Results:

Part 1 : Digit Recognition - KNN

Overview:

I have used KNeighborsClassifier for this part of assignment. Analysis is done based on number of neighbors. Based on number of neighbors the variations in output are significant:

1) Neighbors : 1
   Accuracy : ~98%

```
In [1]: runfile('/MS_CS/Machine
Learning/Homework3/DigitRecognition.py', wdir='/MS_CS/Machine
Learning/Homework3')
/MS_CS/Machine Learning/Homework3/DigitRecognition.py:41:
DataConversionWarning: A column-vector y was passed when a 1d
array was expected. Please change the shape of y to (n_samples,
), for example using ravel().
  classifier.fit(trainFeaturesMatrix, trainMatrix)
0.978853644964
```

2) Neighbors : 2
   Accuracy : ~98%

```
In [2]: runfile('/MS_CS/Machine
Learning/Homework3/DigitRecognition.py', wdir='/MS_CS/Machine
Learning/Homework3')
/MS_CS/Machine Learning/Homework3/DigitRecognition.py:41:
DataConversionWarning: A column-vector y was passed when a 1d
array was expected. Please change the shape of y to (n_samples,
), for example using ravel().
  classifier.fit(trainFeaturesMatrix, trainMatrix)
0.979966611018
```

3) Neighbors : 3
   Accuracy : ~96%

```
In [3]: runfile('/MS_CS/Machine
Learning/Homework3/DigitRecognition.py', wdir='/MS_CS/Machine
Learning/Homework3')
/MS_CS/Machine Learning/Homework3/DigitRecognition.py:41:
DataConversionWarning: A column-vector y was passed when a 1d
array was expected. Please change the shape of y to (n_samples,
), for example using ravel().
  classifier.fit(trainFeaturesMatrix, trainMatrix)
0.979966611018
```

4) Neighbors : 4
Accuracy : ~92%

```
In [4]: runfile('/MS_CS/Machine
Learning/Homework3/DigitRecognition.py', wdir='/MS_CS/Machine
Learning/Homework3')
/MS_CS/Machine Learning/Homework3/DigitRecognition.py:41:
DataConversionWarning: A column-vector y was passed when a 1d
array was expected. Please change the shape of y to (n_samples,
), for example using ravel().
  classifier.fit(trainFeaturesMatrix, trainMatrix)
0.9816360601
```

5) Neighbors : 7
Accuracy : ~98%

```
In [5]: runfile('/MS_CS/Machine
Learning/Homework3/DigitRecognition.py', wdir='/MS_CS/Machine
Learning/Homework3')
/MS_CS/Machine Learning/Homework3/DigitRecognition.py:41:
DataConversionWarning: A column-vector y was passed when a 1d
array was expected. Please change the shape of y to (n_samples,
), for example using ravel().
  classifier.fit(trainFeaturesMatrix, trainMatrix)
0.978853644964
```

6) Neighbors : 8
Accuracy : ~98%

```
In [6]: runfile('/MS_CS/Machine
Learning/Homework3/DigitRecognition.py', wdir='/MS_CS/Machine
Learning/Homework3')
/MS_CS/Machine Learning/Homework3/DigitRecognition.py:41:
DataConversionWarning: A column-vector y was passed when a 1d
array was expected. Please change the shape of y to (n_samples,
), for example using ravel().
  classifier.fit(trainFeaturesMatrix, trainMatrix)
0.982749026155
```

7)Neighbors : 9
Accuracy : ~97.75%

```
In [7]: runfile('/MS_CS/Machine
Learning/Homework3/DigitRecognition.py', wdir='/MS_CS/Machine
Learning/Homework3')
/MS_CS/Machine Learning/Homework3/DigitRecognition.py:41:
DataConversionWarning: A column-vector y was passed when a 1d
array was expected. Please change the shape of y to (n_samples,
), for example using ravel().
  classifier.fit(trainFeaturesMatrix, trainMatrix)
0.977740678909
```

8) Neighbors : 10
Accuracy : ~10%

```
In [8]: runfile('/MS_CS/Machine
Learning/Homework3/DigitRecognition.py', wdir='/MS_CS/Machine
Learning/Homework3')
/MS_CS/Machine Learning/Homework3/DigitRecognition.py:41:
DataConversionWarning: A column-vector y was passed when a 1d
array was expected. Please change the shape of y to (n_samples,
), for example using ravel().
  classifier.fit(trainFeaturesMatrix, trainMatrix)
0.979410127991
```

As this data set was very well cleaned, the results were quite similar. I got accuracy of 97-98% almost all number of neighbors. The highest accuracy was achieved when number of neighbors were 8.

