**1.SINGLE VARIABLE LINEAR REGRESSION**

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.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

df = pd.read\_csv("D:/Downloads/USA\_Housing.csv")

df

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

x=df[['Area Population']]

y=df['Price']

plt.scatter(x,y)

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

model=LinearRegression()

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

predictor=model.predict(x\_test)

predictor

print(mean\_squared\_error(y\_test,predictor))

**2.SINGLE LINEAR REGRESSION USING PCA**

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.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

from sklearn.decomposition import PCA

df = pd.read\_csv("D:/Downloads/USA\_Housing.csv")

df

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

x=df[['Area Population']]

y=df['Price']

plt.scatter(x,y)

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

pca=PCA(n\_components=1)

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

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

model=LinearRegression()

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

predictor=model.predict(x\_test)

predictor

print(mean\_squared\_error(y\_test,predictor))sss

**3.MULTIPLE LINEAR REGRESSION**

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.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

df = pd.read\_csv("D:/Downloads/USA\_Housing.csv")

df

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

x=df[['Area Population']]

y=df['Price']

plt.scatter(x,y)

x=df[['Area Population','Avg. Area Income','Avg. Area Number of Rooms','Avg. Area Number of Bedrooms','Avg. Area House Age']]

y=df['Price']

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=100)

model=LinearRegression()

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

predictor=model.predict(x\_test)

predictor

print(mean\_squared\_error(y\_test,predictor))

**4.MULTIPLE LINEAR REGRESSION USING PCA**

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.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

df = pd.read\_csv("D:/Downloads/USA\_Housing.csv")

df

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

x=df[['Area Population']]

y=df['Price']

plt.scatter(x,y)

x=df[['Area Population','Avg. Area Income','Avg. Area Number of Rooms','Avg. Area Number of Bedrooms','Avg. Area House Age']]

y=df['Price']

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=100)

pca=PCA(n\_components=5)

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

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

model=LinearRegression()

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

predictor=model.predict(x\_test)

predictor

print(mean\_squared\_error(y\_test,predictor))

**5.LOGISTIC REGRESSION**

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

data = pd.read\_csv("C:/Users/welcome/OneDrive - Kumaraguru College of Technology/Desktop/ML PROGRAMS/SPAM data.csv")

le = LabelEncoder()

data['Category\_enc'] = le.fit\_transform(data['Category'])

display(data.head())

X = data['Message']

y = data['Category\_enc']

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

from sklearn.feature\_extraction.text import CountVectorizer

count\_vector = CountVectorizer()

# Fit the training data and then return the matrix

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

# Transform testing data and return the matrix. Note we are not fitting the testing data into the CountVectorizer()

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

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

predictions = clf.predict(testing\_data)

predictions

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))

**6.LOGISTIC USING PCA**

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

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

corr = dataset.corr()

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

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

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

y = dataset['Class']

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)

from sklearn.linear\_model import LogisticRegression

logreg = LogisticRegression()

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

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

**7.KNN**

import numpy as np

import pandas as pd

from sklearn import datasets

import seaborn as sns

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

from sklearn.neighbors import KNeighborsClassifier

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

import matplotlib.pyplot as plt

iris=datasets.load\_iris()

x=iris.data

y=iris.target

iris\_df = sns.load\_dataset("iris")

sns.pairplot(iris\_df, hue="species")

iris\_df = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

cor = iris\_df.corr()

print(cor)

sns.heatmap(cor,annot=True)

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=50)

classifier = KNeighborsClassifier(n\_neighbors=5)

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

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

print(classification\_report(y\_test,y\_pred))

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

print(accuracy\_score(y\_test,y\_pred))

**8.KMEANS**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn import datasets

from sklearn.cluster import KMeans

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

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

iris=datasets.load\_iris()

x=iris.data

y=iris.target

iris\_df = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

cor = iris\_df.corr()

print(cor)

sns.heatmap(cor,annot=True)

iris\_df = sns.load\_dataset("iris")

sns.pairplot(iris\_df, hue="species")

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=5)

kmeans = KMeans(n\_clusters=3)

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

train\_labels= kmeans.predict(x\_train)

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

print(classification\_report(y\_test,y\_pred))

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

print(accuracy\_score(y\_test,y\_pred))

**9.SVM**

import numpy as np

import seaborn as sns

import pandas as pd

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

from sklearn import svm

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

data = pd.read\_csv("C:/Users/welcome/Downloads/archive (2)/lung\_cancer\_examples.csv")

sns.pairplot(data)

x = data[['Age','Smokes','AreaQ','Alkhol']]

y = data['Result']

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

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=.5)

classifier = svm.SVC(kernel='linear')

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

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

print(classification\_report(y\_test,y\_pred))

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

print(accuracy\_score(y\_test,y\_pred))

**10.NAIVE BAYES**

import pandas as pd

from sklearn.datasets import load\_iris

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

from sklearn.naive\_bayes import GaussianNB

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

iris = load\_iris()

x=iris.data

y=iris.target

iris\_df = sns.load\_dataset("iris")

sns.pairplot(iris\_df, hue="species")

iris\_df = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

cor = iris\_df.corr()

sns.heatmap(cor,annot=True)

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=50)

classifier = GaussianNB()

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

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

print(classification\_report(y\_test,y\_pred))

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

print(accuracy\_score(y\_test,y\_pred))