**Aim:** Write a program to implement the linear regression using stochastic gradient descent approach of training for a sample training data set stored as a .CSV file.

**Source code:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

def predict(row, coefficients):

# Predicted value using linear regression equation

yhat = coefficients[0]

for i in range(len(row) - 1):

yhat += coefficients[i + 1] \* row[i]

return yhat

def sgd(train, learning\_rate, epochs):

# Initialize coefficients to zero

coef = [0.0 for i in range(len(train.columns))]

m = len(train.index) # Number of data points

for epoch in range(epochs):

sum\_error = 0

for \_ in range(m):

random\_index = np.random.randint(m) # Randomly select a data point

row = train.iloc[random\_index, :]

yhat = predict(row, coef)

error = yhat - row[-1]

sum\_error += error\*\*2

# Update intercept coefficient

coef[0] = coef[0] - learning\_rate \* error

# Update other coefficients

for i in range(len(train.columns) - 1):

coef[i + 1] = coef[i + 1] - learning\_rate \* error \* row[i]

# Print metrics for each epoch (optional)

# print("epoch=%d, learning\_rate=%.3f, error=%.3f" % (epoch, learning\_rate, sum\_error / m))

return coef

# Read data from CSV

df = pd.read\_csv("Z:\ML\salary\_data.csv")

learning\_rate = 0.001

epochs = 400

# Shuffle the rows for stochasticity

df = df.sample(frac=1).reset\_index(drop=True)

# Perform stochastic gradient descent

coef = sgd(df, learning\_rate, epochs)

print("coefficients:", coef)

# Scatter plot of data points

plt.scatter(df.iloc[:, 0], df.iloc[:, 1], color='blue', label='Data points')

plt.xlabel('X')

plt.ylabel('y')

# Plot the regression line

x\_values = np.linspace(min(df.iloc[:, 0]), max(df.iloc[:, 0]), 100)

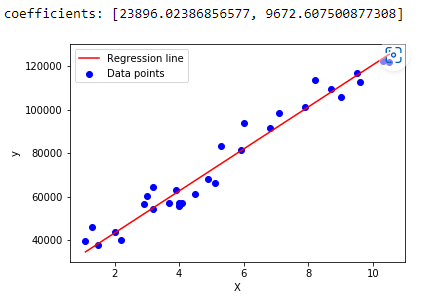
y\_values = coef[0] + coef[1] \* x\_values # Assuming there are two coefficients (intercept and slope)

plt.plot(x\_values, y\_values, color='red', label='Regression line')

plt.legend()

plt.show()

**Output:**



**Aim:** Write a program to implement the linear regression using Batch gradient descent approach of training for a sample training data set stored as a .CSV file.

**Source code:**

import pandas as pd

import matplotlib.pyplot as plt

def predict(row,coefficients):

yhat=coefficients[0]

for i in range(len(row)-1):

yhat+=coefficients[i+1]\*row[i]

return yhat

def bgd(train,learning\_rate,epochs):

coef=[0.0 for i in range(len(train.columns))]

for epoch in range(epochs):

sum\_error=0

for row in range(len(train.index)):

yhat=predict(train.iloc[row,:],coef)

error=yhat-train.iloc[row,:][-1]

sum\_error+=error\*\*2

coef[0]=coef[0]-learning\_rate\*error

for i in range(len(train.columns)-1):

coef[i+1]=coef[i+1]-learning\_rate\*error\*train.iloc[row,i]

#print("epoch=%d,learning\_rate=%.3f,error=%.3f"%(epoch,learning\_rate,sum\_error))

return coef

df=pd.read\_csv("Z:\ML\salary\_data.csv")

learning\_rate=0.001

epochs=40

coef=bgd(df,learning\_rate,epochs)

print("coefficients:",coef)

x=df.iloc[:,0]

y=df.iloc[:,1]

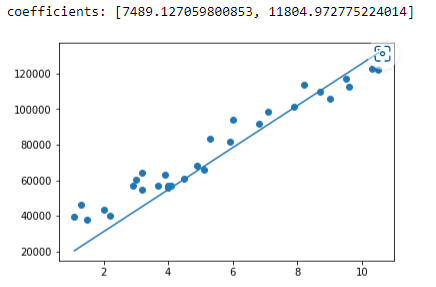
z=coef[0]+coef[1]\*x

plt.scatter(x,y)

plt.plot(x,z)

plt.show()

**Output:**

****

**Aim:** Write a program to implement the Logistic regression for a sample training data set stored as a .CSV file. Compute the performance of the classifier, considering few test data sets.