Part 2: Amazon Data Set - KNN:

Overview:

For this dataset as well used KNeighborsClassifier. Analysis was done based on number of neighbors. I realized one thing while running the algorithm over a large dataset that KNN

classifier takes way too longer to execute. For the sake of analysis this algorithm was trained over a full data set but tested over a limited data set. Following are the observations:

Neighbors : 2
Accuracy ~55%

```
Building test Data..
Training NN...
(147, 3)
Predict Data
Before printing accuracy
0.551020408163
```

Neighbors : 3
Accuracy ~60%

```
In [3]: print ("Training NN...")
   ...: classifier = KNeighborsClassifier(n_neighbors=3)
   ...: classifier.fit(train_data_features, train["rating"][:145035])
   ...: print (test.shape)
   ...: from sklearn.metrics import accuracy_score
   ...: #print(accuracy_score(output["sentiment"],test["rating"][:36208]))
   ...:
   ...: print("Predict Data")
   ...: #Prediction
   ...: prediction=classifier.predict(test_data_features)
   ...: print("Before printing accuracy")
   ...: #print (accuracy_score(prediction, test["rating"][:36208]))
   ...: print (accuracy_score(prediction, test["rating"]))
   ...:
Training NN...
(147, 3)
Predict Data
Before printing accuracy
0.598639455782
```

Neighbors : 5
Accuracy ~55%

```
In [4]: print ("Training NN...")
   ...: classifier = KNeighborsClassifier(n_neighbors=5)
   ...: classifier.fit(train_data_features, train["rating"][:145035])
   ...: print (test.shape)
   ...: from sklearn.metrics import accuracy_score
   ...: #print(accuracy_score(output["sentiment"],test["rating"][:36208]))
   ...:
   ...: print("Predict Data")
   ...: #Prediction
   ...: prediction=classifier.predict(test_data_features)
   ...: print("Before printing accuracy")
   ...: #print (accuracy_score(prediction, test["rating"][:36208]))
   ...: print (accuracy_score(prediction, test["rating"]))
   ...:
Training NN...
(147, 3)
Predict Data
Before printing accuracy
0.551020408163
```

Neighbors : 10

Accuracy ~58%

```
In [5]: print ("Training NN...")
   ...: classifier = KNeighborsClassifier(n_neighbors=10)
   ...: classifier.fit(train_data_features, train["rating"][:145035])
   ...: print (test.shape)
   ...: from sklearn.metrics import accuracy_score
   ...: #print(accuracy_score(output["sentiment"],test["rating"][:36208]))
   ...:
   ...: print("Predict Data")
   ...: #Prediction
   ...: prediction=classifier.predict(test_data_features)
   ...: print("Before printing accuracy")
   ...: #print (accuracy_score(prediction, test["rating"][:36208]))
   ...: print (accuracy_score(prediction, test["rating"]))
   ...:
Training NN...
(147, 3)
Predict Data
Before printing accuracy
0.578231292517
```

Neighbors : 20

Accuracy ~58%

```
In [6]: print ("Training NN...")
   ...: classifier = KNeighborsClassifier(n_neighbors=20)
   ...: classifier.fit(train_data_features, train["rating"][:145035])
   ...: print (test.shape)
   ...: from sklearn.metrics import accuracy_score
   ...: #print(accuracy_score(output["sentiment"],test["rating"][:36208]))
   ...:
   ...: print("Predict Data")
   ...: #Prediction
   ...: prediction=classifier.predict(test_data_features)
   ...: print("Before printing accuracy")
   ...: #print (accuracy_score(prediction, test["rating"][:36208]))
   ...: print (accuracy_score(prediction, test["rating"]))
   ...:
Training NN...
(147, 3)
Predict Data
Before printing accuracy
0.585034013605
```

