Linear Regression

1. Simple Linear Regression

a) Importing the Libraries :-

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

b) Importing Dataset :-

```
dataset = pd.read_csv ('Data.csv')
    X = dataset.iloc [:, :-1].values
    y = dataset.iloc [:,-1].values
```

(integer based)
iloc = locate indexes of rows and columns both
: = range (when specified without lower/upper bound, taking everything in range)

c) Splitting the dataset into Training set and Testing set :-

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split (X, y, test_size = 0.2,
                                                     random_state = 0 )
```

d) Training the Simple Linear Regression model on the Training set :-

```
from sklearn.linear_model import LinearRegression
   regressor = LinearRegression ()
   regressor.fit (X_train, y_train)
```

e) Predicting the Test set :-

```
regressor.predict (X_test)
```

f) Visualizing the Training set results :-

```
plt.scatter (X_train, y_train, color = 'red')
plt.plot (X_train, regressor.predict (X_train), color = 'blue')
plt.title ('Training set Results')
plt.xlabel (' Experience')
plt.ylabel ('Salary')
plt.show ()
```

g) Visualizing the Test set results :-

```
plt.scatter ( X_test, y_test, color = 'red')
plt.plot (X_train, regressor.predict (X_train), color = 'blue')
plt.title ('Test Set Result')
plt.xlabel ('Experience')
plt.ylabel (' Salary')
plt.show()
```

2) Multiple Linear Regression :-

a) Import Libraries

b) Import Dataset :-

```
csv = 50_startups.csv
```

c) Encoding Categorical Data :-

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers = [('encoder', OneHotEncoder(), [3])],
                                        remainder ='pass-through')

X = np.array (ct.fit_transform (x))
```

Encoding the Dependent Variable :-

```
from sklearn.preprocessing import labelEncoder
  le = labelEncoder ()
  y = le.fit_transform (y)
```

d) Splitting the dataset into Training and Testing set.

e) Training MLR model on Training dataset :-

```
from sklearn.linear_model import LinearRegression()
regressor = LinearRegression()
regressor.fit (X_train, y-train )
```

f) Predicting the Test set results :-

```
y-pred = regressor.predict (x_test)
np.set_printoptions (precision = 2)
print(np.concatenate((y-pred.reshape(len(y-pred),1), y_test.reshape(len(y_test),
                                                              1)),1))
```

3) Polynomial Regression :-

a) Import Libraries

b) Import Dataset

c) Training PR model on whole dataset :-

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures (degree = 2)
poly_reg . fit_transform (x)
lin_reg = LinearRegression ()
Lin_reg.fit (x_polynomial, y )
```

d) Visualizing the results :-

(i) Linear Regression

```
plt.scatter (x, y, color = 'red')
plt.plot (x, lin_reg1.predict (x), color = 'blue')
plt.show ()
```

(ii) Polynomial Regression

```
plt.scatter (X, y, color = 'red')
plt.show (X, lin_reg.predict (poly_
    reg.fit_transform(x)), color
    = 'blue')
plt.show()
```

4) Decision Tree :-

a) Regression Model :-

(i) Import Libraries

(ii) Import Dataset

```
dataset = pd.read_csv ('PositionSalaries.csv')
    X = dataset.iloc [:, 1:-1].values
    y = dataset.iloc [:,-1].values
```

(iii) Training DTR on whole dataset :-

```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor()
regressor.fit (X,y)
```

(iv) Predicting a new result :-

```
regressor.predict ([[7.5]])
```

b) Classification Model :-

(i) Import Libraries

(ii) Import Dataset

(iii) Split Dataset into Training and Testing set

(iv) Training DTC on training dataset :-

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier (criterion = 'entropy', random_state = 0)
classifier.fit (X_train, y_train)
```

(v) Predict Results :-

- Test set results

```
y-pred = classifier.predict (X_test)
print(np.concatenate((y-pred.reshape(len(y-pred),1), y-test.reshape(len(y-test),1)),
                                                                            1))
```

- Predicting new result

```
print(classifier.predict(sc.transform([[30,87000]])))
```

(vi) Making Confusion Matrix :-

```
from sklearn.metrics import confusion_matrix, accuracy_score
 cm = confusion_matrix (y_test, y-pred)
accuracy_score (y_test, y-pred)
```

(5) Naive Baye's Classifier :-

a) Import Libraries

b) Import Dataset

c) Split into training and testing set

d) Training Naive Bayes Model on training set

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit (X_train, y_train)
```

e) Predicting the Test Result :-

