Results:

Part 1 : Digit Recognition -

Overview:

Analysis is done based on varying the maximum depth of the tree. Pruning leads to change in the nature of accuracy. After certain threshold point for the maximum depth of the tree, the change in the correct predictions do not vary much. There is an upper and lower bound for the maximum depth of the tree. Following are the few outputs based on pruning:

1) Maximum depth : 9
   Accuracy : ~85%

```
In [96]: runfile('/MS_CS/Machine Learning/PythonCodes/DigitRecognization.py', wdir='/MS_CS/Machine Learning/PythonCodes')
Model is -
Prediction is -
[0 1 4 ..., 1 9 8]
0.851975514747

In [97]: runfile('/MS_CS/Machine Learning/PythonCodes/DigitRecognization.py', wdir='/MS_CS/Machine Learning/PythonCodes')
Model is -
Prediction is -
[0 1 4 ..., 1 9 8]
0.848080133556
```

2) Maximum depth : 7
   Accuracy : ~79.74%

```
In [99]: runfile('/MS_CS/Machine Learning/PythonCodes/DigitRecognization.py', wdir='/MS_CS/Machine Learning/PythonCodes')
Model is -
Prediction is -
[0 1 1 ..., 1 9 8]
0.797440178075
```

3) Maximum depth : 10
   Accuracy : ~85.364%

```
In [100]: runfile('/MS_CS/Machine Learning/PythonCodes/DigitRecognization.py', wdir='/MS_CS/Machine Learning/PythonCodes')
Model is -
Prediction is -
[0 1 4 ..., 1 9 8]
0.853644963829
```

4) Maximum depth : 12
Accuracy : ~85.977%

```
In [101]: runfile('/MS_CS/Machine Learning/PythonCodes/DigitRecognization.py', wdir='/MS_CS/Machine Learning/PythonCodes')
Model is -
Prediction is -
[0 1 4 ..., 8 9 8]
0.859766277129
```

5) Maximum depth : 20
Accuracy : ~85.810%

```
In [103]: runfile('/MS_CS/Machine Learning/PythonCodes/DigitRecognization.py', wdir='/MS_CS/Machine Learning/PythonCodes')
Model is -
Prediction is -
[0 1 4 ..., 8 9 8]
0.858096828047
```

Part 2: Amazon Data Set:

Overview:

Analysis was done based on the two factors first was pruning and no pruning and then with and without stop words in english. After specifying depth of tree 25 and below, the accuracy was pretty much the same. But excluding stopwords and pruning helped improve accuracy from 54% to 60%

ForestTreeClassifier:
Maximum depth - 100
Stopwords - included
Accuracy ~54%

```
In [73]: from sklearn.metrics import accuracy_score
    ...: print(accuracy_score(output["sentiment"],test["rating"][:36208]))
0.546702386213
```

Decision Tree Classifiers:
Without pruning:
Maximum depth - NA
Stopwords - included
Accuracy ~54%

```
In [73]: from sklearn.metrics import accuracy_score
    ...: print(accuracy_score(output["sentiment"],test["rating"][:36208]))
0.546702386213
```

Maximum depth - 100
Stopwords - excluded
Accuracy ~57%

```
In [110]: output = pd.DataFrame( data={"id":test["name"][:36208], "sentiment":result[:36208]} )
     ...:
     ...: from sklearn.metrics import accuracy_score
     ...: print(accuracy_score(output["sentiment"],test["rating"][:36208]))
0.568603623509
```

Maximum depth - 50
Stopwords - excluded
Accuracy ~59%

```
In [114]: test_data_features = vectorizer.transform(clean_test_reviews)
     ...: test_data_features = test_data_features.toarray()
     ...:
     ...: # Use the random forest to make sentiment label predictions
     ...: #result = forest.predict(test_data_features)
     ...: result = model.predict(test_data_features)
     ...: # Copy the results to a pandas dataframe with an "id" column and
     ...: # a "sentiment" column
     ...: output = pd.DataFrame( data={"id":test["name"][:36208], "sentiment":result[:36208]} )
     ...:
     ...: from sklearn.metrics import accuracy_score
     ...: print(accuracy_score(output["sentiment"],test["rating"][:36208]))
0.589980114892
```

Maximum depth - 25
Stopwords - excluded
Accuracy ~60.50%

```
In [117]: test_data_features = vectorizer.transform(clean_test_reviews)
     ...: test_data_features = test_data_features.toarray()
     ...:
     ...: # Use the random forest to make sentiment label predictions
     ...: #result = forest.predict(test_data_features)
     ...: result = model.predict(test_data_features)
     ...: # Copy the results to a pandas dataframe with an "id" column and
     ...: # a "sentiment" column
     ...: output = pd.DataFrame( data={"id":test["name"][:36208], "sentiment":result[:36208]} )
     ...:
     ...: from sklearn.metrics import accuracy_score
     ...: print(accuracy_score(output["sentiment"],test["rating"][:36208]))
0.604148254529
```

Maximum depth - 100
Stopwords - excluded
Accuracy ~60.50%

```
In [121]: test_data_features = vectorizer.transform(clean_test_reviews)
     ...: test_data_features = test_data_features.toarray()
     ...:
     ...: # Use the random forest to make sentiment label predictions
     ...: #result = forest.predict(test_data_features)
     ...: result = model.predict(test_data_features)
     ...: # Copy the results to a pandas dataframe with an "id" column and
     ...: # a "sentiment" column
     ...: output = pd.DataFrame( data={"id":test["name"][:36208], "sentiment":result[:36208]} )
     ...:
     ...: from sklearn.metrics import accuracy_score
     ...: print(accuracy_score(output["sentiment"],test["rating"][:36208]))
0.603043526293
```