**Experiment No. 7**

**Title: Python Linear Regression using Sklearn**

Batch: B1 Roll No: 1914049 Experiment No.:7

### 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.

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

### Perform train test split on the data.

### Build Linear Regression model.

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

### Result: (script and output)

### import matplotlib.pyplot as plt

### import pandas as pd

### import numpy as np

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

### from sklearn.linear\_model import LinearRegression

### from sklearn.metrics import mean\_squared\_error

### from sklearn import metrics

### from sklearn.datasets import load\_diabetes

### 

### data = load\_diabetes()

### df = pd.DataFrame(data.data, columns=data.feature\_names)

### df.head()

### 

### df.info()

### df.describe()

### 

### df.isnull().sum()

### df.corr()

### 

### df.plot("s1", "s2",color='r')

### df.plot.scatter("s1", "s2",color='r')

### 

### x\_train, x\_test, y\_train, y\_test = train\_test\_split(df.s1, df.s2, test\_size = 0.2)

### regr = LinearRegression()

### regr.fit(np.array(x\_train).reshape(-1,1), y\_train)

### preds = regr.predict(np.array(x\_test).reshape(-1,1))

### y\_test.head(10)

### metrics.r2\_score(y\_test, preds)

### 

### df1 = pd.DataFrame({'Actual data': y\_test, 'Predicted data': preds})

### df1

### preds

### residuals = preds - y\_test

### 

### print('Mean Absolute Error:', metrics.mean\_absolute\_error(y\_test, preds))

### print('Mean Squared Error:', metrics.mean\_squared\_error(y\_test, preds))

### print('Root Mean Squared Error:', np.sqrt(metrics.mean\_squared\_error(y\_test, preds)))

### residuals

### plt.hist(residuals, color='r')

### mean\_squared\_error(y\_test, preds) \*\* 0.5

### 

### print(regr.intercept\_)

### print(regr.coef\_)

### plt.scatter( x\_test, y\_test, color='blue')

### plt.plot(x\_test, preds, color='red', linewidth=3)

### plt.xlabel('x\_test')

### plt.ylabel('y\_test')

### # plt.ylabel('preds')

### plt.xticks(())

### plt.yticks(())

### plt.show()

### 

### Questions:

### 1. What is multicollineatry?

Multicollinearity is the occurrence of high intercorrelations among two or more independent variables in a multiple regression model. Multicollinearity can lead to skewed or misleading results when a researcher or analyst attempts to determine how well each independent variable can be used most effectively to predict or understand the dependent variable in a statistical model.

In general, multicollinearity can lead to wider [confidence intervals](https://www.investopedia.com/terms/c/confidenceinterval.asp) that produce less reliable probabilities in terms of the effect of independent variables in a model. That is, the statistical inferences from a model with multicollinearity may not be dependable.

### 2. How does multicollinearity affect the Linear Regession?

Multicollinearity causes the following two basic types of problems:

* The [coefficient](https://statisticsbyjim.com/glossary/regression-coefficient/) [estimates](https://statisticsbyjim.com/glossary/estimator/) can swing wildly based on which other independent variables are in the model. The [coefficients](https://statisticsbyjim.com/glossary/regression-coefficient/) become very sensitive to small changes in the model.
* Multicollinearity reduces the precision of the estimate coefficients, which weakens the statistical [power](https://statisticsbyjim.com/glossary/power/) of your regression model. You might not be able to trust the p-values to identify independent variables that are statistically significant.

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

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

We learnt how to train a linear regression model and how to split the dataset into 2 parts so that we can predict results later using the test dataset after training the model.

References:

1. 1. Yuxi Liu ; Python Machine Learning By Example: The easiest way to get into machine learning 1st Edition, Kindle Edition , Packt publishing Ltd , 1st edition 2017.
2. Martin C. Brown, The Complete Reference Paperback, Osborne, 2nd edition 2001
3. Frank Millstein, Data Analytics with Python: Data Analytics In Python Using Pandas, Copyright at Frank Millstein, 1st edition 2018
4. <https://www.geeksforgeeks.org/learning-model-building-scikit-learn-python-machine-learning-library/>
5. <http://scikit-learn.org/stable/documentation.html>