1. **Write a Program to implement Linear Regression (Scikit) involving single variable and analyze the house price prediction. Draw graph and plot correlation matrix and determine precision, recall and F1 measure**

**Step-1**

import pandas as pd

import numpy as np

import seaborn as sns

from sklearn import preprocessing

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

from sklearn import linear\_model

from sklearn.metrics import r2\_score

from sklearn.metrics import classification\_report,precision\_score, recall\_score, f1\_score, accuracy\_score,confusion\_matrix

**Step-2**

df=pd.read\_csv("C:/Users/Admin/Downloads/house\_price.csv")

df.head()

**Step-3**

x = df[['sqft\_living']]

y = df['price']

sns.pairplot(df)

sns.heatmap(df.corr(),annot=True)

**Step-4**

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=42)

regressor = linear\_model.LinearRegression()

regressor.fit(x\_train,y\_train)

y\_pred = regressor.predict(x\_test)

y\_pred

**Step : 5**

threshold=500000

y\_pred\_bin=np.where(y\_pred < threshold,0,1)

y\_test\_bin=np.where(y\_test < threshold,0,1)

print(precision\_score(y\_test\_bin,y\_pred\_bin))

print(recall\_score(y\_test\_bin,y\_pred\_bin))

print(f1\_score(y\_test\_bin,y\_pred\_bin))

print(confusion\_matrix(y\_test\_bin,y\_pred\_bin))

1. **Write a Program to implement Linear Regression (Scikit) involving multiple variables and analyze the house price prediction. Draw graph, plot correlation matrix and determine accuracy.**

**STEP 1:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

**STEP: 2:**

df=pd.read\_csv("C:/Users/Sankamethra/Downloads/data.csv")

df

**STEP:3:**

X = df.iloc[:,2:32]

print(X.shape)

X.head()

y = df.view

print(y.shape)

y.head()

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=4)

**STEP:4:**

refined\_cols = ['sqft\_living','sqft\_lot','sqft\_above']

from sklearn.linear\_model import LinearRegression

MultiLR = LinearRegression()

MultiLR.fit(X\_train[refined\_cols],y\_train)

y\_pred = MultiLR.predict(X\_test[refined\_cols])

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

print('MSE',mean\_squared\_error(y\_test, y\_pred))

print('r2',r2\_score(y\_test,y\_pred))

1. **Write a Program to implement Logistic Regression by plotting the decision boundary and use it to classify spam mail. Draw graph, plot correlation matrix and determine precision, recall and F1 measure**

**STEP-1:**

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.decomposition **import** PCA

**from** sklearn.metrics **import** f1\_score

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

**from** sklearn.model\_selection **import** train\_test\_split, cross\_val\_score

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**import** numpy **as** np

**from** sklearn.metrics **import** accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix

**STEP-2:**

df**=**pd**.**read\_csv("C:\\Users\\spd85\\Downloads\\spam\_ham\_dataset.csv")

df**.**head()

**STEP-3:**

print(df**.**info())

df**.**describe()

**STEP-4:**

sns**.**heatmap(df**.**corr(), annot **=** **True**)

**STEP-5:**

sns**.**pairplot(df,hue **=**'label')

**STEP-6:**

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

X**=**df['label']

y**=**df['label\_num']

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size**=**0.2, random\_state**=**34, stratify**=**y)

**STEP-7:**

**from** sklearn.feature\_extraction.text **import** CountVectorizer

count\_vector **=** CountVectorizer()

training\_data **=** count\_vector**.**fit\_transform(X\_train)**.**toarray()

testing\_data **=** count\_vector**.**transform(X\_test)**.**toarray()

clf **=** LogisticRegression(random\_state**=**0)**.**fit(training\_data, y\_train)

predictions **=** clf**.**predict(testing\_data)

predictions

**STEP-8:**

print('Accuracy score: ', format(accuracy\_score(y\_test, predictions)))

print('Precision score: ', format(precision\_score(y\_test, predictions)))

print('Recall score: ', format(recall\_score(y\_test, predictions)))

print('F1 score: ', format(f1\_score(y\_test, predictions)))

print('\nConfusion Matrix :\n', confusion\_matrix(y\_test, predictions))

1. **Using Principal component Analysis as Dimensionality reduction component, implement Linear Regression (Scikit) involving single variable and analyze the house price prediction. Draw graph, plot correlation matrix and determine precision, recall and F1 measure**

