Results:

Part 1 : Digit Recognition -

Overview:

I have used MLPClassifier for this part of assignment. Analysis is done based on varying values for solver, learning_rate_init, number of hidden layers and alpha values. Used sgm and adam as solver for this assignment. For this dataset, the difference wasn't significant. By varying value for alpha, did got some variations. But most significant difference was observed by changing value of learning rate init. Following are the few outputs based on variables:

1) solver : 'sgd' alpha : 0.001 hidden_layers : 200 learning_rate : 0.01
   Accuracy : ~97%

```
In [1]: runfile('/MS_CS/Machine Learning/Homework 2/DigitRecog_NN.py',
wdir='/MS_CS/Machine Learning/Homework 2')
/MS_CS/Softwares/anaconda/lib/python3.5/site-
packages/sklearn/neural_network/multilayer_perceptron.py:904:
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().
  y = column_or_1d(y, warn=True)
0.967723984418
```

2) solver : 'sgd' alpha : 0.001 hidden_layers : 100 learning_rate : 0.01
   Accuracy : ~96%

```
In [2]: classifier =
MLPClassifier(solver='sgd',alpha=0.001,hidden_layer_sizes=(100),random_state
=1,learning_rate_init=0.01)
   ...: classifier.fit(trainFeaturesMatrix, trainMatrix)
   ...:
   ...: #Prediction
   ...: prediction=classifier.predict(testFeaturesMatrix)
   ...:
   ...: print (accuracy_score(prediction, testMatrix))
/MS_CS/Softwares/anaconda/lib/python3.5/site-
packages/sklearn/neural_network/multilayer_perceptron.py:904:
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().
  y = column_or_1d(y, warn=True)
0.9571508069
```

3) solver : ‘sgd’ alpha : 0.001 hidden_layers : 64 learning_rate : 0.01

Accuracy : ~96%

```
In [3]: classifier =
MLPClassifier(solver='sgd',alpha=0.001,hidden_layer_sizes=(64),random_state=
1,learning_rate_init=0.01)
   ...: classifier.fit(trainFeaturesMatrix, trainMatrix)
   ...:
   ...: #Prediction
   ...: prediction=classifier.predict(testFeaturesMatrix)
   ...:
   ...: print (accuracy_score(prediction, testMatrix))
/MS_CS/Softwares/anaconda/lib/python3.5/site-
packages/sklearn/neural_network/multilayer_perceptron.py:904:
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().
  y = column_or_1d(y, warn=True)
0.962159154146
```

4) solver : ‘sgd’ alpha : 0.001 hidden_layers : 32 learning_rate : 0.01

Accuracy : ~92%

```
In [4]: classifier =
MLPClassifier(solver='sgd',alpha=0.001,hidden_layer_sizes=(32),random_state=
1,learning_rate_init=0.01)
   ...: classifier.fit(trainFeaturesMatrix, trainMatrix)
   ...:
   ...: #Prediction
   ...: prediction=classifier.predict(testFeaturesMatrix)
   ...:
   ...: print (accuracy_score(prediction, testMatrix))
/MS_CS/Softwares/anaconda/lib/python3.5/site-
packages/sklearn/neural_network/multilayer_perceptron.py:904:
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().
  y = column_or_1d(y, warn=True)
0.920979410128
```

5) solver : ‘sgd’ alpha : 0.001 hidden_layers : 16 learning_rate : 0.01

Accuracy : ~93%

```
In [5]: classifier =
MLPClassifier(solver='sgd',alpha=0.001,hidden_layer_sizes=(16),random_state=
1,learning_rate_init=0.01)
   ...: classifier.fit(trainFeaturesMatrix, trainMatrix)
   ...:
   ...: #Prediction
   ...: prediction=classifier.predict(testFeaturesMatrix)
   ...:
   ...: print (accuracy_score(prediction, testMatrix))
/MS_CS/Softwares/anaconda/lib/python3.5/site-
packages/sklearn/neural_network/multilayer_perceptron.py:904:
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().
  y = column_or_1d(y, warn=True)
0.928213689482
```

6) solver : 'sgd' alpha : 0.001 hidden_layers : 1 learning_rate : 0.01
Accuracy : ~10%

```
In [6]: classifier =
MLPClassifier(solver='sgd',alpha=0.001,hidden_layer_sizes=(1),random_state=1
,learning_rate_init=0.01)
   ...: classifier.fit(trainFeaturesMatrix, trainMatrix)
   ...:
   ...: #Prediction
   ...: prediction=classifier.predict(testFeaturesMatrix)
   ...:
   ...: print (accuracy_score(prediction, testMatrix))
0.100723427935
```

