**Experiment No. 9**

**Title: Sklearn for machine learning(Linear Regression)**

**Batch:B1** **Roll No:1814073** **Experiment No.:9**

### Aim: Building Linear Regression model and its evaluation using sklearn

**Resources needed:** Python IDE

### Theory:

 scikit-learn is an open source Python library that implements a range of machine learning, pre-processing, cross-validation and visualization algorithms using a unified interface.

**Important features of scikit-learn:**

* Simple and efficient tools for data mining and data analysis. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means, etc.
* Accessible to everybody and reusable in various contexts.
* Built on the top of NumPy, SciPy, and matplotlib.
* Open source, commercially usable – BSD license.

**Splitting the dataset**

* Split the dataser into two pieces a training set and a testing set.
* Train the model on the training set.
* Test the model on the testing set, and evaluate how well our model did.

# splitting X and y into training and testing sets

**from sklearn.model\_selection import train\_test\_split**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.4, random\_state=1)

print(X\_train.shape)

print(X\_test.shape)

**Training the model**

 Scikit-learn provides a wide range of machine learning algorithms which have a unified/consistent interface for fitting, predicting accuracy, etc.

# training the model on training set

**from** **sklearn.linear\_model** **import LinearRegression**

reg = LinearRegression().fit(X\_train, y\_train)

# making predictions on the testing set

y\_pred = reg.predict(X\_test)

# comparing actual response values (y\_test) with predicted response values (y\_pred)

**from sklearn.metrics import mean\_absolute\_error,mean\_squared\_error,r2\_score**

print('mean absolute error',mean\_absolute\_error(y\_test,y\_pred))

### Activities:

### Download data set appropriate for building Linear Regression model

### (Using the set of some basic attributes that are related to the target variable y)

### Apply appropriate preprocessing before building the model.

### Slice the predictors and target into variables X and y respectively

### Perform train test split on the data.

### Build Linear Regression model.

### Predict on the test data

### Find the Mean Absolute Error, Root Mean Squared Error and R2 Score.

### Result: (script and output)

### 1.

### import pandas as pd, numpy as np,matplotlib.pyplot as plt

### import seaborn as sns

### df=pd.read\_csv("diamonds.csv")

### print(df)

### x=df.drop(["cut","color","clarity","table","price","x","y","z"],axis=1)

### x=x.iloc[:,1:2].values

### y=df.price

### print(x)

### print(y)

### 

### from sklearn.model\_selection import train\_test\_split

### from sklearn.linear\_model import LinearRegression

### train\_x,test\_x,train\_y,test\_y=train\_test\_split(x,y,random\_state=6,test\_size=0.3)

### model=LinearRegression()

### model.fit(train\_x,train\_y)

### pred=model.predict(test\_x)

### print(model.intercept\_)

### print(model.coef\_)

### 

### 

### y\_test\_pred=model.predict(test\_x)

### print(y\_test\_pred)

### model.score(test\_x,test\_y)

### error=test\_y-y\_test\_pred

### print(model.score(test\_x,test\_y))

### 

### from sklearn.metrics import mean\_absolute\_error,mean\_squared\_error,r2\_score

### import math

### print("mean absolute error=",mean\_absolute\_error(test\_y,y\_test\_pred))

### print("root mean squared error=",math.sqrt(mean\_squared\_error(test\_y,y\_test\_pred)))

### print("r2\_score=",r2\_score(test\_y,y\_test\_pred))

### 

### err=test\_y-y\_test\_pred

### sns.distplot(err)

### 

### plt.scatter(test\_y,model.predict(test\_x))

### 

### Questions:

### What is multicollineatry?

**ANS:**

In regression, multicollinearity refers to the extent to which independent variables are correlated. Multicollinearity exists when:

* One independent variable is correlated with another independent variable.
* One independent variable is correlated with a linear combination of two or more independent variables.

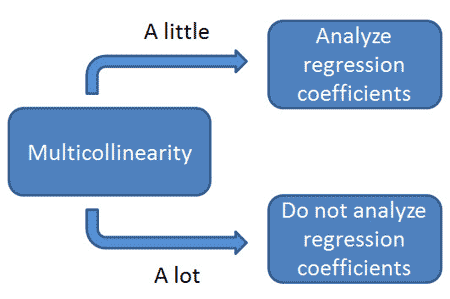
### How does multicollinearity affect the Linear Regession?

### ANS:

The Linear regression analysis makes sense when multicollinearity is small. But it is problematic when multicollinearity is great. Here's why:

* When one independent variable is perfectly correlated with another independent variable (or with a combination of two or more other independent variables), a unique least-squares solution for regression coefficients does not exist.
* When one independent variable is highly correlated with another independent variable (or with a combination of two or more other independent variables), the marginal contribution of that independent variable is influenced by other independent variables. As a result:
  + Estimates for regression coefficients can be unreliable.
  + Tests of significance for regression coefficients can be misleading.

With this in mind, the analysis of regression coefficients should be contingent on the extent of multicollinearity. This means that the analysis of regression coefficients should be preceded by an analysis of multicollinearity.



If the set of independent variables is characterized by a little bit of multicollinearity, the analysis of regression coefficients should be straightforward. If there is a lot of multicollinearity, the analysis will be hard to interpret and can be skipped.

### Outcomes:

### CO4: Illustrate python libraries for machine learning and image processing.

**Conclusion:** (Conclusion to be based on the objectives and outcomes achieved)

### Linear rgression was studied and applied using sklearn.

References:

1.<https://www.geeksforgeeks.org/learning-model-building-scikit-learn-python-machine-learning-library/>

2. <http://scikit-learn.org/stable/documentation.html>