WEEK – 4

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

dataset = pd.read_csv('Salary_Data.csv')

dataset.head()

# data preprocessing

X = dataset.iloc[:, :-1].values #independent variable array

y = dataset.iloc[:,1].values #dependent variable vector

# splitting the dataset

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=1/3,random_state=0)

# fitting the regression model

from sklearn.linear_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X_train,y_train) #actually produces the linear eqn for the data

# predicting the test set results

y_pred = regressor.predict(X_test)

y_pred

y_test

# visualizing the results

#plot for the TRAIN

plt.scatter(X_train, y_train, color='red') # plotting the observation line

plt.plot(X_train, regressor.predict(X_train), color='blue') # plotting the regression line

plt.title("Salary vs Experience (Training set)") # stating the title of the graph

plt.xlabel("Years of experience") # adding the name of x-axis

plt.ylabel("Salaries") # adding the name of y-axis

plt.show() # specifies end of graph
```

```
#plot for the TEST

plt.scatter(X_test, y_test, color='red')
plt.plot(X_train, regressor.predict(X_train), color='blue') # plotting the regression line
plt.title("Salary vs Experience (Testing set)")
plt.xlabel("Years of experience")
plt.ylabel("Salaries")
plt.show()
```

WEEK – 5

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn import preprocessing
# importing data
df = pd.read_csv('Real-estate1.csv')
df.drop('No', inplace=True, axis=1)
print(df.head())
print(df.columns)
# plotting a scatterplot
sns.scatterplot(x='X4 number of convenience stores',
```

```
y='Y house price of unit area', data=df)

# creating feature variables
X = df.drop('Y house price of unit area', axis=1)
y = df['Y house price of unit area']
print(X)
print(y)
# creating train and test sets
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.3, random_state=101)
# creating a regression model
model = LinearRegression()
# fitting the model
model.fit(X_train, y_train)
# making predictions
predictions = model.predict(X_test)
# model evaluation
print('mean_squared_error : ', mean_squared_error(y_test, predictions))

print('mean_absolute_error : ', mean_absolute_error(y_test, predictions))
```

WEEK – 6

```
import numpy as np
import pandas as pd
```

```
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier

import matplotlib.pyplot as plt

from sklearn.tree import DecisionTreeClassifier, plot_tree

def importdata():

  balance_data = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-' +
'databases/balance-scale/balance-scale.data',sep=',', header=None)

  print("Dataset Length: ", len(balance_data))

  print("Dataset Shape: ", balance_data.shape)

  print("Dataset: ", balance_data.head())

  return balance_data


def splitdataset(balance_data):

# Separating the target variable

  X = balance_data.values[:, 1:5]

  Y = balance_data.values[:, 0]

  X_train, X_test, y_train, y_test = train_test_split(

    X, Y, test_size=0.3, random_state=100)

  return X, Y, X_train, X_test, y_train, y_test

def train_using_gini(X_train, X_test, y_train):

# Creating the classifier object

  clf_gini = DecisionTreeClassifier(criterion="gini",random_state=100,max_depth=3,
min_samples_leaf=5)

  clf_gini.fit(X_train, y_train)

  return clf_gini

def train_using_entropy(X_train, X_test, y_train):


# Decision tree with entropy
```

```
    clf_entropy = DecisionTreeClassifier(criterion="entropy",
random_state=100,max_depth=3, min_samples_leaf=5)
    clf_entropy.fit(X_train, y_train)
    return clf_entropy
# Function to make predictions
def prediction(X_test, clf_object):
    y_pred = clf_object.predict(X_test)
    print("Predicted values:")
    print(y_pred)
    return y_pred
# Placeholder function for cal_accuracy
def cal_accuracy(y_test, y_pred):
    print("Confusion Matrix: ",confusion_matrix(y_test, y_pred))
    print("Accuracy : ",accuracy_score(y_test, y_pred)*100)
    print("Report : ",classification_report(y_test, y_pred))
# Function to plot the decision tree
def plot_decision_tree(clf_object, feature_names, class_names):
    plt.figure(figsize=(15, 10))
    plot_tree(clf_object, filled=True,
feature_names=feature_names,class_names=class_names, rounded=True)
    plt.show()
if _name_ == "_main_":
    data = importdata()
    X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
    clf_gini = train_using_gini(X_train, X_test, y_train)
    clf_entropy = train_using_entropy(X_train, X_test, y_train)
    plot_decision_tree(clf_gini, ['X1', 'X2', 'X3', 'X4'], ['L', 'B', 'R'])
    plot_decision_tree(clf_entropy, ['X1', 'X2', 'X3', 'X4'], ['L', 'B', 'R'])
    print("Results Using Gini Index:")
```

```
    y_pred_gini = prediction(X_test, clf_gini)
    cal_accuracy(y_test, y_pred_gini)
```