7) solver : 'sgd' alpha : 0.001 hidden_layers : 64 learning_rate : 0.1
Accuracy : ~10%

```
In [7]: classifier =
MLPClassifier(solver='sgd',alpha=0.001,hidden_layer_sizes=(64),random_state=
1,learning_rate_init=0.1)
   ...: classifier.fit(trainFeaturesMatrix, trainMatrix)
   ...:
   ...: #Prediction
   ...: prediction=classifier.predict(testFeaturesMatrix)
   ...:
   ...: print (accuracy_score(prediction, testMatrix))
0.10183639399
```

8) solver : 'sgd' alpha : 0.001 hidden_layers : 64 learning_rate : 0.5
Accuracy : ~10%

```
In [8]: classifier =
MLPClassifier(solver='sgd',alpha=0.001,hidden_layer_sizes=(64),random_state=
1,learning_rate_init=0.5)
   ...: classifier.fit(trainFeaturesMatrix, trainMatrix)
   ...:
   ...: #Prediction
   ...: prediction=classifier.predict(testFeaturesMatrix)
   ...:
   ...: print (accuracy_score(prediction, testMatrix))
0.101279910963
```

9) solver : 'sgd' alpha : 0.001 hidden_layers : 64 learning_rate : 0.05
Accuracy : ~20%

```
In [9]: classifier =
MLPClassifier(solver='sgd',alpha=0.001,hidden_layer_sizes=(64),random_state=
1,learning_rate_init=0.05)
   ...: classifier.fit(trainFeaturesMatrix, trainMatrix)
   ...:
   ...: #Prediction
   ...: prediction=classifier.predict(testFeaturesMatrix)
   ...:
   ...: print (accuracy_score(prediction, testMatrix))
0.191986644407
```

10) solver : 'sgd' alpha : 0.001 hidden_layers : 64 learning_rate : 0.02
Accuracy : ~93%

```
In [10]: classifier =
MLPClassifier(solver='sgd',alpha=0.001,hidden_layer_sizes=(64),random_state=
1,learning_rate_init=0.02)
    ...: classifier.fit(trainFeaturesMatrix, trainMatrix)
    ...:
    ...: #Prediction
    ...: prediction=classifier.predict(testFeaturesMatrix)
    ...:
    ...: print (accuracy_score(prediction, testMatrix))
/MS_CS/Softwares/anaconda/lib/python3.5/site-
packages/sklearn/neural_network/multilayer_perceptron.py:904:
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().
  y = column_or_1d(y, warn=True)
0.928213689482
```

11) solver : 'sgd' alpha : 0.001 hidden_layers : 64 learning_rate : 0.002
Accuracy : ~96%

```
In [11]: classifier =
MLPClassifier(solver='sgd',alpha=0.001,hidden_layer_sizes=(64),random_state=
1,learning_rate_init=0.002)
    ...: classifier.fit(trainFeaturesMatrix, trainMatrix)
    ...:
    ...: #Prediction
    ...: prediction=classifier.predict(testFeaturesMatrix)
    ...:
    ...: print (accuracy_score(prediction, testMatrix))
/MS_CS/Softwares/anaconda/lib/python3.5/site-
packages/sklearn/neural_network/multilayer_perceptron.py:904:
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().
  y = column_or_1d(y, warn=True)
0.958820255982
```

12) solver : 'sgd' alpha : 0.01 hidden_layers : 64 learning_rate : 0.01
Accuracy : ~96%

```
In [12]: classifier =
MLPClassifier(solver='sgd',alpha=0.01,hidden_layer_sizes=(64),random_state=1
,learning_rate_init=0.01)
    ...: classifier.fit(trainFeaturesMatrix, trainMatrix)
    ...:
    ...: #Prediction
    ...: prediction=classifier.predict(testFeaturesMatrix)
    ...:
    ...: print (accuracy_score(prediction, testMatrix))
/MS_CS/Softwares/anaconda/lib/python3.5/site-
packages/sklearn/neural_network/multilayer_perceptron.py:904:
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().
  y = column_or_1d(y, warn=True)
0.961602671119
```

13) solver : 'sgd' alpha : 0.1 hidden_layers : 64 learning_rate : 0.01
Accuracy : ~96.5%

```
In [13]: classifier =
MLPClassifier(solver='sgd',alpha=0.1,hidden_layer_sizes=(64),random_state=1,
learning_rate_init=0.01)
    ...: classifier.fit(trainFeaturesMatrix, trainMatrix)
    ...:
    ...: #Prediction
    ...: prediction=classifier.predict(testFeaturesMatrix)
    ...:
    ...: print (accuracy_score(prediction, testMatrix))
/MS_CS/Softwares/anaconda/lib/python3.5/site-
packages/sklearn/neural_network/multilayer_perceptron.py:904:
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().
  y = column_or_1d(y, warn=True)
0.965498052309
```

