18D070067_A6

October 3, 2021

DS203 Assignment 6

Name: Vinit Awale

Roll No: 18D070067

```
[ ]: # Imports
     import numpy as np
     import matplotlib.pyplot as plt
     import pandas as pd
```

1 EXERCISE 1

1.1 Question 2

```
[ ]: df = pd.read_excel('Real estate valuation data set.xlsx')
```

```
[ ]: df.head()
```

```
[ ]:    No  X1 transaction date  X2 house age  \
     0   1           2012.916667          32.0
     1   2           2012.916667          19.5
     2   3           2013.583333          13.3
     3   4           2013.500000          13.3
     4   5           2012.833333           5.0

        X3 distance to the nearest MRT station  X4 number of convenience stores  \
     0                                84.87882                               10
     1                               306.59470                                9
     2                               561.98450                                5
     3                               561.98450                                5
     4                               390.56840                                5

        X5 latitude  X6 longitude  Y house price of unit area
     0     24.98298     121.54024                        37.9
     1     24.98034     121.53951                        42.2
     2     24.98746     121.54391                        47.3
     3     24.98746     121.54391                        54.8
     4     24.97937     121.54245                        43.1
```

[ ]:
```
# Remove the first column from df
df = df.iloc[:, 1:]
```

[ ]:
```
df.head()
```

[ ]:
```
   X1 transaction date  X2 house age  X3 distance to the nearest MRT station  \
0          2012.916667          32.0                                84.87882
1          2012.916667          19.5                               306.59470
2          2013.583333          13.3                               561.98450
3          2013.500000          13.3                               561.98450
4          2012.833333           5.0                               390.56840

   X4 number of convenience stores  X5 latitude  X6 longitude  \
0                               10     24.98298     121.54024
1                                9     24.98034     121.53951
2                                5     24.98746     121.54391
3                                5     24.98746     121.54391
4                                5     24.97937     121.54245

   Y house price of unit area
0                        37.9
1                        42.2
2                        47.3
3                        54.8
4                        43.1
```

1.2 Question 3

[ ]:
```
# Split the dataset into train and test
from sklearn.model_selection import train_test_split
df_train, df_test = train_test_split(df, test_size=0.2, random_state=0)
```

[ ]:
```
# Split the dataset into X and y
X_train = df_train.iloc[:, :-1].values
y_train = df_train.iloc[:, -1].values
X_test = df_test.iloc[:, :-1].values
y_test = df_test.iloc[:, -1].values
```

1.3 Question 4

[ ]:
```
# Train a linear regression model
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

[ ]:
```
LinearRegression()
```

1.4 Question 5

[ ]:

```
## Reporting the coefficients of the trained model
print(regressor.coef_)
```

```
[ 5.02223987e+00 -2.63131890e-01 -4.46354435e-03  1.09259467e+00
  2.25488372e+02 -6.81792744e+00]
```

1.5 Question 6

[ ]:

```
## Make predictions on the test set
y_pred = regressor.predict(X_test)
```

1.6 Question 7

[ ]:

```
# Compute the mean squared error and the R2 score
from sklearn.metrics import mean_squared_error, r2_score

print("Train_test_split: 80:20")
print("MSE :" ,mean_squared_error(y_test, y_pred))
print("R2 Score: ", r2_score(y_test, y_pred))
```

```
Train_test_split: 80:20
MSE : 52.7799807924661
R2 Score:  0.5921381263269844
```

1.7 Question 8

Repeating the above parts for the following train-test splits:
- train-test split = 0.4 - train-test split = 0.3 - train-test split = 0.1

1.7.1 train-test split = 0.4

[ ]:

```
df_train , df_test = train_test_split(df, test_size=0.4, random_state=0)

# Split the dataset into X and y
X_train = df_train.iloc[:, :-1].values
y_train = df_train.iloc[:, -1].values
X_test = df_test.iloc[:, :-1].values
y_test = df_test.iloc[:, -1].values

# Train a linear regression model
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Make predictions on the test set
y_pred = regressor.predict(X_test)

# Compute the mean squared error and the R2 score
```

```
print("Train_test_split: 70:30")
print("MSE :" ,mean_squared_error(y_test, y_pred))
print("R2 Score: ", r2_score(y_test, y_pred))
```

```
Train_test_split: 70:30
MSE : 67.73912214762815
R2 Score:  0.5963978418530073
```

1.7.2 train-test split = 0.3

[ ]:

