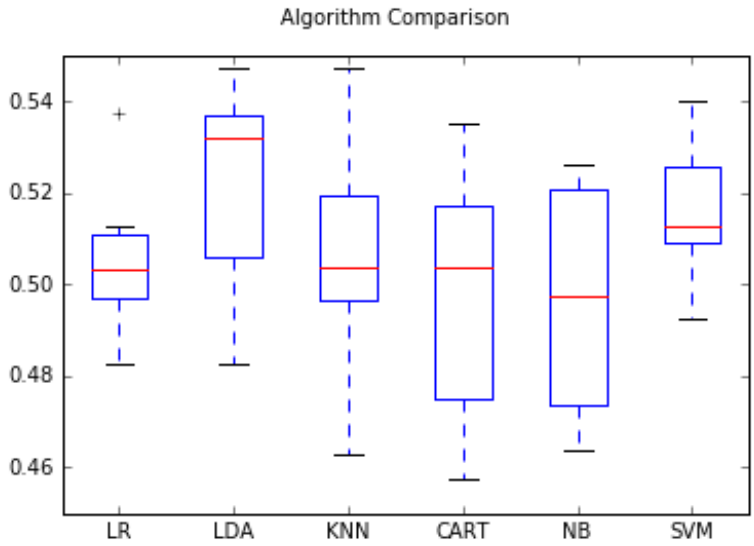
```
Python 2.7.12 |Anaconda 4.2.0 (64-bit)| (default, Jun 29 2016, 11:07:13) [MSC v.1500 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import pandas as pd
   ...: import numpy
   ...: from pandas.tools.plotting import scatter_matrix
   ...: import matplotlib.pyplot as plt
   ...: from sklearn import model_selection
   ...: from sklearn.metrics import classification_report
   ...: from sklearn.metrics import confusion_matrix
   ...: from sklearn.metrics import accuracy_score
   ...: from sklearn.linear_model import LogisticRegression
   ...: from sklearn.tree import DecisionTreeClassifier
   ...: from sklearn.neighbors import KNeighborsClassifier
   ...: from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
   ...: from sklearn.naive_bayes import GaussianNB
   ...: from sklearn.svm import SVC
   ...:
   ...: noaa = pd.read_csv('C:\Users\Dev\Downloads\NOAA_NewYork_Twenty.csv')
   ...: noaadf = pd.DataFrame(noaa)
   ...:
   ...: msft = pd.read_csv('C:\Users\Dev\Downloads\MSFT_Twenty_Compiled.csv')
   ...: msftdf = pd.DataFrame(msft)
   ...:

In [2]: noaadf.columns = ['Date', 'PRCP', 'SNOW', 'TMAX', 'TMIN', 'AWND']
   ...:
   ...:
   ...: noaadf['Date'] = pd.to_datetime(noaadf['Date'].astype(str), format='%Y%m%d')
   ...: msftdf['Date'] = pd.to_datetime(msftdf['Date'].astype(str))
   ...:
   ...: msftdf1 = pd.merge(msftdf, noaadf, how='inner', on=['Date'])
   ...:
   ...: msftdf1.loc[msftdf1.Gain_Loss > 0, 'Final_GorL' ] = int(1)
   ...: msftdf1.loc[msftdf1.Gain_Loss < 0, 'Final_GorL' ] = int(0)
   ...:
   ...: msftdf1.Final_GorL = msftdf1.Final_GorL.fillna(0)
   ...: msftdf1 = msftdf1.fillna(msftdf1.mean())
   ...: msftdf1.isnull().any()
   ...:
   ...: array = msftdf1[['MSFT_Open', 'MSFT_High', 'MSFT_Low', 'MSFT_Volume',
   ...:       'Close', 'SP_Open', 'SP_High', 'SP_Low', 'SP_Close', 'SP_Volume',
   ...:       'NASDAQ_Open', 'NASDAQ_High', 'NASDAQ_Low', 'NASDAQ_Close',
   ...:       'NASDAQ_Volume', 'DJIA_Open', 'DJIA_High', 'DJIA_Low',
   ...:       'DJIA_Close', 'DJIA_Volume', 'AMD_Open', 'AMD_High', 'AMD_Low',
   ...:       'AMD_Close', 'AMD_Volume', 'INTC_Open', 'INTC_High', 'INTC_Low',
   ...:       'INTC_Close', 'INTC_Volume', 'INTC_Adj_Close', 'Final_GorL']].values
   ...: array = numpy.asarray(array, dtype="float")
   ...:

In [3]: X = array[:,1:31]
   ...: Y = array[:,31]
   ...: validation_size = 0.20
   ...: seed = 7
   ...: X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, Y, test_size=validation_size,
random_state=seed)
   ...:
   ...: seed = 7
   ...: scoring = 'accuracy'
   ...:
   ...: # Spot Check Algorithms
   ...: models = []
```

```
...: models.append(('LR', LogisticRegression()))
...: models.append(('LDA', LinearDiscriminantAnalysis()))
...: models.append(('KNN', KNeighborsClassifier()))
...: models.append(('CART', DecisionTreeClassifier()))
...: models.append(('NB', GaussianNB()))
...: models.append(('SVM', SVC()))
...: # evaluate each model in turn
...: results = []
...: names = []
...: for name, model in models:
...:  kfold = model_selection.KFold(n_splits=10, random_state=seed)
...:  cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold, scoring=scoring)
...:  results.append(cv_results)
...:  names.append(name)
...:  msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
...:  print(msg)
...:
...: # Compare Algorithms
...: fig = plt.figure()
...: fig.suptitle('Algorithm Comparison')
...: ax = fig.add_subplot(111)
...: plt.boxplot(results)
...: ax.set_xticklabels(names)
...: plt.show()
...:
LR: 0.503978 (0.014744)
LDA: 0.522111 (0.020065)
KNN: 0.505963 (0.021531)
CART: 0.498509 (0.024655)
NB: 0.496770 (0.023514)
SVM: 0.514909 (0.014761)
```




```
In [4]: # Make predictions on validation dataset