14) solver : 'sgd' alpha : 0.2 hidden_layers : 64 learning_rate : 0.01
Accuracy : ~96%

```
In [14]: classifier =
MLPClassifier(solver='sgd',alpha=0.2,hidden_layer_sizes=(64),random_state=1,
learning_rate_init=0.01)
    ...: classifier.fit(trainFeaturesMatrix, trainMatrix)
    ...:
    ...: #Prediction
    ...: prediction=classifier.predict(testFeaturesMatrix)
    ...:
    ...: print (accuracy_score(prediction, testMatrix))
/MS_CS/Softwares/anaconda/lib/python3.5/site-
packages/sklearn/neural_network/multilayer_perceptron.py:904:
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().
  y = column_or_1d(y, warn=True)
0.961046188091
```

15) solver : 'sgd' alpha : 0.5 hidden_layers : 64 learning_rate : 0.01
Accuracy : ~96.6%

```
In [15]: classifier =
MLPClassifier(solver='sgd',alpha=0.5,hidden_layer_sizes=(64),random_state=1,
learning_rate_init=0.01)
    ...: classifier.fit(trainFeaturesMatrix, trainMatrix)
    ...:
    ...: #Prediction
    ...: prediction=classifier.predict(testFeaturesMatrix)
    ...:
    ...: print (accuracy_score(prediction, testMatrix))
/MS_CS/Softwares/anaconda/lib/python3.5/site-
packages/sklearn/neural_network/multilayer_perceptron.py:904:
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().
  y = column_or_1d(y, warn=True)
0.966054535337
```

16) solver : 'sgd' alpha : 1.0 hidden_layers : 64 learning_rate : 0.01
Accuracy : ~96%

```
In [17]: classifier =
MLPClassifier(solver='sgd',alpha=1.00,hidden_layer_sizes=(64),random_state=1
,learning_rate_init=0.01)
    ...: classifier.fit(trainFeaturesMatrix, trainMatrix)
    ...:
    ...: #Prediction
    ...: prediction=classifier.predict(testFeaturesMatrix)
    ...:
    ...: print (accuracy_score(prediction, testMatrix))
/MS_CS/Softwares/anaconda/lib/python3.5/site-
packages/sklearn/neural_network/multilayer_perceptron.py:904:
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().
  y = column_or_1d(y, warn=True)
0.961602671119
```

17) solver : 'adam' alpha : 1.0 hidden_layers : 64 learning_rate : 0.01
Accuracy : ~95.5%

```
In [18]: classifier =
MLPClassifier(solver='adam',alpha=1.00,hidden_layer_sizes=(64),random_state=
1,learning_rate_init=0.01)
    ...: classifier.fit(trainFeaturesMatrix, trainMatrix)
    ...:
    ...: #Prediction
    ...: prediction=classifier.predict(testFeaturesMatrix)
    ...:
    ...: print (accuracy_score(prediction, testMatrix))
/MS_CS/Softwares/anaconda/lib/python3.5/site-
packages/sklearn/neural_network/multilayer_perceptron.py:904:
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().
  y = column_or_1d(y, warn=True)
0.955481357819
```

18) solver : 'adam' alpha : 0.01 hidden_layers : 64 learning_rate : 0.01
Accuracy : ~96.5%

```
In [19]: classifier =
MLPClassifier(solver='adam',alpha=0.01,hidden_layer_sizes=(64),random_state=
1,learning_rate_init=0.01)
    ...: classifier.fit(trainFeaturesMatrix, trainMatrix)
    ...:
    ...: #Prediction
    ...: prediction=classifier.predict(testFeaturesMatrix)
    ...:
    ...: print (accuracy_score(prediction, testMatrix))
/MS_CS/Softwares/anaconda/lib/python3.5/site-
packages/sklearn/neural_network/multilayer_perceptron.py:904:
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().
  y = column_or_1d(y, warn=True)
0.9632721202
```

Part 2: Amazon Data Set:

Overview:

For this dataset as well used MLPClassifier. Analysis was done based various factors. If we have higher number of hidden layers it takes way too much time to run. By decreasing number of nodes in each hidden layer, accuracy was improved to ~70% which is a significant gain compared with Decision tree classifier where maximum accuracy that I got was 60%.