Neighbors : 50
Accuracy ~58%

```
In [7]: print ("Training NN...")
   ...: classifier = KNeighborsClassifier(n_neighbors=50)
   ...: classifier.fit(train_data_features, train["rating"][:145035])
   ...: print (test.shape)
   ...: from sklearn.metrics import accuracy_score
   ...: #print(accuracy_score(output["sentiment"],test["rating"][:36208]))
   ...:
   ...: print("Predict Data")
   ...: #Prediction
   ...: prediction=classifier.predict(test_data_features)
   ...: print("Before printing accuracy")
   ...: #print (accuracy_score(prediction, test["rating"][:36208]))
   ...: print (accuracy_score(prediction, test["rating"]))
   ...:
   ...:
Training NN...
(147, 3)
Predict Data
Before printing accuracy
0.578231292517
```

Neighbors : 100
Accuracy ~58%

```
In [8]: print ("Training NN...")
   ...: classifier = KNeighborsClassifier(n_neighbors=100)
   ...: classifier.fit(train_data_features, train["rating"][:145035])
   ...: print (test.shape)
   ...: from sklearn.metrics import accuracy_score
   ...: #print(accuracy_score(output["sentiment"],test["rating"][:36208]))
   ...:
   ...: print("Predict Data")
   ...: #Prediction
   ...: prediction=classifier.predict(test_data_features)
   ...: print("Before printing accuracy")
   ...: #print (accuracy_score(prediction, test["rating"][:36208]))
   ...: print (accuracy_score(prediction, test["rating"]))
   ...:
Training NN...
(147, 3)
Predict Data
Before printing accuracy
0.578231292517
```

For amazon data set increasing number of neighbors didnt had much effect on accuracy. When neighbors are 3 thats when I got the maximum accuracy. Otherwise it was always around 57-58%.

Part 3 : Digit Recognition - Boosting

Overview:

I have used AdaBoostClassifier with Decision Tree for this part of assignment. Analysis is done based on max depth for decision tree, number of estimators and learning rate. Following are observations:

1. Max_Depth : 100 estimators : 100 learning_rate : 0.05
   Accuracy with Decision Tree : 85.95%
   Accuracy with Decision Tree + Boosting : 86%

```
In [10]: model = DecisionTreeClassifier(max_depth=100)
    ...: #model.fit(df,trainLabels)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print(prediction)
    ...:
    ...: #Accuracy calculation
    ...: import numpy as np
    ...: from sklearn.metrics import accuracy_score
    ...: print("Accuracy Before Boosting..")
    ...: print(accuracy_score(prediction,Z))
    ...: print("After Boosting...")
    ...: model =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=100),n_estimators=100,lear
ning_rate=0.05)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print("Accuracy After Boosting..")
    ...: print(accuracy_score(prediction,Z))
[0 1 4 ..., 1 9 8]
Accuracy Before Boosting..
0.857540345019
After Boosting...
Accuracy After Boosting..
0.859766277129
```

2. Max_Depth : 64 estimators : 100 learning_rate : 0.05
   Accuracy with Decision Tree : 86%
   Accuracy with Decision Tree + Boosting: 85.5%

```
In [11]: model = DecisionTreeClassifier(max_depth=64)
    ...: #model.fit(df,trainLabels)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print(prediction)
    ...:
    ...: #Accuracy calculation
    ...: import numpy as np
    ...: from sklearn.metrics import accuracy_score
    ...: print("Accuracy Before Boosting..")
    ...: print(accuracy_score(prediction,Z))
    ...: print("After Boosting...")
    ...: model =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=64),n_estimators=100,learn
ing_rate=0.05)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print("Accuracy After Boosting..")
    ...: print(accuracy_score(prediction,Z))
[0 1 4 ..., 1 9 8]
Accuracy Before Boosting..
0.864218141347
After Boosting...
Accuracy After Boosting..
0.85531441291
```

2. Max_Depth : 32 estimators : 100 learning_rate : 0.05
   Accuracy with Decision Tree : 85.5%
   Accuracy with Decision Tree + Boosting : 85%

```
In [12]: model = DecisionTreeClassifier(max_depth=32)
    ...: #model.fit(df,trainLabels)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print(prediction)
    ...:
    ...: #Accuracy calculation
    ...: import numpy as np
    ...: from sklearn.metrics import accuracy_score
    ...: print("Accuracy Before Boosting..")
    ...: print(accuracy_score(prediction,Z))
    ...: print("After Boosting...")
    ...: model =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=32),n_estimators=100,learn
ing_rate=0.05)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print("Accuracy After Boosting..")
    ...: print(accuracy_score(prediction,Z))
[0 1 4 ..., 1 9 8]
Accuracy Before Boosting..
0.854201446856
After Boosting...
Accuracy After Boosting..
0.847523650529
```