```
print(classifier.predict (sc.transform ([[30, 87000]])))
```

f) Predicting the Test Results :-

```
y_pred = classifier.predict (X_test)
print (np.concatenate((y_pred.reshape (len (y_pred), 1), y_test.reshape
      (len(y_test), 1)), 1))
```

g) Making the confusion matrix :-

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix (y_test, y_pred)
accuracy_score (y_test, y_pred)
```

6) K-Nearest Neighbour :-

a) Import Libraries

b) Import Dataset

c) Split into training and testing set

d) Training K-NN Model on Training set

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier (n_neighbors = 5, metric = 'minkowski', p=2)
classifier.fit (X_train, y_train)
```

e) Predicting new test data result

f) Predicting the Test Results

g) Making confusion Matrix

7) Support Vector Machine :-

a) Import Libraries

b) Import Dataset

c) Split into training and testing set

d) Training SVM Model on Training set

```
from sklearn.svm import SVC
classifier = SVC (kernel = 'linear', random_state = 0)
classifier.fit (X_train, y_train)
```

e) Predicting new test data result

f) Predicting the test data result

g) Make Confusion Matrix

8) Artificial Neural Network :-

a) Import Libraries :-

```
import pandas as pd
import numpy as np
import tensorflow as tf
```

b) Import Dataset :-

```
dataset = pd.read_csv('Churn_Modelling.csv')
    X = dataset.iloc[:,3:-1].values
    y = dataset.iloc[:,-1].values
```

c) Encoding Categorical data :-

(i) label encoding 'Gender' column

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:,2] = le.fit_transform(X[:,2])
```

(ii) One Hot Encoding 'Geography' column

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers = [('encoder', OneHotEncoder(), [1])],
        remainder = 'passthrough')
  X = np.array(ct.fit_transform(X))
```

d) Split Dataset into Training and Testing set

e) Feature Scaling :-

```
from sklearn.preprocessing import StandardScaler
  sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test  = sc.transform(X_test)
```

f) Initialize ANN :-

```
ann = tf.keras.models.Sequential()
```

g) Adding input layer and 1st Hidden Layer :-

```
ann.add(tf.keras.layers.Dense(units = 6, activation = 'relu'))
```

h) Adding second hidden layer :-

```
ann.add(tf.keras.layers.Dense(units = 6, activation = 'relu'))
```

i) Adding the output layer :-

```
ann.add(tf.keras.layers.Dense(units = 1, activation = 'sigmoid'))
```

j) Training the ANN :-

(i) Compiling the ANN :-

```
ann.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

(ii) Training the ANN on training set :-

```
ann.fit(X_train, y_train, batch_size = 32, epochs = 100)
```

k) Making Predictions and Evaluating the Model :-

e.g. Geography = France, Credit Score = 600, Gender = Male, Age = 40 years, Tenure = 3 years, Balance = 60000, Number of Products = 2, Credit Card = Yes, Active Member = Yes, Estimated Salary = 50000, Goodbye = ?

```
print(ann.predict (sc.transform([[1,0,0,600,1,40,3,60000,2,1,1,50000]])))
```

l) Predicting the Test Set results :-

```
    y_pred = ann.predict (X_test)
    y_pred = (y_pred > 0.5)
    print (np.concatenate((y_pred.reshape(len(y_pred),1),y_test.reshape(len(
                                                                y_test),1)),1))
```

m) Make Confusion Matrix :-

```
    from sklearn.metrics import confusion_matrix, accuracy_score
    cm = confusion_matrix(y_test, y_pred)
    accuracy_score (y_test, y_pred)
```

9) K-Means Clustering :-

a) Import Libraries

b) Import Dataset :-

```
    dataset = pd.read_csv ('Mall_Customers.csv')
    X = dataset.iloc [:,[3,4]].values
```

c) Using the elbow method to find optimal no. of clusters :-

```
    from sklearn.cluster import KMeans
    wcss = [ ]
    for i in range (1,11):
        kmeans = KMeans (n_clusters = i , init = 'k-means++', random_state = 0)
        kmeans.fit (x)
        wcss.append (kmeans.inertia_)
    plt.plot (range (1,11), wcss)
    plt.show ()
```

d) Training the k-Means Model on the dataset :-

```
    kmeans = KMeans(n_clusters = 5, init ='k-means++', random_state = 0)
    y_kmeans = kmeans.fit_predict (x)
    print(y_kmeans)
```