MLPClassifier:

Solver : 'sgd'

Number of nodes in hidden layer - 200
max_iterations:200
Accuracy ~64%

```
Review 36000 of 36456

0.645906981882
```

MLPClassifiers:
Solver : 'sgd'
Number of nodes in hidden layer - 2
Max_iterations: 5
Accuracy ~67%

```
In [24]: Out[23]:
MLPClassifier(activation='relu', alpha=0.001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=2, learning_rate='constant',
       learning_rate_init=0.01, max_iter=5, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True,
       solver='sgd', tol=0.0001, validation_fraction=0.1, verbose=False,
       warm_start=False)prediction=classifier.predict(test_data_features)
    ...:
    ...: print (accuracy_score(prediction, test["rating"][:36208]))
0.669133893062
```

MLPClassifiers:
Solver : 'sgd'
Number of nodes in hidden layer - 5
Max_iterations: 5
Accuracy ~68%

```
In [26]: Out[25]:
MLPClassifier(activation='relu', alpha=0.001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=5, learning_rate='constant',
       learning_rate_init=0.01, max_iter=5, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True,
       solver='sgd', tol=0.0001, validation_fraction=0.1, verbose=False,
       warm_start=False)prediction=classifier.predict(test_data_features)
    ...:
    ...: print (accuracy_score(prediction, test["rating"][:36208]))
    ...:
0.676977463544
```

MLPClassifiers:
Solver : 'sgd'
Number of nodes in hidden layer - 10
Max_iterations: 5
Accuracy ~68%

```
In [28]: Out[27]:
MLPClassifier(activation='relu', alpha=0.001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=10, learning_rate='constant',
       learning_rate_init=0.01, max_iter=5, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True,
       solver='sgd', tol=0.0001, validation_fraction=0.1, verbose=False,
       warm_start=False)prediction=classifier.predict(test_data_features)
    ...:
    ...: print (accuracy_score(prediction, test["rating"][:36208]))
0.678662174105
```

MLPClassifiers:
Solver : 'sgd'
Number of nodes in hidden layer - 2
Max_iterations: 10
Accuracy ~67%

```
In [30]: Out[29]:
MLPClassifier(activation='relu', alpha=0.001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=2, learning_rate='constant',
       learning_rate_init=0.01, max_iter=10, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True,
       solver='sgd', tol=0.0001, validation_fraction=0.1, verbose=False,
       warm_start=False)prediction=classifier.predict(test_data_features)
    ...:
    ...: print (accuracy_score(prediction, test["rating"][:36208]))
0.669907202828
```

MLPClassifiers:
Solver : 'lbfgs'
Number of nodes in hidden layer - 2
Max_iterations: 5
Accuracy ~61%

```
Out[31]:
MLPClassifier(activation='relu', alpha=0.001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=2, learning_rate='constant',
       learning_rate_init=0.01, max_iter=10, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True,
       solver='lbfgs', tol=0.0001, validation_fraction=0.1, verbose=False,
       warm_start=False)

In [32]: prediction=classifier.predict(test_data_features)
    ...:
    ...: print (accuracy_score(prediction, test["rating"][:36208]))
0.606136765356
```

MLPClassifiers:
Solver : 'adam'
Number of nodes in hidden layer - 2
Max_iterations: 5

Accuracy ~66%

```
In [34]: Out[33]:
MLPClassifier(activation='relu', alpha=0.001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=2, learning_rate='constant',
       learning_rate_init=0.01, max_iter=5, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True,
       solver='adam', tol=0.0001, validation_fraction=0.1, verbose=False,
       warm_start=False)prediction=classifier.predict(test_data_features)
    ...:
    ...: print (accuracy_score(prediction, test["rating"][:36208]))
0.666289217852
```

MLPClassifiers:
Solver : 'adam'
Number of nodes in hidden layer - 10
Max_iterations: 5
Accuracy ~67%

```
In [36]Out[35]: :
MLPClassifier(activation='relu', alpha=0.001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=10, learning_rate='constant',
       learning_rate_init=0.01, max_iter=5, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True,
       solver='adam', tol=0.0001, validation_fraction=0.1, verbose=False,
       warm_start=False)
0.671011931065
```

MLPClassifiers:
Solver : 'adam'
Number of nodes in hidden layer - NA
Max_iterations: 200
Accuracy ~67%

```
Out[37]:
MLPClassifier(activation='relu', alpha=0.001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(), learning_rate='constant',
       learning_rate_init=0.01, max_iter=200, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True,
       solver='adam', tol=0.0001, validation_fraction=0.1, verbose=False,
       warm_start=False)

In [38]: prediction=classifier.predict(test_data_features)
    ...:
    ...: print (accuracy_score(prediction, test["rating"][:36208]))
    ...:
0.663361688025
```

MLPClassifiers:
Solver : 'adam'
Number of nodes in hidden layer - 2
Max_iterations: 200
Accuracy ~67%

```
Out[39]:
MLPClassifier(activation='relu', alpha=0.001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=2, learning_rate='constant',
       learning_rate_init=0.01, max_iter=200, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True,
       solver='adam', tol=0.0001, validation_fraction=0.1, verbose=False,
       warm_start=False)

In [40]: prediction=classifier.predict(test_data_features)
    ...:
    ...: print (accuracy_score(prediction, test["rating"][:36208]))
0.667200618648
```