3. Max_Depth : 16 estimators : 100 learning_rate : 0.05
   Accuracy with Decision Tree : 86%
   Accuracy with Decision Tree + Boosting: 85.5%

```
In [13]: model = DecisionTreeClassifier(max_depth=16)
    ...: #model.fit(df,trainLabels)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print(prediction)
    ...:
    ...: #Accuracy calculation
    ...: import numpy as np
    ...: from sklearn.metrics import accuracy_score
    ...: print("Accuracy Before Boosting..")
    ...: print(accuracy_score(prediction,Z))
    ...: print("After Boosting...")
    ...: model =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=16),n_estimators=100,learn
ing_rate=0.05)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print("Accuracy After Boosting..")
    ...: print(accuracy_score(prediction,Z))
[0 1 4 ..., 1 9 8]
Accuracy Before Boosting..
0.863661658319
After Boosting...
Accuracy After Boosting..
0.854201446856
```

4. Max_Depth : 8estimators : 100 learning_rate : 0.05
   Accuracy with Decision Tree : 83%
   Accuracy with Decision Tree + Boosting : 83.5%

```
In [14]: model = DecisionTreeClassifier(max_depth=8)
    ...: #model.fit(df,trainLabels)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print(prediction)
    ...:
    ...: #Accuracy calculation
    ...: import numpy as np
    ...: from sklearn.metrics import accuracy_score
    ...: print("Accuracy Before Boosting..")
    ...: print(accuracy_score(prediction,Z))
    ...: print("After Boosting...")
    ...: model =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=8),n_estimators=100,learni
ng_rate=0.05)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print("Accuracy After Boosting..")
    ...: print(accuracy_score(prediction,Z))
[0 1 4 ..., 1 9 8]
Accuracy Before Boosting..
0.836950473011
After Boosting...
Accuracy After Boosting..
0.936004451864
```

5. Max_Depth : 6estimators : 100 learning_rate : 0.05
   Accuracy with Decision Tree : 75.23%
   Accuracy with Decision Tree + Boosting : 93%

```
In [15]: model = DecisionTreeClassifier(max_depth=6)
    ...: #model.fit(df,trainLabels)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print(prediction)
    ...:
    ...: #Accuracy calculation
    ...: import numpy as np
    ...: from sklearn.metrics import accuracy_score
    ...: print("Accuracy Before Boosting..")
    ...: print(accuracy_score(prediction,Z))
    ...: print("After Boosting...")
    ...: model =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=6),n_estimators=100,learni
ng_rate=0.05)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print("Accuracy After Boosting..")
    ...: print(accuracy_score(prediction,Z))
[0 1 1 ..., 1 9 8]
Accuracy Before Boosting..
0.752365052866
After Boosting...
Accuracy After Boosting..
0.929883138564
```

6. Max_Depth : 4estimators : 100 learning_rate : 0.05
   Accuracy with Decision Tree : 53%
   Accuracy with Decision Tree + Boosting : 90%

```
In [16]: model = DecisionTreeClassifier(max_depth=4)
    ...: #model.fit(df,trainLabels)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print(prediction)
    ...:
    ...: #Accuracy calculation
    ...: import numpy as np
    ...: from sklearn.metrics import accuracy_score
    ...: print("Accuracy Before Boosting..")
    ...: print(accuracy_score(prediction,Z))
    ...: print("After Boosting...")
    ...: model =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=4),n_estimators=100,learni
ng_rate=0.05)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print("Accuracy After Boosting..")
    ...: print(accuracy_score(prediction,Z))
[0 5 8 ..., 8 5 8]
Accuracy Before Boosting..
0.531441291041
After Boosting...
Accuracy After Boosting..
0.893155258765
```