**STEP-1:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

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

from sklearn.metrics import classification\_report,confusion\_matrix, precision\_score, recall\_score, f1\_score

from sklearn.decomposition import PCA

**STEP-2:**

df = pd.read\_csv('C:/Users/welcome/Downloads/USA\_Housing.csv')

df.head()

**STEP-3:**

X = df[['sqft\_living']]

y = df['price']

**STEP-4:**

pca = PCA(n\_components=1)

X\_pca = pca.fit\_transform(X)

correl=df.corr()

sns.heatmap(correl,annot=True)

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.2,random\_state=42)

regressor = LinearRegression()

regressor.fit(X\_train,y\_train)

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

y\_pred

threshold = 500000

**STEP-5**:

y\_pred\_bin = np.where(y\_pred < threshold, 0, 1)

y\_test\_bin = np.where(y\_test < threshold, 0, 1)

**STEP-6:**

cm = confusion\_matrix(y\_test\_bin, y\_pred\_bin)

precision = precision\_score(y\_test\_bin, y\_pred\_bin)

recall = recall\_score(y\_test\_bin, y\_pred\_bin)

f1 = f1\_score(y\_test\_bin, y\_pred\_bin)

**STEP-7**

print("Confusion Matrix:")

print(cm)

print("Precision:", precision)

print("Recall:", recall)

print("F1-score:", f1)

1. **Using Principal component Analysis as Dimensionality reduction component implement Linear Regression (Scikit) involving multiple variables and analyze the house price prediction. Draw graph, plot correlation matrix and determine accuracy**

**Step : 1**

import pandas as pd

import numpy as np

import seaborn as sns

from sklearn import preprocessing

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

from sklearn import linear\_model

from sklearn.metrics import r2\_score

from sklearn.metrics import classification\_report,precision\_score, recall\_score, f1\_score, accuracy\_score,confusion\_matrix

**Step : 2**

data = pd.read\_csv("C:/Users/Admin/Downloads/house\_price.csv")

data.head()

**Step : 3**

x = data[['sqft\_living']]

y = data['price']

**Step : 4**

plt.figure(figsize=(20,12))

sns.heatmap(data.corr(),annot=True)

plt.show()

**Step : 5**

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=42)

regressor = linear\_model.LinearRegression()

regressor.fit(x\_train,y\_train)

y\_pred = regressor.predict(x\_test)

y\_pred

**Step : 6**

threshold=500000

y\_pred\_bin=np.where(y\_pred < threshold,0,1)

y\_test\_bin=np.where(y\_test < threshold,0,1)

print(precision\_score(y\_test\_bin,y\_pred\_bin))

print(recall\_score(y\_test\_bin,y\_pred\_bin))

print(f1\_score(y\_test\_bin,y\_pred\_bin))

print(confusion\_matrix(y\_test\_bin,y\_pred\_bin))

1. **Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs, plot correlation matrix and determine accuracy**

**STEP-1:**

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

**STEP-2:**

def kernel(point, xmat, k):

m,n = np.shape(xmat)

weights = np.mat(np.eye((m)))

for j in range(m):

diff = point - X[j]

weights[j,j] = np.exp(diff\*diff.T/(-2.0\*k\*\*2))

  return weights

**STEP-3:**

def localWeight(point, xmat, ymat, k):

wei = kernel(point,xmat,k)

W = (X.T\*(wei\*X)).I\*(X.T\*(wei\*ymat.T))

    return W

**STEP-4:**

def localWeightRegression(xmat, ymat, k):

m,n = np.shape(xmat)

ypred = np.zeros(m)

for i in range(m):

ypred[i] = xmat[i]\*localWeight(xmat[i],xmat,ymat,k)

    return ypred

**STEP-5:**

data = pd.read\_csv('C:\\Users\\spd85\\Downloads\\abc\\tips.csv')

data.head()

**STEP-6**

bill = np.array(data.total\_bill)

tip = np.array(data.tip)

mbill = np.mat(bill)

mtip = np.mat(tip)

**STEP-7**

m= np.shape(mbill)[1]

one = np.mat(np.ones(m))

X = np.hstack((one.T,mbill.T))

**STEP-8**

ypred = localWeightRegression(X,mtip,0.5)

SortIndex = X[:,1].argsort(0)

xsort = X[SortIndex][:,0]

**STEP : 9**

fig = plt.figure()

ax = fig.add\_subplot(1,1,1)

ax.scatter(bill,tip, color='green')

ax.plot(xsort[:,1],ypred[SortIndex], color = 'red', linewidth=5)

plt.xlabel('Total bill')

plt.ylabel('Tip')

plt.show();