```
df_train , df_test = train_test_split(df, test_size=0.3, random_state=0)

# Split the dataset into X and y
X_train = df_train.iloc[:, :-1].values
y_train = df_train.iloc[:, -1].values
X_test = df_test.iloc[:, :-1].values
y_test = df_test.iloc[:, -1].values

# Train a linear regression model
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Make predictions on the test set
y_pred = regressor.predict(X_test)

# Compute the mean squared error and the R2 score
print("Train_test_split: 60:40")
print("MSE :" ,mean_squared_error(y_test, y_pred))
print("R2 Score: ", r2_score(y_test, y_pred))
```

```
Train_test_split: 60:40
MSE : 71.57636713662353
R2 Score:  0.5800106026205316
```

1.7.3 train-test split = 0.1

[ ]:

```
df_train , df_test = train_test_split(df, test_size=0.1, random_state=0)

# Split the dataset into X and y
X_train = df_train.iloc[:, :-1].values
y_train = df_train.iloc[:, -1].values
X_test = df_test.iloc[:, :-1].values
y_test = df_test.iloc[:, -1].values

# Train a linear regression model
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

```python
# Make predictions on the test set
y_pred = regressor.predict(X_test)

# Compute the mean squared error and the R2 score
print("Train_test_split: 90:10")
print("MSE :" ,mean_squared_error(y_test, y_pred))
print("R2 Score: ", r2_score(y_test, y_pred))
```

```
Train_test_split: 90:10
MSE : 52.7799807924661
R2 Score:  0.5921381263269844
```

1.8 Question 9

1.9 Using Lasso and Ridge regression using the lambda values as 0.001, 0.005, 0.01, 0.05, 0.1, and 0.5

LASSO Regression

[ ]:

```python
# Using a training set of 80% and a test set of 20%
# Train a linear regression model with Lasso Regression

lambdas = [0.001, 0.005, 0.01, 0.05, 0.1, 0.5]


df_train , df_test = train_test_split(df, test_size=0.2, random_state=0)
X_train = df_train.iloc[:, :-1].values
y_train = df_train.iloc[:, -1].values
X_test = df_test.iloc[:, :-1].values
y_test = df_test.iloc[:, -1].values

# Import a Lasso Regression model
from sklearn.linear_model import Lasso

# Train a linear regression model for each lambdas
for lambda_ in lambdas:
    regressor = Lasso(alpha=lambda_)
    regressor.fit(X_train, y_train)
    y_pred = regressor.predict(X_test)
    print("Lambda: ", lambda_)
    print("MSE :" ,mean_squared_error(y_test, y_pred))
    print("R2 Score: ", r2_score(y_test, y_pred))
    print("\n")
```

```
Lambda:  0.001
MSE : 59.56611506860197
R2 Score:  0.6570728061861305


Lambda:  0.005
```

```
MSE : 59.74335383113915
R2 Score:  0.6560524275463298


Lambda:  0.01
MSE : 60.232208348465505
R2 Score:  0.6532380504862026


Lambda:  0.05
MSE : 63.134469187702535
R2 Score:  0.636529487838308


Lambda:  0.1
MSE : 63.224173619907354
R2 Score:  0.6360130517878246


Lambda:  0.5
MSE : 65.91449801213228
R2 Score:  0.6205246253021932
```

RIDGE Regression

[ ]:

```
# Using a training set of 80% and a test set of 20%
# Train a linear regression model with Lasso Regression

lambdas = [0.001, 0.005, 0.01, 0.05, 0.1, 0.5]


df_train , df_test = train_test_split(df, test_size=0.2, random_state=0)
X_train = df_train.iloc[:, :-1].values
y_train = df_train.iloc[:, -1].values
X_test = df_test.iloc[:, :-1].values
y_test = df_test.iloc[:, -1].values

# Import a ridge regression model
from sklearn.linear_model import Ridge

# Train a linear regression model for each lambdas
for lambda_ in lambdas:
    regressor = Ridge(alpha=lambda_)
    regressor.fit(X_train, y_train)
    y_pred = regressor.predict(X_test)
    print("Lambda: ", lambda_)
    print("MSE :" ,mean_squared_error(y_test, y_pred))
    print("R2 Score: ", r2_score(y_test, y_pred))
    print("\n")
```

```
Lambda:  0.001
MSE : 59.52097175120972
R2 Score:  0.6573327001062719


Lambda:  0.005
MSE : 59.57847897006466
R2 Score:  0.6570016261531821


Lambda:  0.01
MSE : 59.70242510464357
R2 Score:  0.656288057707997


Lambda:  0.05
MSE : 60.76543675414683
R2 Score:  0.6501682091743839


Lambda:  0.1
MSE : 61.48968561782382
R2 Score:  0.6459986468291212


Lambda:  0.5
MSE : 62.651579037130254
R2 Score:  0.6393095275314353
```