WEEK – 7

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

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']

df = pd.read_csv(url, names=names)
df

X = df.iloc[:, :-1].values
y = df.iloc[:, 4].values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

WEEK – 8

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# load dataset
diab_df = pd.read_csv("diabetes.csv")
diab_df.head()
#split dataset in features and target variable
diab_cols = ['Pregnancies', 'Insulin', 'BMI',
'Age','Glucose','BloodPressure','DiabetesPedigreeFunction']
X = diab_df[diab_cols]# Features
y = diab_df.Outcome # Target variable
# Splitting Data
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=0)
# Model Development and Prediction and instantiate the model
logreg = LogisticRegression(solver='liblinear')
# fit the model with data
logreg.fit(X_train,y_train)
# predicting
y_pred=logreg.predict(X_test)
y_pred

# Model Evaluation using Confusion Matrix
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
# Visualizing Confusion Matrix using Heatmap
class_names=[0,1] # name of classes
fig, ax = plt.subplots()

tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
# Confusion Matrix Evaluation Metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
```

WEEK – 9

```
import numpy as np
from sklearn.datasets import make_blobs
class KMeans:
  def _init_ (self, n_clusters, max_iters=100):
    self.n_clusters = n_clusters
    self.max_iters = max_iters
def fit(self, X):
  self.centroids = X[np.random.choice(X.shape[0], self.n_clusters, replace=False)]
  for _ in range(self.max_iters):
    labels = self._assign_labels(X)
    new_centroids = self._update_centroids(X, labels)
    if np.all(self.centroids == new_centroids):
      break
      self.centroids = new_centroids
def _assign_labels(self, X):
  distances = np.linalg.norm(X[:, np.newaxis] - self.centroids, axis=2)
  return np.argmin(distances, axis=1)
def _update_centroids(self, X, labels):
  new_centroids = np.array([X[labels == i].mean(axis=0)
                 for i in range(self.n_clusters)])
  return new_centroids
  X, _ = make_blobs(n_samples=300, centers=3, random_state=42)
  kmeans = KMeans(n_clusters=3)
```

```
    kmeans.fit(X)

    labels = kmeans._assign_labels(X)

    print("Cluster Assignments:", labels)

    print("Final Centroids:", kmeans.centroids)
```

EXPERIMENT 10 ML

```
import pandas as pd

from pandas.plotting import scatter_matrix

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split, KFold, cross_val_score

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

from sklearn.naive_bayes import GaussianNB

from sklearn.svm import SVC
```

```
# 1. Load dataset

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']

dataset = pd.read_csv(url, names=names)


# 2. Summarize the Dataset

# Shape

print(dataset.shape)


# Plotting a scatterplot

sns.scatterplot(x='sepal-length', y='petal-length', data=dataset)

plt.show()


# Creating feature variables

X = dataset.drop('class', axis=1)

y = dataset['class']

print(X)

print(y)


# Creating train and test sets

validation_size = 0.20

seed = 7

X_train, X_validation, Y_train, Y_validation = train_test_split(X, y,
test_size=validation_size, random_state=seed)


# Test Harness

seed = 7

scoring = 'accuracy'
```

```
# Build and evaluate models
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))

# Evaluate each model in turn
results = []
names = []
for name, model in models:
  kfold = KFold(n_splits=10, random_state=seed, shuffle=True)
  cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring=scoring)
  results.append(cv_results)
  names.append(name)
  msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
  print(msg)

# Compare Algorithms
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

```
# Multivariate Plots

# Scatter plot matrix

scatter_matrix(dataset)

plt.show()


# Box plot

dataset.plot(kind='box', subplots=True, layout=(2, 2), sharex=False, sharey=False)

plt.show()


# Histogram

dataset.hist()

plt.show()


# Descriptive statistics

print(dataset.describe())


# Class distribution

print(dataset.groupby('class').size())
```