7. Max_Depth : 2estimators : 100 learning_rate : 0.05
   Accuracy with Decision Tree : 30%
   Accuracy with Decision Tree + Boosting : 78%

```
In [17]: model = DecisionTreeClassifier(max_depth=2)
    ...: #model.fit(df,trainLabels)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print(prediction)
    ...:
    ...: #Accuracy calculation
    ...: import numpy as np
    ...: from sklearn.metrics import accuracy_score
    ...: print("Accuracy Before Boosting..")
    ...: print(accuracy_score(prediction,Z))
    ...: print("After Boosting...")
    ...: model =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=2),n_estimators=100,learni
ng_rate=0.05)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print("Accuracy After Boosting..")
    ...: print(accuracy_score(prediction,Z))
[0 1 1 ..., 1 1 1]
Accuracy Before Boosting..
0.298831385643
After Boosting...
Accuracy After Boosting..
0.780189204229
```

8. Max_Depth : 12 estimators : 100 learning_rate : 0.05
   Accuracy with Decision Tree : 85.5%
   Accuracy with Decision Tree + Boosting : 85.5%

```
In [18]: model = DecisionTreeClassifier(max_depth=12)
    ...: #model.fit(df,trainLabels)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print(prediction)
    ...:
    ...: #Accuracy calculation
    ...: import numpy as np
    ...: from sklearn.metrics import accuracy_score
    ...: print("Accuracy Before Boosting..")
    ...: print(accuracy_score(prediction,Z))
    ...: print("After Boosting...")
    ...: model =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=12),n_estimators=100,learn
ing_rate=0.05)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print("Accuracy After Boosting..")
    ...: print(accuracy_score(prediction,Z))
[0 1 4 ..., 8 9 8]
Accuracy Before Boosting..
0.85531441291
After Boosting...
Accuracy After Boosting..
0.855870895938
```

9. Max_Depth : 12 estimators : 100 learning_rate : 0.01
   Accuracy with Decision Tree : 85%
   Accuracy with Decision Tree + Boosting : 88%

```
In [19]: model = DecisionTreeClassifier(max_depth=12)
    ...: #model.fit(df,trainLabels)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print(prediction)
    ...:
    ...: #Accuracy calculation
    ...: import numpy as np
    ...: from sklearn.metrics import accuracy_score
    ...: print("Accuracy Before Boosting..")
    ...: print(accuracy_score(prediction,Z))
    ...: print("After Boosting...")
    ...: model =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=12),n_estimators=100,learn
ing_rate=0.01)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print("Accuracy After Boosting..")
    ...: print(accuracy_score(prediction,Z))
[0 1 4 ..., 1 9 8]
Accuracy Before Boosting..
0.847523650529
After Boosting...
Accuracy After Boosting..
0.881469115192
```

10. Max_Depth : 12 estimators : 100 learning_rate : 0.001
    Accuracy with Decision Tree : 85.5%
    Accuracy with Decision Tree + Boosting : 89%

```
In [20]: model = DecisionTreeClassifier(max_depth=12)
    ...: #model.fit(df,trainLabels)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print(prediction)
    ...:
    ...: #Accuracy calculation
    ...: import numpy as np
    ...: from sklearn.metrics import accuracy_score
    ...: print("Accuracy Before Boosting..")
    ...: print(accuracy_score(prediction,Z))
    ...: print("After Boosting...")
    ...: model =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=12),n_estimators=100,learn
ing_rate=0.001)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print("Accuracy After Boosting..")
    ...: print(accuracy_score(prediction,Z))
[0 1 4 ..., 8 9 8]
Accuracy Before Boosting..
0.854757929883
After Boosting...
Accuracy After Boosting..
0.88592097941
```

11. Max_Depth : 16 estimators : 100 learning_rate : 0.001
    Accuracy with Decision Tree : 86%
    Accuracy with Decision Tree + Boosting : 86%

```
In [24]: model = DecisionTreeClassifier(max_depth=16)
    ...: #model.fit(df,trainLabels)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print(prediction)
    ...:
    ...: #Accuracy calculation
    ...: import numpy as np
    ...: from sklearn.metrics import accuracy_score
    ...: print("Accuracy Before Boosting..")
    ...: print(accuracy_score(prediction,Z))
    ...: print("After Boosting...")
    ...: model =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=16),n_estimators=100,learn
ing_rate=0.001)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print("Accuracy After Boosting..")
    ...: print(accuracy_score(prediction,Z))
[0 1 4 ..., 8 9 8]
Accuracy Before Boosting..
0.862548692265
After Boosting...
Accuracy After Boosting..
0.857540345019
```

