1,  1, -1, -1, -1,  1, -1,  1, -1,  1,  1, -1, -1,  1,  1, -1,
```

```
       -1,  1, -1, -1, -1,  1, -1,  1, -1, -1,  1, -1,  1, -1, -1, -1, -1,
       -1, -1,  1,  1, -1,  1, -1, -1, -1,  1,  1, -1,  1,  1,  1,  1, -1,
       -1,  1, -1, -1,  1, -1, -1, -1, -1,  1,  1,  1, -1,  1, -1, -1, -1,
        1, -1, -1, -1, -1, -1, -1,  1, -1,  1,  1, -1, -1,  1, -1,  1,  1,
       -1, -1, -1, -1,  1, -1, -1,  1, -1,  1, -1,  1, -1, -1,  1,  1,  1,
        1,  1,  1, -1,  1, -1,  1, -1, -1,  1, -1,  1, -1, -1, -1, -1, -1,
       -1,  1, -1, -1,  1, -1, -1,  1, -1, -1, -1, -1, -1, -1,  1,  1, -1,
        1,  1,  1,  1,  1, -1, -1,  1, -1,  1, -1,  1, -1, -1,  1,  1,  1,
       -1, -1,  1,  1, -1,  1, -1, -1, -1,  1, -1, -1, -1,  1,  1, -1,  1,
       -1,  1,  1, -1, -1,  1,  1, -1,  1, -1,  1, -1,  1,  1, -1,  1, -1,
        1, -1, -1, -1,  1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1,  1, -1,  1,  1, -1, -1, -1,  1, -1,  1, -1, -1, -1, -1,  1,  1,
       -1, -1,  1, -1,  1, -1,  1, -1,  1,  1,  1, -1, -1, -1,  1, -1, -1,
       -1,  1, -1, -1, -1,  1,  1,  1, -1,  1,  1,  1, -1, -1,  1, -1,  1,
        1, -1,  1,  1, -1,  1,  1,  1,  1, -1,  1, -1,  1, -1,  1,  1, -1,
       -1, -1,  1, -1, -1, -1, -1,  1,  1,  1, -1,  1, -1,  1, -1, -1, -1,
       -1, -1, -1,  1, -1, -1,  1, -1, -1, -1,  1,  1, -1,  1,  1,  1, -1,
       -1, -1,  1, -1,  1,  1, -1,  1, -1, -1,  1,  1,  1,  1, -1,  1,  1,
       -1, -1,  1, -1,  1, -1,  1,  1, -1, -1,  1,  1,  1, -1,  1,  1, -1,
        1, -1,  1,  1, -1, -1, -1,  1,  1, -1, -1, -1, -1,  1, -1,  1, -1,
        1, -1,  1, -1, -1,  1, -1,  1,  1, -1,  1, -1,  1,  1, -1,  1,  1,
       -1,  1, -1, -1,  1, -1,  1, -1, -1,  1,  1, -1, -1, -1, -1, -1, -1,
        1,  1, -1, -1, -1,  1, -1, -1, -1, -1,  1, -1, -1,  1, -1, -1, -1,
        1,  1, -1, -1, -1,  1, -1, -1, -1, -1, -1, -1, -1,  1,  1, -1,  1,
        1,  1, -1, -1, -1,  1,  1,  1,  1, -1, -1,  1, -1,  1,  1, -1, -1,
       -1, -1, -1,  1, -1,  1, -1,  1, -1, -1, -1, -1, -1, -1, -1, -1,  1,
        1, -1, -1,  1, -1,  1,  1, -1, -1,  1, -1, -1,  1, -1, -1,  1, -1,
       -1, -1, -1, -1, -1, -1, -1,  1,  1,  1,  1,  1, -1,  1,  1, -1,  1,
       -1, -1,  1,  1,  1, -1,  1, -1, -1,  1,  1, -1, -1, -1, -1, -1,  1,
       -1,  1, -1, -1, -1, -1, -1, -1,  1, -1,  1,  1,  1, -1,  1, -1, -1,
        1, -1,  1, -1, -1, -1, -1,  1,  1, -1,  1, -1, -1, -1, -1, -1, -1,
       -1,  1, -1, -1, -1, -1,  1, -1,  1,  1, -1, -1, -1, -1, -1,  1, -1,
       -1,  1, -1, -1,  1,  1, -1,  1, -1, -1,  1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1,  1, -1,  1,  1, -1, -1, -1,  1,  1, -1, -1, -1,
       -1, -1, -1,  1, -1,  1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  1,  1,
        1, -1,  1,  1, -1,  1,  1,  1, -1,  1, -1, -1, -1,  1,  1, -1, -1,
       -1, -1])
```

In [11]:
```
model.score(x_test,y_test)
```

Out[11]:
```
0.4746666666666667
```

In [12]:
```
model.score(x_train,y_train)
```

Out[12]:
```
0.9617777777777777
```

In [13]:
```
accuracy_score(y_test,model.predict(x_test))
```

Out[13]:
```
0.4746666666666667
```