1.9.1 Repeating the same for all the train- test splits mentioned earlier

```
[ ]: split_ = [0.4, 0.3, 0.1]
     lambdas = [0.001, 0.005, 0.01, 0.05, 0.1, 0.5]
     from sklearn.model_selection import train_test_split

     for i in range(len(split_)):
         df_train , df_test = train_test_split(df, test_size=float(split_[i]),␣
      ↪random_state=0)
         X_train = df_train.iloc[:, :-1].values
         y_train = df_train.iloc[:, -1].values
         X_test = df_test.iloc[:, :-1].values
```

```
    y_test = df_test.iloc[:, -1].values
    print("Train_test_split: ", split_[i])

    print("RIDGE REGRESSION")

    for lambda_ in lambdas:
        regressor = Ridge(alpha=lambda_)
        regressor.fit(X_train, y_train)
        y_pred = regressor.predict(X_test)
        print("Lambda: ", lambda_)
        print("MSE :" ,mean_squared_error(y_test, y_pred))
        print("R2 Score: ", r2_score(y_test, y_pred))
        print("\n")

    print("LASSO REGRESSION")

    for lambda_ in lambdas:
        regressor = Lasso(alpha=lambda_)
        regressor.fit(X_train, y_train)
        y_pred = regressor.predict(X_test)
        print("Lambda: ", lambda_)
        print("MSE :" ,mean_squared_error(y_test, y_pred))
        print("R2 Score: ", r2_score(y_test, y_pred))
        print("\n")
```

```
Train_test_split:  0.4
RIDGE REGRESSION
Lambda:  0.001
MSE : 67.68519729642419
R2 Score:  0.5967191360421502


Lambda:  0.005
MSE : 67.58330024841091
R2 Score:  0.5973262574098783


Lambda:  0.01
MSE : 67.60254382430406
R2 Score:  0.5972116006426409


Lambda:  0.05
MSE : 68.5102217560316
R2 Score:  0.591803488453805


Lambda:  0.1
```

```
MSE : 69.1776868834366
R2 Score:  0.5878266083678567


Lambda:  0.5
MSE : 70.17696837935294
R2 Score:  0.5818727052825876


LASSO REGRESSION
Lambda:  0.001
MSE : 67.70201607015997
R2 Score:  0.5966189266924873


Lambda:  0.005
MSE : 67.60269617519755
R2 Score:  0.597210692907975


Lambda:  0.01
MSE : 67.7480547340377
R2 Score:  0.5963446198590041


Lambda:  0.05
MSE : 70.51352506958072
R2 Score:  0.5798674385739535


Lambda:  0.1
MSE : 70.49383140937229
R2 Score:  0.579984777026386


Lambda:  0.5
MSE : 72.57954574973373
R2 Score:  0.5675577070798132


Train_test_split:  0.3
RIDGE REGRESSION
Lambda:  0.001
MSE : 71.56798006421099
R2 Score:  0.5800598155329668


Lambda:  0.005
```

```
MSE : 71.60266438572339
R2 Score:  0.5798562979771973


Lambda:  0.01
MSE : 71.73124656545355
R2 Score:  0.5791018149775846


Lambda:  0.05
MSE : 72.94442259295877
R2 Score:  0.571983249324


Lambda:  0.1
MSE : 73.7396491207691
R2 Score:  0.5673170903171072


Lambda:  0.5
MSE : 74.94498771259576
R2 Score:  0.5602445124667503


LASSO REGRESSION
Lambda:  0.001
MSE : 71.58812829489112
R2 Score:  0.5799415915492675


Lambda:  0.005
MSE : 71.70217411767905
R2 Score:  0.5792724036831814


Lambda:  0.01
MSE : 72.15616699625191
R2 Score:  0.5766085049256149


Lambda:  0.05
MSE : 75.40099211324512
R2 Score:  0.557568810680076


Lambda:  0.1
MSE : 75.44636734505886
R2 Score:  0.5573025619582175
```

```
Lambda:  0.5
MSE : 77.9396336339094
R2 Score:  0.5426727973019293


Train_test_split:  0.1
RIDGE REGRESSION
Lambda:  0.001
MSE : 52.8189978180053
R2 Score:  0.5918366188822557


Lambda:  0.005
MSE : 52.99977418638194
R2 Score:  0.5904396538357579


Lambda:  0.01
MSE : 53.250304679880564
R2 Score:  0.5885036577448848


Lambda:  0.05
MSE : 54.854257844909505
R2 Score:  0.5761089707186657


Lambda:  0.1
MSE : 55.85541148496585
R2 Score:  0.5683724692395629


Lambda:  0.5
MSE : 57.42752341816405
R2 Score:  0.5562238380905826


LASSO REGRESSION
Lambda:  0.001
MSE : 52.85549359460964
R2 Score:  0.5915545946070759


Lambda:  0.005
MSE : 53.36913418862332
R2 Score:  0.5875853924223937
```

```
Lambda:  0.01
MSE : 54.29555172133913
R2 Score:  0.5804264206868273


Lambda:  0.05
MSE : 57.84274360016901
R2 Score:  0.5530151881654284


Lambda:  0.1
MSE : 57.66184509835855
R2 Score:  0.5544130970086159


Lambda:  0.5
MSE : 57.96134074407502
R2 Score:  0.552098718462342
```