12. Max_Depth : 16 estimators : 300 learning_rate : 0.001
    Accuracy with Decision Tree : 86%
    Accuracy with Decision Tree + Boosting : 85%

```
In [26]: model = DecisionTreeClassifier(max_depth=16)
    ...: #model.fit(df,trainLabels)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print(prediction)
    ...:
    ...: #Accuracy calculation
    ...: import numpy as np
    ...: from sklearn.metrics import accuracy_score
    ...: print("Accuracy Before Boosting..")
    ...: print(accuracy_score(prediction,Z))
    ...: print("After Boosting...")
    ...: model =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=16),n_estimators=300,learn
ing_rate=0.001)
    ...: model.fit(trainData3,Y)
    ...: prediction=model.predict(testData3)
    ...: print("Accuracy After Boosting..")
    ...: print(accuracy_score(prediction,Z))
[0 1 4 ..., 1 9 8]
Accuracy Before Boosting..
0.857540345019
After Boosting...
Accuracy After Boosting..
0.85141903172
```

As you can see from results, for lesser depth of decision tree, boosting really improves accuracy. This can be seen when Max depth is 2, accuracy is improved from 30% to ~80%. Which is a real improvement when you think of efficiency with time saving approach.

Part 4 : Amazon Data Set - Boosting

Overview:

I have used AdaBoostClassifier with Decision Tree for this part of assignment. Analysis is done based on max depth for decision tree, number of estimators and learning rate. Following are observations:

1. Max_Depth : 16 estimators : 2 learning_rate : 0.01
   Accuracy with Decision Tree : 60%
   Accuracy with Decision Tree + Boosting : 60%

```
Accuracy Before Boosting..
0.598293194874
Building AdaBoost Model..
Accuracy After Boosting..
0.598541758727
```

2. Max_Depth : 8 estimators : 2 learning_rate : 0.01

Accuracy with Decision Tree : 60%

Accuracy with Decision Tree + Boosting : 60%

```
In [3]: model=DecisionTreeClassifier(max_depth=8)
   ...: model=model.fit( train_data_features, train["rating"][:145035] )
   ...: result = model.predict(test_data_features)
   ...: output = pd.DataFrame( data={"id":test["name"][:36208],
"sentiment":result[:36208]} )
   ...: print("Accuracy Before Boosting..")
   ...: print(accuracy_score(output["sentiment"],test["rating"][:36208]))
   ...: print("Building AdaBoost Model..")
   ...: model =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=8),n_estimators=2,learning
_rate=0.01)
   ...: model=model.fit( train_data_features, train["rating"][:145035] )
   ...: result = model.predict(test_data_features)
   ...: output = pd.DataFrame( data={"id":test["name"][:36208],
"sentiment":result[:36208]} )
   ...: print("Accuracy After Boosting..")
   ...: print(accuracy_score(output["sentiment"],test["rating"][:36208]))
   ...:
Accuracy Before Boosting..
0.598237958462
Building AdaBoost Model..
Accuracy After Boosting..
0.597989394609
```

3. Max_Depth : 6 estimators : 2 learning_rate : 0.001

Accuracy with Decision Tree : 59%

Accuracy with Decision Tree + Boosting : 59.5%

```
In [4]: model=DecisionTreeClassifier(max_depth=6)
   ...: model=model.fit( train_data_features, train["rating"][:145035] )
   ...: result = model.predict(test_data_features)
   ...: output = pd.DataFrame( data={"id":test["name"][:36208],
"sentiment":result[:36208]} )
   ...: print("Accuracy Before Boosting..")
   ...: print(accuracy_score(output["sentiment"],test["rating"][:36208]))
   ...: print("Building AdaBoost Model..")
   ...: model =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=6),n_estimators=2,learning
_rate=0.01)
   ...: model=model.fit( train_data_features, train["rating"][:145035] )
   ...: result = model.predict(test_data_features)
   ...: output = pd.DataFrame( data={"id":test["name"][:36208],
"sentiment":result[:36208]} )
   ...: print("Accuracy After Boosting..")
   ...: print(accuracy_score(output["sentiment"],test["rating"][:36208]))
   ...:
Accuracy Before Boosting..
0.593542863456
Building AdaBoost Model..
Accuracy After Boosting..
0.596304684048
```