...: cart = DecisionTreeClassifier()
...: cart.fit(X_train, Y_train)
...: predictions = cart.predict(X_validation)
...: print('Decision Tree Classifier')
...: print(accuracy_score(Y_validation, predictions))
...: print(confusion_matrix(Y_validation, predictions))
...: print(classification_report(Y_validation, predictions))
...:
...: lda = LinearDiscriminantAnalysis()
...: lda.fit(X_train, Y_train)
...: predictions = lda.predict(X_validation)
...: print('Linear Discrimination Analysis')
...: print(accuracy_score(Y_validation, predictions))
...: print(confusion_matrix(Y_validation, predictions))
...: print(classification_report(Y_validation, predictions))
```

```
...:
...: gnb = GaussianNB()
...: gnb.fit(X_train, Y_train)
...: predictions = gnb.predict(X_validation)
...: print('Gaussian Naive Bayes')
...: print(accuracy_score(Y_validation, predictions))
...: print(confusion_matrix(Y_validation, predictions))
...: print(classification_report(Y_validation, predictions))
...:
...: lr = LogisticRegression()
...: lr.fit(X_train, Y_train)
...: predictions = lr.predict(X_validation)
...: print('Logistic Regression')
...: print(accuracy_score(Y_validation, predictions))
...: print(confusion_matrix(Y_validation, predictions))
...: print(classification_report(Y_validation, predictions))
...:
...: knn = KNeighborsClassifier()
...: knn.fit(X_train, Y_train)
...: predictions = knn.predict(X_validation)
...: print('K Nearest Neighbors Classifier')
...: print(accuracy_score(Y_validation, predictions))
...: print(confusion_matrix(Y_validation, predictions))
...: print(classification_report(Y_validation, predictions))
...:
...: svm = SVC()
...: svm.fit(X_train, Y_train)
...: predictions = svm.predict(X_validation)
...: print('Support Vector Machine')
...: print(accuracy_score(Y_validation, predictions))
...: print(confusion_matrix(Y_validation, predictions))
...: print(classification_report(Y_validation, predictions))
...:
Decision Tree Classifier
0.526315789474
[[277 229]
 [248 253]]
             precision    recall  f1-score   support

        0.0       0.53      0.55      0.54       506
        1.0       0.52      0.50      0.51       501

avg / total       0.53      0.53      0.53      1007

Linear Discrimination Analysis
0.521350546177
[[321 185]
 [297 204]]
             precision    recall  f1-score   support

        0.0       0.52      0.63      0.57       506
        1.0       0.52      0.41      0.46       501

avg / total       0.52      0.52      0.52      1007

Gaussian Naive Bayes
0.508440913605
[[201 305]
 [190 311]]
             precision    recall  f1-score   support

        0.0       0.51      0.40      0.45       506
        1.0       0.50      0.62      0.56       501

avg / total       0.51      0.51      0.50      1007

Logistic Regression
0.518371400199
```

```
[[460  46]
 [439  62]]
             precision    recall  f1-score   support

        0.0       0.51      0.91      0.65       506
        1.0       0.57      0.12      0.20       501

avg / total       0.54      0.52      0.43      1007
```

K Nearest Neighbors Classifier

```
0.51539225422
[[270 236]
 [252 249]]
             precision    recall  f1-score   support

        0.0       0.52      0.53      0.53       506
        1.0       0.51      0.50      0.51       501

avg / total       0.52      0.52      0.52      1007
```

Support Vector Machine

```
0.502482621648
[[506   0]
 [501   0]]
             precision    recall  f1-score   support

        0.0       0.50      1.00      0.67       506
        1.0       0.00      0.00      0.00       501

avg / total       0.25      0.50      0.34      1007
```

```
C:\Users\Dev\Anaconda2\lib\site-packages\sklearn\metrics\classification.py:1113: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

In [**5**]: