Assignment 1 **Vidya Chockalingam, A53226608**

Question 1

Code:

```
timeStruct = [d['review/timeStruct'] for d in data]
def feature(datum):
  feat = [1] #the intercept vector does not have a feature and should be multiplied with 1
  feat.append(datum['year'])
  return feat
X = [feature(d) for d in timeStruct]
y = [d['review/overall'] for d in data]
theta,residuals,rank,s = numpy.linalg.lstsq(X, y)
print(theta)
print(residuals/len(y)) ##MSE
```

Answer:

```
[ -3.91707489e+01    2.14379786e-02]
[ 0.49004382]
```

Question 2:

A better representation for year would be giving binary codes for various years. For ex, year 1999 could be 0,0,0,1 and year 2012 can be 1,1,0,1. Any other year not provided in the dataset could be included in the bias.

Code

```
timeStruct = [d['review/timeStruct'] for d in data]
year = [d['year'] for d in timeStruct]
min(year)  ### Answer : 1999
max(year) ### 2012

def feature(datum):
  if datum['year'] == 1999:
    feat = [1,0,0,0,0]
  elif datum['year'] == 2000:
    feat = [1,0,0,0,1]
```

```
 elif datum['year'] == 2001:
  feat = [1,0,0,1,0]
 elif datum['year'] == 2002:
  feat = [1,0,0,1,1]
 elif datum['year'] == 2003:
  feat = [1,0,1,0,0]
 elif datum['year'] == 2004:
  feat = [1,0,1,0,1]
 elif datum['year'] == 2005:
  feat = [1,0,1,1,0]
 elif datum['year'] == 2006:
  feat = [1,0,1,1,1]
 elif datum['year'] == 2007:
  feat = [1,1,0,0,0]
 elif datum['year'] == 2008:
  feat = [1,1,0,0,1]
 elif datum['year'] == 2009:
  feat = [1,1,0,1,0]
 elif datum['year'] == 2010:
  feat = [1,1,0,1,1]
 elif datum['year'] == 2011:
  feat = [1,1,1,0,0]
 elif datum['year'] == 2012:
  feat = [1,1,1,0,1] # any other year will get added to the bias term
 return featX = [feature(d) for d in timeStruct]
y = [d['review/overall'] for d in data]
theta,residuals,rank,s = numpy.linalg.lstsq(X, y)

#### theta = array([  3.70613430e+00,   1.64047010e-01,   7.07739382e-02
,
####          4.47664682e-02,  -3.61124198e-03])
```

residuals/len(y) ##MSE

thetamat = numpy.asmatrix(theta)

Xmat = numpy.asmatrix(X)

(thetamat * Xmat.transpose()).shape

ymat = numpy.asmatrix(y)

sum(numpy.squeeze(numpy.asarray(ymat-thetamat*Xmat.transpose()))**2)/len(y) ###MSE

Answer

```
#### theta = array([  3.70613430e+00,   1.64047010e-01,   7.07739382e-02
,
####          4.47664682e-02,  -3.61124198e-03])


##### 0.48990915696982806
```

Question 3

Code

```
### Using pandas
df = pd.read_csv('C:/Users/BHEL/Desktop/Recommendation Systems/Assignment 1/winequality-whi
te.csv',delimiter=";")
df_train = df.iloc[0: int(len(df)/2)]
df_test = df.iloc[int(len(df)/2) + 1 :len(df)]

y = df_train["quality"].values
X = numpy.vstack([numpy.ones(len(df_train)),df_train["fixed acidity"].values,df_train["volatile acidit
y"],df_train["citric acid"] \
          ,df_train["residual sugar"],df_train["chlorides"],df_train["free sulfur dioxide"], \
          df_train["total sulfur dioxide"],df_train["density"],df_train["pH"], df_train["sulphates"], \
          df_train["alcohol"]]).T
theta,residuals,rank,s = numpy.linalg.lstsq(X, y)
theta

#### array([  2.56420279e+02,   1.35421303e-01,  -1.72994866e+00,
         1.02651152e-01,   1.09038568e-01,  -2.76775146e-01,
         6.34332168e-03,   3.85023977e-05,  -2.58652809e+02,
         1.19540566e+00,   8.33006285e-01,   9.79304353e-02])

y_test = df_test["quality"].values
X_test = numpy.vstack([numpy.ones(len(df_test)),df_test["fixed acidity"].values,df_test["volatile aci
dity"],df_test["citric acid"] \
          ,df_test["residual sugar"],df_test["chlorides"],df_test["free sulfur dioxide"], \
          df_test["total sulfur dioxide"],df_test["density"],df_test["pH"], df_test["sulphates"], \
          df_test["alcohol"]]).T
theta_test,residuals_test,rank_test,s_test = numpy.linalg.lstsq(X_test, y_test)

print(residuals/len(y))
print(residuals_test/len(y_test))
```

Answer

theta

```
#### array([  2.56420279e+02,   1.35421303e-01,  -1.72994866e+00,
          1.02651152e-01,   1.09038568e-01,  -2.76775146e-01,
          6.34332168e-03,   3.85023977e-05,  -2.58652809e+02,
          1.19540566e+00,   8.33006285e-01,   9.79304353e-02])
```

MSE of the train and test dataset

```
Train - [ 0.6023075]
Test - [ 0.49562901]
```

MSE of the test set is lower – this could be because the training and the test set features are not ide ntically distributed.

Question 4

In [6]:
```
# Removing Fixed Acidity
y = df_train["quality"].values
X = numpy.vstack([numpy.ones(len(df_train)),df_train["volatile acidity"],df_train["citric acid"] \
                  ,df_train["residual sugar"],df_train["chlorides"],df_train["free sulfur dioxide"], \
                 df_train["total sulfur dioxide"],df_train["density"],df_train["pH"], df_train["sulphates"], \
                 df_train["alcohol"]]).T
theta,residuals,rank,s = numpy.linalg.lstsq(X, y)
theta
```

Out[6]:
```
array([  1.59265497e+02,  -1.79843308e+00,   1.53362739e-01,
         7.40365514e-02,  -8.95644126e-01,   6.66762254e-03,
        -2.78710665e-04,  -1.59274221e+02,   7.11764816e-01,
         7.06744801e-01,   2.12169554e-01])
```

In [7]:
```
y_test = df_test["quality"].values
X_test = numpy.vstack([numpy.ones(len(df_test)),df_test["volatile acidity"],df_test["citric acid"] \
                  ,df_test["residual sugar"],df_test["chlorides"],df_test["free sulfur dioxide"], \
                 df_test["total sulfur dioxide"],df_test["density"],df_test["pH"], df_test["sulphates"], \
                 df_test["alcohol"]]).T
theta_test,residuals_test,rank_test,s_test = numpy.linalg.lstsq(X_test, y_test)
```

In [8]:
```
residuals/len(y)
residuals_test/len(y_test)
```

Out[8]:
```
array([ 0.49593076])
```

In [9]:
```
# Removing Volatile Acidity
y = df_train["quality"].values
X = numpy.vstack([numpy.ones(len(df_train)),df_train["fixed acidity"].values,df_train["citric acid"] \
                  ,df_train["residual sugar"],df_train["chlorides"],df_train["free sulfur dioxide"], \
                 df_train["total sulfur dioxide"],df_train["density"],df_train["pH"], df_train["sulphates"], \
                 df_train["alcohol"]]).T
theta,residuals,rank,s = numpy.linalg.lstsq(X, y)
theta
```

Out[9]:
```
array([  2.76319617e+02,   1.70596631e-01,   3.37705416e-01,
         1.11733261e-01,  -1.03169500e+00,   8.78146518e-03,
        -7.94469068e-04,  -2.79834990e+02,   1.46305736e+00,
         9.14939034e-01,   4.97740129e-02])
```

In [10]:
```
y_test = df_test["quality"].values
X_test = numpy.vstack([numpy.ones(len(df_test)),df_test["fixed acidity"].values,df_test["citric acid"] \
                  ,df_test["residual sugar"],df_test["chlorides"],df_test["free sulfur dioxide"], \
                 df_test["total sulfur dioxide"],df_test["density"],df_test["pH"], df_test["sulphates"], \
                 df_test["alcohol"]]).T
theta_test,residuals_test,rank_test,s_test = numpy.linalg.lstsq(X_test, y_test)
```