4. Max_Depth : 4 estimators : 2 learning_rate : 0.01

Accuracy with Decision Tree : 59%

Accuracy with Decision Tree + Boosting : 59%

```
In [5]: model=DecisionTreeClassifier(max_depth=4)
   ...: model=model.fit( train_data_features, train["rating"][:145035] )
   ...: result = model.predict(test_data_features)
   ...: output = pd.DataFrame( data={"id":test["name"][:36208],
"sentiment":result[:36208]} )
   ...: print("Accuracy Before Boosting..")
   ...: print(accuracy_score(output["sentiment"],test["rating"][:36208]))
   ...: print("Building AdaBoost Model..")
   ...: model =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=4),n_estimators=2,learning
_rate=0.01)
   ...: model=model.fit( train_data_features, train["rating"][:145035] )
   ...: result = model.predict(test_data_features)
   ...: output = pd.DataFrame( data={"id":test["name"][:36208],
"sentiment":result[:36208]} )
   ...: print("Accuracy After Boosting..")
   ...: print(accuracy_score(output["sentiment"],test["rating"][:36208]))
   ...:
Accuracy Before Boosting..
0.590422006186
Building AdaBoost Model..
Accuracy After Boosting..
0.590422006186
```

5. Max_Depth : 4 estimators : 3 learning_rate : 0.01

Accuracy with Decision Tree : 59%

Accuracy with Decision Tree + Boosting : 59%

```
Accuracy Before Boosting..
0.590422006186
Building AdaBoost Model..
Accuracy After Boosting..
0.590422006186
```

6. Max_Depth : 4 estimators : 4 learning_rate : 0.01

Accuracy with Decision Tree : 59%

Accuracy with Decision Tree + Boosting : 59%

```
In [3]: print("Building AdaBoost Model..")
   ...: model =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=4),n_estimators=4,learning
_rate=0.01)
   ...: model=model.fit( train_data_features, train["rating"][:145035] )
   ...: result = model.predict(test_data_features)
   ...: output = pd.DataFrame( data={"id":test["name"][:36208],
"sentiment":result[:36208]} )
   ...: print("Accuracy After Boosting..")
   ...: print(accuracy_score(output["sentiment"],test["rating"][:36208]))
   ...:
Building AdaBoost Model..
Accuracy After Boosting..
0.590422006186
```

7. Max_Depth : 4 estimators : 4 learning_rate : 0.001

Accuracy with Decision Tree : 59%

Accuracy with Decision Tree + Boosting : 59%

```
In [4]: print("Building AdaBoost Model..")
   ...: model =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=4),n_estimators=4,learning
_rate=0.001)
   ...: model=model.fit( train_data_features, train["rating"][:145035] )
   ...: result = model.predict(test_data_features)
   ...: output = pd.DataFrame( data={"id":test["name"][:36208],
"sentiment":result[:36208]} )
   ...: print("Accuracy After Boosting..")
   ...: print(accuracy_score(output["sentiment"],test["rating"][:36208]))
   ...:
Building AdaBoost Model..
Accuracy After Boosting..
0.590422006186
```

8. Max_Depth : 6 estimators : 5 learning_rate : 0.001

Accuracy with Decision Tree : 59.5%

Accuracy with Decision Tree + Boosting : 59.5%

```
In [6]: model=DecisionTreeClassifier(max_depth=6)
   ...: model=model.fit( train_data_features, train["rating"][:145035] )
   ...: result = model.predict(test_data_features)
   ...: output = pd.DataFrame( data={"id":test["name"][:36208],
"sentiment":result[:36208]} )
   ...: print("Accuracy Before Boosting..")
   ...: print(accuracy_score(output["sentiment"],test["rating"][:36208]))
   ...: print("Building AdaBoost Model..")
   ...: model =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=6),n_estimators=5,learning
_rate=0.001)
   ...: model=model.fit( train_data_features, train["rating"][:145035] )
   ...: result = model.predict(test_data_features)
   ...: output = pd.DataFrame( data={"id":test["name"][:36208],
"sentiment":result[:36208]} )
   ...: print("Accuracy After Boosting..")
   ...: print(accuracy_score(output["sentiment"],test["rating"][:36208]))
   ...:
Accuracy Before Boosting..
0.593542863456
Building AdaBoost Model..
Accuracy After Boosting..
0.593542863456
```