2 EXERCISE 2

2.1 Question 2

```
[ ]: df = pd.read_csv('haberman.data', sep=',', header=None)
```

```
[ ]: df.head()
```

```
[ ]:     0   1  2  3
     0  30  64  1  1
     1  30  62  3  1
     2  30  65  0  1
     3  31  59  2  1
     4  31  65  4  1
```

2.2 Question 3

```
[ ]: # split the dataset in train and test
     from sklearn.model_selection import train_test_split

     df_train , df_test = train_test_split(df, test_size=0.2, random_state=0)
```

2.3 Question 4

[ ]:
```
X_train = df_train.iloc[:, :-1].values
y_train = df_train.iloc[:, -1].values
X_test = df_test.iloc[:, :-1].values
y_test = df_test.iloc[:, -1].values
```

[ ]:
```
# Train a Logistic Regression model
from sklearn.linear_model import LogisticRegression

Regressor = LogisticRegression()
Regressor.fit(X_train, y_train)
```

[ ]:
```
LogisticRegression()
```

2.4 Question 5

[ ]:
```
## Report the coeffiecients
print(Regressor.coef_)
```

```
[[0.02318509 0.00266963 0.09251148]]
```

2.5 Question 6

[ ]:
```
# Make predictions on the test set
y_pred = Regressor.predict(X_test)
```

[ ]:
```
y_pred
```

[ ]:
```
array([1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)
```

2.6 Question 7

2.7 Compute the mean number of misclassifications

[ ]:
```
misclassification_count = np.sum(y_test != y_pred)

print("Error :", misclassification_count/len(y_test))
```

```
Error : 0.3709677419354839
```

2.8 Question 8

2.8.1 Repeating the same for all the train- test splits mentioned

[ ]:

```
splits = [0.4 , 0.3 , 0.1]

for split in splits:
    df_train , df_test = train_test_split(df, test_size=float(split),␣
 ↪random_state=0)
    X_train = df_train.iloc[:, :-1].values
    y_train = df_train.iloc[:, -1].values
    X_test = df_test.iloc[:, :-1].values
    y_test = df_test.iloc[:, -1].values
    print("Train_test_split: ", split)

    # Train a Logistic Regression model
    Regressor = LogisticRegression()
    Regressor.fit(X_train, y_train)

    # Make predictions on the test set
    y_pred = Regressor.predict(X_test)

    # Report the error
    misclassification_count = np.sum(y_test != y_pred)
    print("Error :", misclassification_count/len(y_test))
    print("\n")
```

```
Train_test_split:  0.4
Error : 0.34959349593495936


Train_test_split:  0.3
Error : 0.3695652173913043


Train_test_split:  0.1
Error : 0.3870967741935484
```

2.8.2 Hence, we can see that as the train-test split increases, the mean number of misclassifications decreases.