**Source code:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.simplefilter("ignore")

df=pd.read\_csv('Z:\ML\Iris.csv')

print(df)

df.info()

df.isnull().sum()

df.columns

df=df.drop(columns="Id")

print(df)

df['Species'].value\_counts()

sns.countplot(df['Species'])

X=df.iloc[:,:4]

y=df.iloc[:,4]

print(X)

print(y)

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=9)

print(X\_train.shape)

print(X\_test.shape)

print(y\_train.shape)

print(y\_test.shape)

from sklearn.linear\_model import LogisticRegression

model=LogisticRegression()

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

LogisticRegression()

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

print(y\_pred)

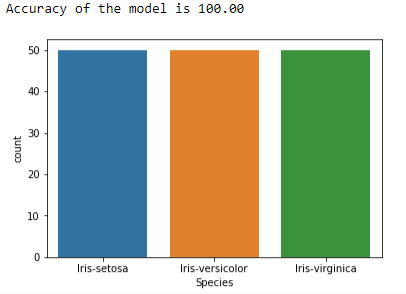
from sklearn.metrics import accuracy\_score,confusion\_matrix

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

accuracy=accuracy\_score(y\_test,y\_pred)\*100

print("Accuracy of the model is {:.2f}".format(accuracy))

**Output:**

****

**Aim:** Linear regression

**Source code:**

import numpy as np

import pandas as pd

import warnings

from sklearn.linear\_model import LinearRegression

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

from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error

import statsmodels.api as sm

warnings.simplefilter("ignore")

housing = pd.read\_csv("Z:\ML\Housing.csv")

#print(housing.head())

varlist = ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning', 'prefarea']

def binary\_map(x):

return x.map({'yes': 1, "no": 0})

housing[varlist] = housing[varlist].apply(binary\_map)

status = pd.get\_dummies(housing['furnishingstatus'])

status = pd.get\_dummies(housing['furnishingstatus'], drop\_first = True)

housing = pd.concat([housing, status], axis = 1)

housing.drop(['furnishingstatus'], axis = 1, inplace = True)

np.random.seed(0)

#print(housing.head())

df\_train, df\_test = train\_test\_split(housing, train\_size = 0.9, test\_size = 0.1, random\_state = 0)

scaler = MinMaxScaler()

num\_vars = ['area', 'bedrooms', 'bathrooms', 'stories', 'parking','price']

df\_train[num\_vars] = scaler.fit\_transform(df\_train[num\_vars])

df\_train

y\_train = df\_train.pop('price')

X\_train = df\_train

X\_train\_lm = sm.add\_constant(X\_train)

#X\_train\_lm

scaler = MinMaxScaler()

num\_vars = ['area', 'bedrooms', 'bathrooms', 'stories', 'parking','price']

df\_test[num\_vars] = scaler.fit\_transform(df\_test[num\_vars])

#df\_test

y\_test = df\_test.pop('price')

X\_test = df\_test

X\_test\_lm = sm.add\_constant(X\_test)

#X\_test\_lm

model = LinearRegression()

model.fit(X\_train\_lm,y\_train)

predictions = model.predict(X\_test\_lm)

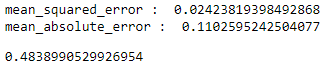
print('mean\_squared\_error : ', mean\_squared\_error(y\_test, predictions))

print('mean\_absolute\_error : ', mean\_absolute\_error(y\_test, predictions))

from sklearn.metrics import r2\_score

r2\_score(y\_test, predictions)

**Output:**

****

**Aim:** Logistic regression – Housing data

**Source code:**

import pandas as pd

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

from sklearn.linear\_model import LogisticRegression

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

import warnings

warnings.simplefilter("ignore")

housing = pd.read\_csv("Z:\ML\Housing.csv")

def binary\_map(x):\

return x.map({'yes':1,'no':0})

varlist = ['mainroad', 'guestroom', 'basement', 'hotwaterheating', 'airconditioning', 'prefarea']

housing[varlist]=housing[varlist].apply(binary\_map)

print(housing.columns)

target\_variable\_name=housing.columns[-1]

features\_to\_keep=housing.columns[:-1]

x = housing[features\_to\_keep]

y = housing[target\_variable\_name]

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

model=LogisticRegression()

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

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

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

classification\_report\_result=classification\_report(y\_test,y\_pred)

print("Accuracy: {:.2f}".format(accuracy))

print("Classification Report: ", classification\_report\_result)

**Output:**