In [11]:
```
residuals/len(y)
residuals_test/len(y_test)
```

Out[11]:
```
array([ 0.52603379])
```

In [12]:
```
# Removing Citric Acid
y = df_train["quality"].values
X = numpy.vstack([numpy.ones(len(df_train)),df_train["fixed acidity"].values,df_train["volatile acidity"] \
                  ,df_train["residual sugar"],df_train["chlorides"],df_train["free sulfur dioxide"], \
                 df_train["total sulfur dioxide"],df_train["density"],df_train["pH"], df_train["sulphates"], \
                 df_train["alcohol"]]).T
theta,residuals,rank,s = numpy.linalg.lstsq(X, y)
theta
```

Out[12]:
```
array([  2.54710681e+02,   1.37865465e-01,  -1.75200556e+00,
         1.08761156e-01,  -2.16042405e-01,   6.39876960e-03,
         2.76634223e-05,  -2.56894507e+02,   1.18003096e+00,
         8.35323848e-01,   1.01084607e-01])
```

In [13]:
```
y_test = df_test["quality"].values
X_test = numpy.vstack([numpy.ones(len(df_test)),df_test["fixed acidity"].values,df_test["volatile acidity"] \
                  ,df_test["residual sugar"],df_test["chlorides"],df_test["free sulfur dioxide"], \
                 df_test["total sulfur dioxide"],df_test["density"],df_test["pH"], df_test["sulphates"], \
                 df_test["alcohol"]]).T
theta_test,residuals_test,rank_test,s_test = numpy.linalg.lstsq(X_test, y_test)
```

In [14]:
```
residuals/len(y)
residuals_test/len(y_test)
```

Out[14]:
```
array([ 0.49575561])
```

In [15]:
```
# Removing Residual Sugar
y = df_train["quality"].values
X = numpy.vstack([numpy.ones(len(df_train)),df_train["fixed acidity"].values,df_train["volatile acidity"],df_train["citric acid"]
                  ,df_train["chlorides"],df_train["free sulfur dioxide"], \
                 df_train["total sulfur dioxide"],df_train["density"],df_train["pH"], df_train["sulphates"], \
                 df_train["alcohol"]]).T
theta,residuals,rank,s = numpy.linalg.lstsq(X, y)
theta
```

Out[15]:
```
array([ -1.47920104e+01,  -7.46073461e-02,  -1.76142965e+00,
         6.81139124e-02,  -1.82547037e+00,   8.93883453e-03,
        -8.05570058e-04,   1.66496148e+01,   2.40618592e-01,
         4.00610149e-01,   3.87434345e-01])
```

In [16]:
```
y_test = df_test["quality"].values
X_test = numpy.vstack([numpy.ones(len(df_test)),df_test["fixed acidity"].values,df_test["volatile acidity"],df_test["citric acid"
                  ,df_test["chlorides"],df_test["free sulfur dioxide"], \
                 df_test["total sulfur dioxide"],df_test["density"],df_test["pH"], df_test["sulphates"], \
                 df_test["alcohol"]]).T
theta_test,residuals_test,rank_test,s_test = numpy.linalg.lstsq(X_test, y_test)
```

In [17]:
```
residuals/len(y)
residuals_test/len(y_test)
```

Out[17]:
```
array([ 0.50989045])
```

In [18]:
```
# Removing Chlorides
y = df_train["quality"].values
X = numpy.vstack([numpy.ones(len(df_train)),df_train["fixed acidity"].values,df_train["volatile acidity"],df_train["citric acid"]
                  ,df_train["residual sugar"],df_train["free sulfur dioxide"], \
                 df_train["total sulfur dioxide"],df_train["density"],df_train["pH"], df_train["sulphates"], \
                 df_train["alcohol"]]).T
theta,residuals,rank,s = numpy.linalg.lstsq(X, y)
theta
```

