LAB EXERCISE - 3 (06-07-2022)

Instructions:

Linear Regression, Multiple Regression, Logistic Regression:

Use the following Template (in that order) for Regression experiments:

- **Load dataset**
- **plot and visualize data**
- **Reshape data**
- **Split data to Train, Validation, Test**
- **Use the model**
- **print the parameters of the model**
- **Print Score**
- **Start prediction**
- **Show classification report**

****

- ✔ **Identify your own dataset (other than IRIS or standard datasets) with atleast 500 to 1000 instances. You can also use a random dataset.Demonstrate Linear Regression, Logistic Regression. You may also attempt polynomial regression for a single variable** $y=a+a1*x+a2*x^{2.}$
- ✔ **Use Gradient Descent approach to arrive at the optimized value of a parameter for linear regression.**

****

LAB EXERCISE - 3.1

LINEAR REGRESSION AND MULTIPLE REGRESSION

Aim of the Experiment

To write Python program for finding linear regression.

Consider the dataset given.

First one can write a linear regression for this problem and can verify that the results are same.

Table 5.1: Sample Data

x **(Week)**	**y** **(Sales in Thousands)**
1	1.2
2	1.8
3	2.6
4	3.2
5	3.8

In listing 2, A random dataset is taken, and multiple regression is applied. This experiment will help to understand the concepts of multiple regression.

The command

X,y = make_regression(n_samples = 50,n_features=1,noise=0.1)

Can create a regression dataset with 50 samples and 1 feature. The number of features field can be changed with 2 for multiple regression as

X,y = make_regression(n_samples = 50,n_features=2,noise=0.1)

WARNING – Random dataset is used for Listing 2 and 3. So, the dataset would be generated at every run. As dataset is generated again, the results would vary every time the program is run.

Listing - 1

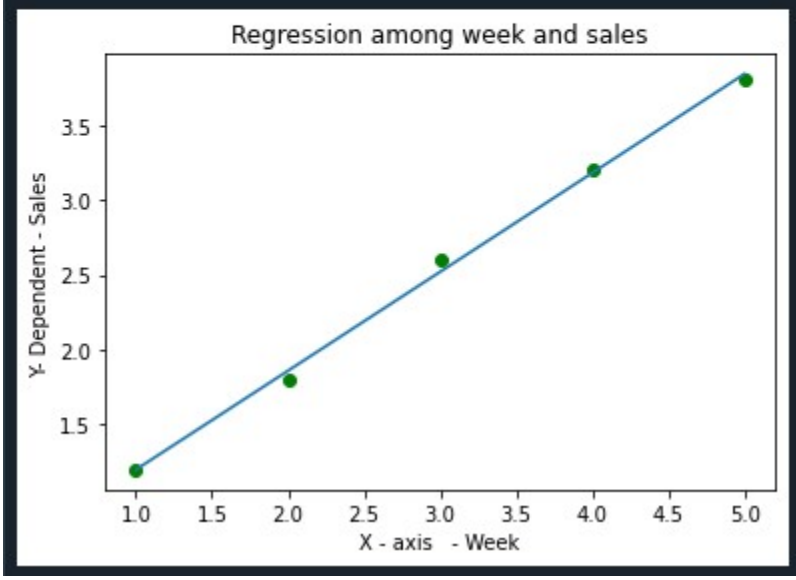
```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn import linear_model
```

```
salesdata = {'week': [1,2,3,4,5],
             'sales': [1.2,1.8,2.6,3.2,3.8]
             }

df = pd.DataFrame(salesdata,columns=['week','sales'])


plt.scatter(df['week'], df['sales'], color='green')
plt.title('Regression among week and sales')
plt.xlabel('X - axis   - Week')
plt.ylabel('Y- Dependent - Sales')


"""

week = df['week'].values.reshape(1,-1)
sales = df['sales'].values.reshape(1,-1)

"""

X = df[['week']]
y = df['sales']

regr = linear_model.LinearRegression()
regr.fit(X,y)

print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)

print('\nThe Regression Equation is',regr.coef_,'* X+',regr.intercept_)
```

```
# Fit the model for the given data

pred = regr.predict(X)
plt.plot(X,pred)

# Compute Adjusted R squared Error
print("\nAdjusted R Squared for Regression model:",regr.score(X,y))
```

Output

```
Intercept:
 0.5400000000000005
Coefficients:
 [0.66]

The Regression Equation is [0.66] * X+ 0.5400000000000005

Adjusted R Squared for Regression model: 0.9972527472527473
```