1. **Using Principal component Analysis as Dimensionality reduction component implement Logistic Regression for detecting credit card frauds. Draw graph, plot correlation matrix and determine precision, recall and F1 measure**

**STEP-1:**

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**import** warnings

warnings**.**filterwarnings('ignore')

**from** sklearn.decomposition **import** PCA

**STEP-2:**

dataset **=** pd**.**read\_csv("C:/Users/welcome/Downloads/archive (1)/creditcard.csv")

**STEP-3:**

corr **=** dataset**.**corr()

plt**.**figure(figsize **=** (35,15))

sns**.**heatmap(corr, annot **=** **True**, cmap **=** 'coolwarm', linewidth **=** 2)

**STEP-4:**

x **=** dataset**.**drop(columns **=** 'Class')

In [6]:

y **=** dataset['Class']

**STEP-5:**

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

x\_train,x\_test,y\_train,y\_test **=** train\_test\_split(x,y, test\_size **=** 0.33, random\_state **=** 0)

**from** sklearn.preprocessing **import** StandardScaler

pca **=** PCA(n\_components**=**5)

x\_train **=** pca**.**fit\_transform(x\_train)

x\_test **=** pca**.**transform(x\_test)

**STEP-6:**

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

logreg **=** LogisticRegression()

logreg**.**fit(x\_train,y\_train)

**STEP-7:**

y\_pred **=** logreg**.**predict(x\_test)

**from** sklearn.metrics **import** accuracy\_score, precision\_score, recall\_score, f1\_score

acc **=** accuracy\_score(y\_test,y\_pred)

f1 **=** f1\_score(y\_test,y\_pred)

prec **=** precision\_score(y\_test,y\_pred)

rec **=** recall\_score(y\_test,y\_pred)

results **=** pd**.**DataFrame([['Logistic Regression', acc, f1, prec, rec]],

columns **=** ['Model','Accuracy Score','F1 score','Precision','Recall'])

results

1. **Use a sample dataset and with help of Support Vector Machine, classify the subject whether it has cancer or not. Draw graph, plot correlation matrix and determine precision, recall and F1 measure**

**STEP-1:**

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

from sklearn.svm import SVC

from sklearn.model\_selection import train\_test\_split,cross\_val\_score

from sklearn.metrics import confusion\_matrix, classification\_report

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

**STEP-2:**

# df = pd.read\_csv('C:\\Users\\spd85\\Downloads\\cancer\\data.csv')

# df.head()

**STEP-3:**

df.dtypes

df[df.duplicated()].shape

df.describe()

df.columns

**STEP-4:**

# plt.figure(figsize=(20,12))

# sns.heatmap(df.corr(),annot=True)

# plt.show()

# X = df.iloc[:,2:32]

# print(X.shape)

# X.head()

**STEP-5:**

**Normalization and PCA.**

scaler = StandardScaler( )

X\_scaled = pd.DataFrame(scaler.fit\_transform(X))

X\_scaled\_drop = X\_scaled.drop(X\_scaled.columns[[2,3,12,13,22,23]],axis=1)

pca = PCA(n\_components=0.95)

x\_pca = pca.fit\_transform(X\_scaled\_drop)

x\_pca = pd.DataFrame(x\_pca)

print("Before PCA, X dataframe shape = ",X.shape,"\nAfter PCA, x\_pca dataframe shape = ",x\_pca.shape)

print(pca.explained\_variance\_ratio\_)

print(pca.explained\_variance\_ratio\_.sum())

y = df.diagnosis

print(y.shape)

y.head()

print(x\_pca.shape)

print(y.shape)

**STEP-6:**

# X\_train, X\_test, y\_train, y\_test = train\_test\_split(x\_pca, y, test\_size=0.25, random\_state=0)

# svc = SVC()

# svc.fit(X\_train, y\_train)

# y\_pred = svc.predict(X\_test)

# cm = confusion\_matrix(y\_test, y\_pred)

# print("Confusion matrix:\n",cm)

# report = classification\_report(y\_test, y\_pred)

# print("Classification report:\n",report)

1. **Implement KNN algorithm using the balanced iris data set for multiclass classification and predict the flower species. Draw graph, plot correlation matrix and determine precision, recall and F1 measure**

**STEP 1:**

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split,cross\_val\_score

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings("ignore")

import pandas as pd

import numpy as np

iris = load\_iris()