Out[18]:
```
array([  2.58709684e+02,   1.37904183e-01,  -1.73584539e+00,
         9.75957321e-02,   1.10074038e-01,   6.31755375e-03,
         3.99792536e-05,  -2.61016310e+02,   1.20593153e+00,
         8.35397592e-01,   9.72781925e-02])
```

In [19]:
```
y_test = df_test["quality"].values
X_test = numpy.vstack([numpy.ones(len(df_test)),df_test["fixed acidity"].values,df_test["volatile acidity"],df_test["citric acid"
                  ,df_test["residual sugar"],df_test["free sulfur dioxide"], \
                 df_test["total sulfur dioxide"],df_test["density"],df_test["pH"], df_test["sulphates"], \
                 df_test["alcohol"]]).T
theta_test,residuals_test,rank_test,s_test = numpy.linalg.lstsq(X_test, y_test)
```

In [20]:
```
residuals/len(y)
residuals_test/len(y_test)
```

Out[20]:
```
array([ 0.49563808])
```

In [21]:
```
# Removing Free Sulfur Dioxide
y = df_train["quality"].values
X = numpy.vstack([numpy.ones(len(df_train)),df_train["fixed acidity"].values,df_train["volatile acidity"],df_train["citric acid"]
                  ,df_train["residual sugar"],df_train["chlorides"], \
                 df_train["total sulfur dioxide"],df_train["density"],df_train["pH"], df_train["sulphates"], \
                 df_train["alcohol"]]).T
theta,residuals,rank,s = numpy.linalg.lstsq(X, y)
theta
```

Out[21]:
```
array([  2.82300292e+02,   1.44692601e-01,  -1.86565663e+00,
         1.35540475e-01,   1.21404594e-01,  -9.31572520e-02,
         1.55712233e-03,  -2.84834933e+02,   1.29264990e+00,
         8.11809727e-01,   7.21620256e-02])
```

In [22]:
```
y_test = df_test["quality"].values
X_test = numpy.vstack([numpy.ones(len(df_test)),df_test["fixed acidity"].values,df_test["volatile acidity"],df_test["citric acid"
                  ,df_test["residual sugar"],df_test["chlorides"], \
                 df_test["total sulfur dioxide"],df_test["density"],df_test["pH"], df_test["sulphates"], \
                 df_test["alcohol"]]).T
theta_test,residuals_test,rank_test,s_test = numpy.linalg.lstsq(X_test, y_test)
```

In [23]:
```
residuals/len(y)
residuals_test/len(y_test)
```

Out[23]:
```
array([ 0.49663839])
```

In [25]:
```
# Removing Total Sulfur Dioxide
y = df_train["quality"].values
X = numpy.vstack([numpy.ones(len(df_train)),df_train["fixed acidity"].values,df_train["volatile acidity"],df_train["citric acid"]
                  ,df_train["residual sugar"],df_train["chlorides"],df_train["free sulfur dioxide"], \
                 df_train["density"],df_train["pH"], df_train["sulphates"], \
                 df_train["alcohol"]]).T
theta,residuals,rank,s = numpy.linalg.lstsq(X, y)
theta
```

Out[25]:
```
array([  2.55873246e+02,   1.35090722e-01,  -1.72825856e+00,
         1.02416787e-01,   1.08891972e-01,  -2.77158772e-01,
         6.39868008e-03,  -2.58094731e+02,   1.19355492e+00,
         8.34256998e-01,   9.83042785e-02])
```

In [26]:
```
y_test = df_test["quality"].values
X_test = numpy.vstack([numpy.ones(len(df_test)),df_test["fixed acidity"].values,df_test["volatile acidity"],df_test["citric acid"
                  ,df_test["residual sugar"],df_test["chlorides"],df_test["free sulfur dioxide"], \
                 df_test["density"],df_test["pH"], df_test["sulphates"], \
                 df_test["alcohol"]]).T
theta_test,residuals_test,rank_test,s_test = numpy.linalg.lstsq(X_test, y_test)
```