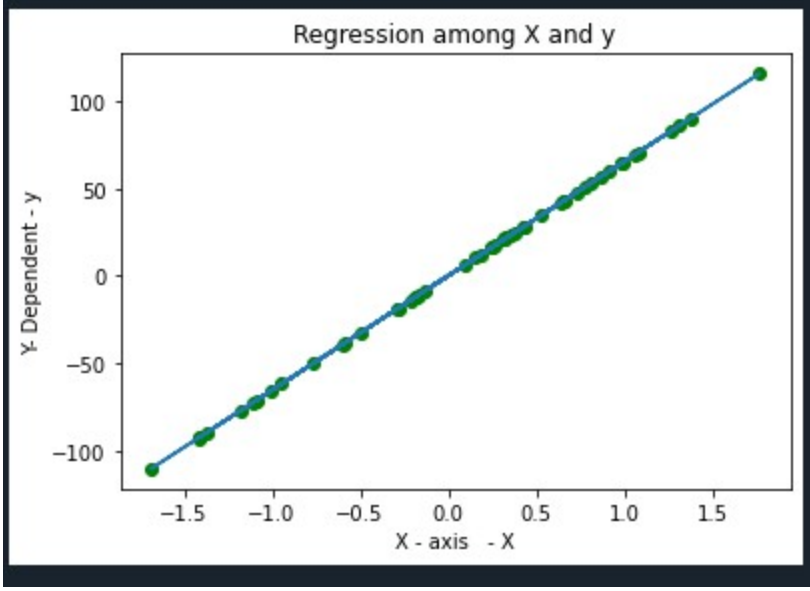
Listing 2

NOTE – Random dataset is used for Listing 2. So, the random dataset would be generated at every run. As dataset is generated again, the results would vary every time the program is run.

```
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.datasets import make_regression

X,y = make_regression(n_samples = 50,n_features=1,noise=0.1)

plt.scatter(X,y,color='green')
plt.title('Regression among X and y')
plt.xlabel('X - axis   - X')
plt.ylabel('Y- Dependent - y')

regr = linear_model.LinearRegression()
regr.fit(X,y)

print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)
print('\nThe Regression Equation is',regr.coef_,'* X +',regr.intercept_)

# Fit the model for the given data

pred = regr.predict(X)
plt.plot(X,pred)

# Compute Adjusted R squared Error
print("\nAdjusted R Squared for Regression model:",regr.score(X,y))
```

Output

```
Intercept:
 -0.01950347285833942
Coefficients:
 [65.47285324]

The Regression Equation is [65.47285324] * X + -0.01950347285833942

Adjusted R Squared for Regression model: 0.99999677094372
```



Listing 3

NOTE– Random dataset is used for Listing 3. So, the random dataset would be generated at every run. As dataset is generated again, the results would vary every time the program is run.

Multiple Regression

```
from sklearn import linear_model

from sklearn.datasets import make_regression


print("Multiple regression \n\n")


# Multiple Regression

# Create random dataset with 2 features. Dataset has 50 samples with noise 0.1.
```

```
X,y = make_regression(n_samples = 50,n_features=2,noise=0.1)

regr = linear_model.LinearRegression()
regr.fit(X,y)

print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)

# Compute Adjusted R squared Error

print("\nAdjusted R Squared for Regression model:",regr.score(X,y))
```

Output

```
Multiple regression


Intercept:
 -0.012331786831634162
Coefficients:
 [53.95803654 36.80928639]

Adjusted R Squared for Regression model: 0.999998052358959
```

LAB EXERCISE – 3.2

Logistic Regression

Aim of the Experiment

The main aim of this experiment is to explore logistic regression model of scikit-learn. The objectives of this experiment are:

1. Explore random dataset generation for logistic regression.
2. Explore logistic regression model in python for randomly generated dataset

Random dataset for classification model can be as follows:

X, y = make_blobs(n_samples=200, centers=3, n_features=3)

The n_samples and n_features can be changed. This has to be imported using the command,

from sklearn.datasets import make_blobs

Logistic regression model can be created by scikit-learn as

model = LogisticRegression()

The algorithm can be applied to the given data as

model.fit(X_train,y_train)

The predictions of the constructed model can be done as

predicted = model.predict(X_test)

The classification report can be generated as follows:

report_lr = classification_report(y_test,predicted)

This classification report must be imported as

from sklearn.metrics import classification_report

NOTE – Random dataset is used for Listing 1. So, the dataset would be generated at every run. As dataset is generated again, the results would vary every time the program is run.

Listing - 1

```
import pandas as pd
from sklearn.datasets import make_blobs
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

X, y = make_blobs(n_samples=200, centers=3, n_features=3)
df = pd.DataFrame(dict(x=X[:,0], y=X[:,1], label=y))
# Print the sample top five records
print("Top five Records\n\n")
df_top = df.head(5)
print(df_top)
 # Condition the input
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.40,random_state=0)
# Construct the logistic regression model
model = LogisticRegression()
# Fit the model
model.fit(X_train,y_train)
#Prediction for the test sample
predicted = model.predict(X_test)
# Print the classification report
print("\n\nClassification Report")
report_lr = classification_report(y_test,predicted)
print(report_lr)
```

Output

The top five records of 200 samples is displayed as follows:

```
Top five Records


          x         y  label
0  0.624773 -4.846126      1
1  7.688025 -4.351219      0
2  4.140132  6.958442      2
3  5.603908  5.247835      2
4  4.720044  6.363697      2
```

The Classification report generated for this problem is shown as follows:

```
Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        26
           1       1.00      1.00      1.00        28
           2       1.00      1.00      1.00        26

    accuracy                           1.00        80
   macro avg       1.00      1.00      1.00        80
weighted avg       1.00      1.00      1.00        80
```