9. Max_Depth : 10 estimators : 5 learning_rate : 0.001

Accuracy with Decision Tree : 60%

Accuracy with Decision Tree + Boosting : 60%

```
In [7]: model=DecisionTreeClassifier(max_depth=10)
   ...: model=model.fit( train_data_features, train["rating"][:145035] )
   ...: result = model.predict(test_data_features)
   ...: output = pd.DataFrame( data={"id":test["name"][:36208],
"sentiment":result[:36208]} )
   ...: print("Accuracy Before Boosting..")
   ...: print(accuracy_score(output["sentiment"],test["rating"][:36208]))
   ...: print("Building AdaBoost Model..")
   ...: model =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=10),n_estimators=5,learnin
g_rate=0.001)
   ...: model=model.fit( train_data_features, train["rating"][:145035] )
   ...: result = model.predict(test_data_features)
   ...: output = pd.DataFrame( data={"id":test["name"][:36208],
"sentiment":result[:36208]} )
   ...: print("Accuracy After Boosting..")
   ...: print(accuracy_score(output["sentiment"],test["rating"][:36208]))
   ...:
Accuracy Before Boosting..
0.603181617322
Building AdaBoost Model..
Accuracy After Boosting..
0.602491162174
```

10. Max_Depth : 12 estimators : 5 learning_rate : 0.001

    Accuracy with Decision Tree : 60.5%

    Accuracy with Decision Tree + Boosting : 60.5%

```
In [8]: model=DecisionTreeClassifier(max_depth=12)
   ...: model=model.fit( train_data_features, train["rating"][:145035] )
   ...: result = model.predict(test_data_features)
   ...: output = pd.DataFrame( data={"id":test["name"][:36208],
"sentiment":result[:36208]} )
   ...: print("Accuracy Before Boosting..")
   ...: print(accuracy_score(output["sentiment"],test["rating"][:36208]))
   ...: print("Building AdaBoost Model..")
   ...: model =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=12),n_estimators=5,learnin
g_rate=0.001)
   ...: model=model.fit( train_data_features, train["rating"][:145035] )
   ...: result = model.predict(test_data_features)
   ...: output = pd.DataFrame( data={"id":test["name"][:36208],
"sentiment":result[:36208]} )
   ...: print("Accuracy After Boosting..")
   ...: print(accuracy_score(output["sentiment"],test["rating"][:36208]))
   ...:
Accuracy Before Boosting..
0.60597105612
Building AdaBoost Model..
Accuracy After Boosting..
0.605639637649
```

11. Max_Depth : 2 estimators : 5 learning_rate : 0.05

Accuracy with Decision Tree : 58%

Accuracy with Decision Tree + Boosting : 58%

```
In [9]: model=DecisionTreeClassifier(max_depth=2)
   ...: model=model.fit( train_data_features, train["rating"][:145035] )
   ...: result = model.predict(test_data_features)
   ...: output = pd.DataFrame( data={"id":test["name"][:36208],
"sentiment":result[:36208]} )
   ...: print("Accuracy Before Boosting..")
   ...: print(accuracy_score(output["sentiment"],test["rating"][:36208]))
   ...: print("Building AdaBoost Model..")
   ...: model =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=2),n_estimators=5,learning
_rate=0.05)
   ...: model=model.fit( train_data_features, train["rating"][:145035] )
   ...: result = model.predict(test_data_features)
   ...: output = pd.DataFrame( data={"id":test["name"][:36208],
"sentiment":result[:36208]} )
   ...: print("Accuracy After Boosting..")
   ...: print(accuracy_score(output["sentiment"],test["rating"][:36208]))
   ...:
Accuracy Before Boosting..
0.58227463544
Building AdaBoost Model..
Accuracy After Boosting..
0.58227463544
```

As you can see here after experimenting with lots of combinations here, Boosting wasn't really useful for this data set. Results after and before boosting were pretty much the same all the time. Might have seen difference in case of large value of max depth and estimators, but the execution was taking way too long.