In [27]:
```
residuals/len(y)
residuals_test/len(y_test)
```

Out[27]:
```
array([ 0.4957691])
```

In [28]:
```
# Removing Density
y = df_train["quality"].values
X = numpy.vstack([numpy.ones(len(df_train)),df_train["fixed acidity"].values,df_train["volatile acidity"],df_train["citric acid"]
                  ,df_train["residual sugar"],df_train["chlorides"],df_train["free sulfur dioxide"], \
                 df_train["total sulfur dioxide"],df_train["pH"], df_train["sulphates"], \
                 df_train["alcohol"]]).T
theta,residuals,rank,s = numpy.linalg.lstsq(X, y)
theta
```

Out[28]:
```
array([  1.35278847e+00,  -6.30214557e-02,  -1.81229926e+00,
         2.98038927e-02,   1.74234814e-02,  -1.45313766e+00,
         8.17207399e-03,  -1.03082385e-03,   3.24191788e-01,
         4.57723912e-01,   3.87103702e-01])
```

In [29]:
```
y_test = df_test["quality"].values
X_test = numpy.vstack([numpy.ones(len(df_test)),df_test["fixed acidity"].values,df_test["volatile acidity"],df_test["citric acid"
                  ,df_test["residual sugar"],df_test["chlorides"],df_test["free sulfur dioxide"], \
                 df_test["total sulfur dioxide"],df_test["pH"], df_test["sulphates"], \
                 df_test["alcohol"]]).T
theta_test,residuals_test,rank_test,s_test = numpy.linalg.lstsq(X_test, y_test)
```

In [30]:
```
residuals/len(y)
residuals_test/len(y_test)
```

Out[30]:
```
array([ 0.50117674])
```

In [31]:
```
# Removing pH
y = df_train["quality"].values
X = numpy.vstack([numpy.ones(len(df_train)),df_train["fixed acidity"].values,df_train["volatile acidity"],df_train["citric acid"]
                  ,df_train["residual sugar"],df_train["chlorides"],df_train["free sulfur dioxide"], \
                 df_train["total sulfur dioxide"],df_train["density"], df_train["sulphates"], \
                 df_train["alcohol"]]).T
theta,residuals,rank,s = numpy.linalg.lstsq(X, y)
theta
```

Out[31]:
```
array([  1.03156070e+02,  -3.95816093e-02,  -1.91850711e+00,
        -1.27755184e-02,   5.14623196e-02,  -1.22611859e+00,
         7.57413959e-03,  -6.04094775e-04,  -1.00780974e+02,
         7.57619203e-01,   2.83982725e-01])
```

In [32]:
```
y_test = df_test["quality"].values
X_test = numpy.vstack([numpy.ones(len(df_test)),df_test["fixed acidity"].values,df_test["volatile acidity"],df_test["citric acid"
                  ,df_test["residual sugar"],df_test["chlorides"],df_test["free sulfur dioxide"], \
                 df_test["total sulfur dioxide"],df_test["density"],df_test["sulphates"], \
                 df_test["alcohol"]]).T
theta_test,residuals_test,rank_test,s_test = numpy.linalg.lstsq(X_test, y_test)
```

In [33]:
```
residuals/len(y)
residuals_test/len(y_test)
```

Out[33]:
```
array([ 0.49564268])
```

In [34]:
```
# Removing Sulphates
y = df_train["quality"].values
X = numpy.vstack([numpy.ones(len(df_train)),df_train["fixed acidity"].values,df_train["volatile acidity"],df_train["citric acid"]
                  ,df_train["residual sugar"],df_train["chlorides"],df_train["free sulfur dioxide"], \
                 df_train["total sulfur dioxide"],df_train["density"],df_train["pH"],  \
                 df_train["alcohol"]]).T
theta,residuals,rank,s = numpy.linalg.lstsq(X, y)
theta
```

Out[34]:
```
array([  2.02754232e+02,   9.89499345e-02,  -1.77602652e+00,
         1.16540757e-01,   8.82235158e-02,  -4.48945940e-01,
         6.12915393e-03,   3.85168935e-04,  -2.04365676e+02,
         1.13522515e+00,   1.59797632e-01])
```