**STEP 2:**

x = iris.data

y = iris.target

print(x.shape)

data = np.c\_[iris.data, iris.target]

columns = np.append(iris.feature\_names, ["target"])

df = pd.DataFrame(data, columns=columns)

print(df)

print(iris.feature\_names)

**STEP 3:**

df[iris.feature\_names].describe()

**STEP 4:**

df[iris.feature\_names].describe()

sns.heatmap(df[iris.feature\_names].corr(),annot=True)

plt.plot()

**STEP 5:**

sns.boxplot(x="target", y="petal width (cm)", data=df)

**STEP 6:**

sns.boxplot(x="target", y="petal length (cm)", data=df)

**STEP 7:**

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.2, random\_state=4)

print(x\_train.shape)

print(x\_test.shape)

**STEP 8:**

from sklearn.neighbors import KNeighborsClassifier

from sklearn import metrics

test\_k = range(1,26)

scores = []

for k in test\_k:

knn = KNeighborsClassifier(n\_neighbors=k)

knn.fit(x\_train,y\_train)

y\_pred = knn.predict(x\_test)

scores.append(metrics.accuracy\_score(y\_test,y\_pred))

**STEP 9:**

plt.plot(test\_k,scores)

plt.xlabel('k value for kNN')

plt.ylabel('Accuracy on test data')

cv\_scores = []

for k in test\_k:

knn = KNeighborsClassifier(n\_neighbors=k)

score = cross\_val\_score(knn, x\_train, y\_train, cv=10, scoring='accuracy')

cv\_scores.append(score.mean())

MSE = [1 - x for x in cv\_scores]

**STEP 10:**

plt.title("The optimal number of neighbors")

plt.xlabel("Number of Neighbors K")

plt.ylabel("MisClassification Error")

plt.plot(test\_k, MSE)

plt.show()

**STEP 11:**

knn = KNeighborsClassifier(n\_neighbors=12)

knn.fit(x\_train, y\_train)

y\_pred = knn.predict(x\_test)

metrics.confusion\_matrix(y\_test,y\_pred)

metrics.accuracy\_score(y\_test,y\_pred)

1. **Write a program to implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets. Draw graph, plot correlation matrix and determine precision, recall and F1 measure**

**STEP-1:**

import pandas as pd

df=pd.read\_csv("tennis.csv")

df

**STEP-2:**

X\_train = pd.get\_dummies(df[['outlook', 'temp', 'humidity', 'windy']])

y\_train = pd.DataFrame(df['play'])

print(X\_train.info())

print(X\_train.head())

**STEP-3:**

print(y\_train.info())

print(y\_train)

**STEP-4:**

from sklearn.naive\_bayes import GaussianNB

classifier=GaussianNB()

classifier.fit(X\_train,y\_train)

classifier.score(X\_train,y\_train)

X\_train.head()

classifier.predict([[True,0,0,1,0,1,0,1,0]])

**STEP-5:**

y\_train.head()

a=classifier.predict([[True,0,0,1,0,1,1,1,1]])

if(a[0]=="yes"):

print("yOU CAN PLAY!!!!")

else:

print("You cant play!!!")

1. **Write a program to implement k-means clustering algorithm for iris dataset. Draw graph, plot correlation matrix and determine precision, recall and F1 measure**

**STEP-1:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_score

from sklearn.preprocessing import MinMaxScaler

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_score

from sklearn.preprocessing import MinMaxScaler

**STEP-2:**

iris= pd.read\_csv("C:/Users/Sankamethra/Documents/3rdYear/ML/LAB/archive (7)/IRIS.csv")

x=iris.iloc[:,[0,1,2,3]].values

iris.info()

iris[0:10]

**STEP-3:**

iris\_outcome=pd.crosstab(index=iris["Species"],columns="count")

iris\_outcome

**STEP-4:**

iris\_setosa= iris.loc[iris["Species"]=="Iris-setosa"]

iris\_virginica=iris.loc[iris["Species"]=="Iris-virginica"]

iris\_versicolor=iris.loc[iris["Species"]=="Iris-versicolor"]

**STEP-5:**

sns.FacetGrid(iris,hue="Species",size=3).map(sns.distplot,"PetalLengthCm").add\_legend()

sns.FacetGrid(iris,hue="Species",size=3).map(sns.distplot,"PetalWidthCm").add\_legend()

sns.FacetGrid(iris,hue="Species",size=3).map(sns.distplot,"SepalLengthCm").add\_legend()