In [35]:
```
y_test = df_test["quality"].values
X_test = numpy.vstack([numpy.ones(len(df_test)),df_test["fixed acidity"].values,df_test["volatile acidity"],df_test["citric acid"
                  ,df_test["residual sugar"],df_test["chlorides"],df_test["free sulfur dioxide"], \
                 df_test["total sulfur dioxide"],df_test["density"],df_test["pH"], \
                 df_test["alcohol"]]).T
theta_test,residuals_test,rank_test,s_test = numpy.linalg.lstsq(X_test, y_test)
```

In [36]:
```
residuals/len(y)
residuals_test/len(y_test)
```

Out[36]:
```
array([ 0.50020054])
```

In [37]:
```
# Removing Alcohol
y = df_train["quality"].values
X = numpy.vstack([numpy.ones(len(df_train)),df_train["fixed acidity"].values,df_train["volatile acidity"],df_train["citric acid"]
                  ,df_train["residual sugar"],df_train["chlorides"],df_train["free sulfur dioxide"], \
                 df_train["total sulfur dioxide"],df_train["density"],df_train["pH"], df_train["sulphates"], \
                ]).T
theta,residuals,rank,s = numpy.linalg.lstsq(X, y)
theta
```

Out[37]:
```
array([  3.25993179e+02,   1.90940672e-01,  -1.68438300e+00,
         1.34455930e-01,   1.32486181e-01,  -1.97765161e-01,
         5.90527049e-03,   2.12840502e-04,  -3.29032181e+02,
         1.44529192e+00,   9.37096269e-01])
```

In [38]:
```
y_test = df_test["quality"].values
X_test = numpy.vstack([numpy.ones(len(df_test)),df_test["fixed acidity"].values,df_test["volatile acidity"],df_test["citric acid"
                  ,df_test["residual sugar"],df_test["chlorides"],df_test["free sulfur dioxide"], \
                 df_test["total sulfur dioxide"],df_test["density"],df_test["pH"], df_test["sulphates"], \
                ]).T
theta_test,residuals_test,rank_test,s_test = numpy.linalg.lstsq(X_test, y_test)
```

In [39]:
```
residuals/len(y)
residuals_test/len(y_test)
```

Out[39]:
```
array([ 0.50846051])
```

Answer

Based on the MSE values, removing **Volatile acidity** resulted in the highest increase in MSE of 0.5260. So Volatile acidity is an important variable.

Chlorides adds the least value to the MSE of 0.495638. So, it's not very important.

Question 5

Code

df = pd.read_csv('C:/Users/BHEL/Desktop/Recommendation Systems/Assignment 1/winequality-white.csv',delimiter=";")

```
df.loc[df['quality'] <= 5, 'quality'] = 0
df.loc[df['quality'] > 5, 'quality'] = 1
df_train = df.iloc[0: int(len(df)/2)]
df_test = df.iloc[int(len(df)/2) + 1 :len(df)]

X_train = numpy.vstack([numpy.ones(len(df_train)),df_train["volatile acidity"],df_train["citric acid"] \
          ,df_train["residual sugar"],df_train["chlorides"],df_train["free sulfur dioxide"], \
          df_train["total sulfur dioxide"],df_train["density"],df_train["pH"], df_train["sulphates"], \
          df_train["alcohol"]]).T
y_train = df_train["quality"].values

clf = svm.SVC(C=1000)
clf.fit(X_train, y_train)

y_test = df_test["quality"].values
X_test = numpy.vstack([numpy.ones(len(df_test)),df_test["volatile acidity"],df_test["citric acid"] \
          ,df_test["residual sugar"],df_test["chlorides"],df_test["free sulfur dioxide"], \
          df_test["total sulfur dioxide"],df_test["density"],df_test["pH"], df_test["sulphates"], \
          df_test["alcohol"]]).T

train_predictions = clf.predict(X_train)
test_predictions = clf.predict(X_test)

accuracy_train = sum(z[0] == z[1] for z in zip(train_predictions, y_train))/len(train_predictions)
accuracy_test = sum(z[0] == z[1] for z in zip(test_predictions, y_test))/len(test_predictions)
```

Answer

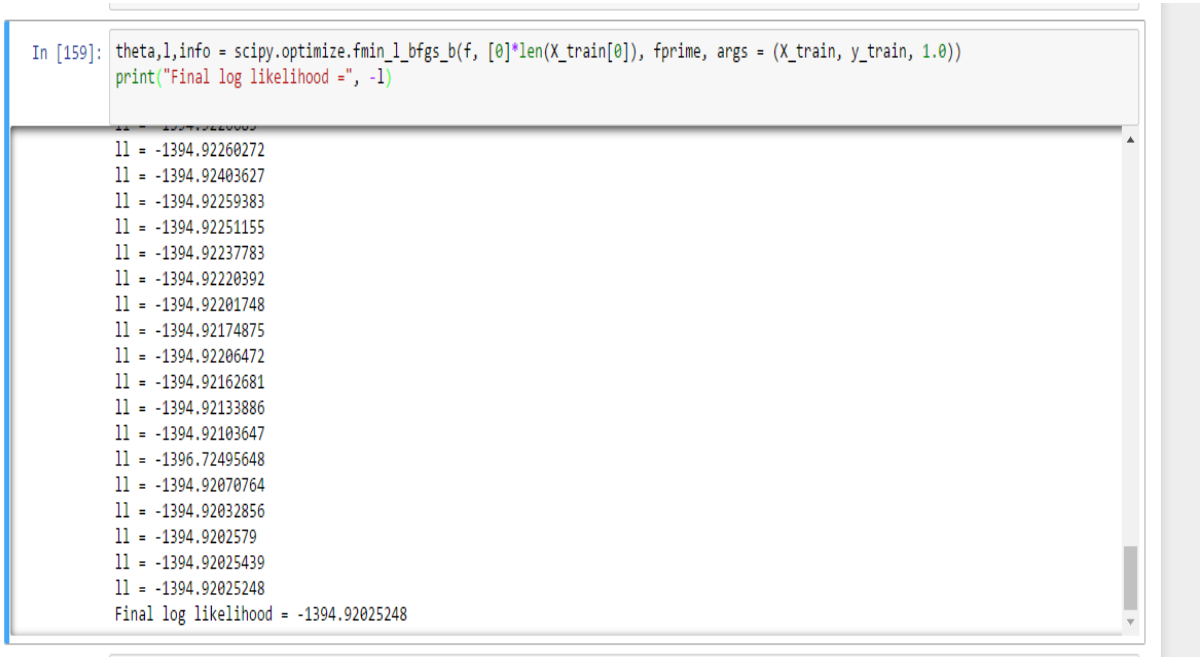
```
Accuracy train - 0.99959167006941607
Accuracy test - 0.64991830065359479
```

<u>Question 6</u>

Code

```
def fprime(theta, X, y, lam):
```

```
dl = [0.0]*len(theta)
for i in range(len(X)):
  logit = inner(X[i],theta)
  logitsigm = 1- sigmoid(logit)
  for k in range(len(theta)):
     dl[k] += X[i,k]*logitsigm
     if not y[i]:
        dl[k] -= X[i,k]
for m in range(len(theta)):
     dl[m] -= 2*lam*theta[m]
# Negate the return value since we're doing gradient *ascent*
return numpy.array([-x for x in dl])
```




```
theta = numpy.matrix(theta).transpose()

X_train = numpy.matrix(X_train)

y_pred_train = (X_train * theta)

for i in range(len(y_pred_train)):

   if(y_pred_train[i,0]<0):

      y_pred_train[i,0] = 0

   else:

      y_pred_train[i,0] = 1

accuracy_train = sum(z[0] == z[1] for z in zip(y_pred_train, y_train))/len(y_pred_train)

X_test = numpy.matrix(X_test)
```

```
y_pred_test = (X_test * theta)

for i in range(len(y_pred_test)):

   if(y_pred_test[i,0]<0):

      y_pred_test[i,0] = 0

   else:

      y_pred_test[i,0] = 1

accuracy_test = sum(z[0] == z[1] for z in zip(y_pred_test, y_test))/len(y_pred_test)
```

Answer

Final log – likelihood value converged at -1394.92

```
Training accuracy - matrix([[ 0.70600245]])
Testing accuracy - matrix([[ 0.76062092]])
```

Testing accuracy is greater than training accuracy which might be due to the fact that training and testing datasets are not identically distributed.