CATALOG

Teacher's Signature

Q1. *IMPLEMENT SIMPLE LINEAR REGRESSION USING PYTHON OF THE FOLLOWING DATA POINT.*

Code:

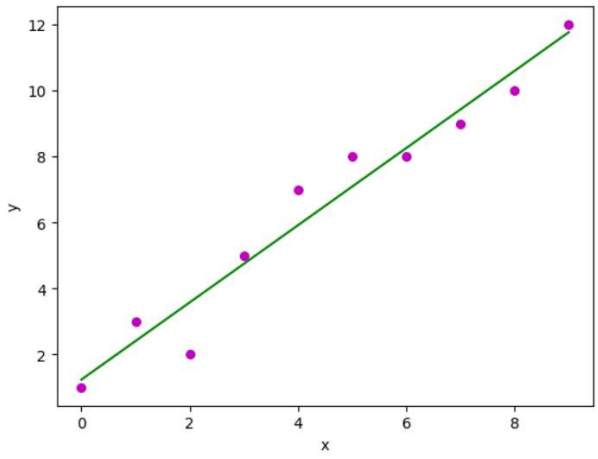
```
import numpy as np
import matplotlib.pyplot as plt
def plot_regression_line(x, y, b):
   plt.scatter(x, y, color="m",marker="o", s=30)
   y_pred = b[0] + b[1]*x
   plt.plot(x, y_pred, color="g")
   plt.xlabel('x')
   plt.ylabel('y')
   plt.show()
x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])
n = np.size(x)
x_mean = np.mean(x)
y_mean = np.mean(y)
Sxy = np.sum(x*y) - n*x_mean*y_mean
Sxx = np.sum(x*x) - n*x_mean*x_mean
b1 = Sxy/Sxx
b0 = y_mean-b1*x_mean
p = float(input("enter the value of x to predict y: "))
y_pred = b1 * p + b0
print("value at", p, "is", y_pred)
plot_regression_line(x, y, [b0, b1])
```

INPUT SET:

X	0	1	2	3	4	5	6	7	8	9
Y	1	3	2	5	7	8	8	9	10	12

OUTPUT SCREEN:

```
enter the value of x to predict y: 10
value at 10.0 is 12.933333333333334
```

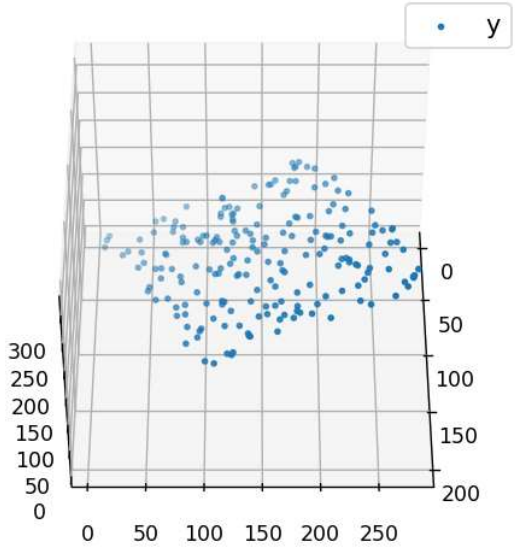


Q2. *IMPLEMENT MULTIPLE REGRESSION MODEL USING RANDOM DATA SET.*

Code:

```
import numpy as np
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
def generate_dataset(n):
    x = []
    y = []
    random_x1 = np.random.rand()
    random_x2 = np.random.rand()
    for i in range(n):
        x1 = i
        x2 = i/2 + np.random.rand()*n
        x.append([1, x1, x2])
        y.append(random_x1 * x1 + random_x2 * x2 + 1)
    return np.array(x), np.array(y)
x, y = generate_dataset(200)
mpl.rcParams['legend.fontsize'] = 12
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
ax.scatter(x[:, 1], x[:, 2], y, label='y', s=5)
ax.legend()
ax.view_init(45, 0)
plt.show()
```

OUTPUT SCREEN:



Q3. ***IMPLEMENT POLYNOMIAL REGRESSION OF THE FOLLOWING DATA.***

Code:

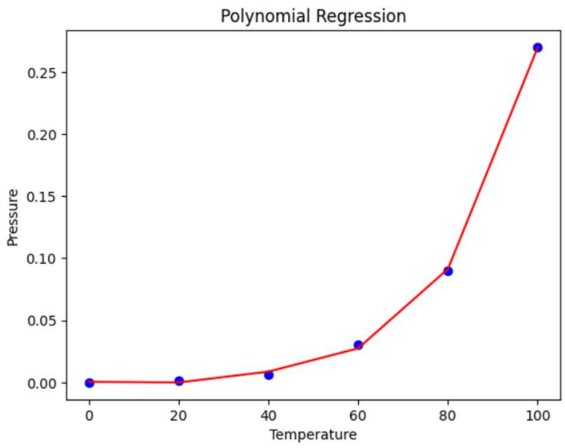
```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
import pandas as pd
import matplotlib.pyplot as plt

datas = pd.read_csv('4.data.csv')
X = datas.iloc[:, 1:2].values
y = datas.iloc[:, 2].values
poly = PolynomialFeatures(degree = 4)
X_poly = poly.fit_transform(X)
poly.fit(X_poly, y)
lin = LinearRegression()
lin.fit(X_poly, y)
p = float(input("enter first independent variable value: "))
predarray = np.array([[p]])
y_pred = lin.predict(poly.fit_transform(predarray))

print("value at", p, "is", y_pred)
plt.scatter(X, y, color = 'blue')
plt.plot(X, lin.predict(poly.fit_transform(X)), color = 'red')
plt.title('Polynomial Regression')
plt.xlabel('Temperature')
plt.ylabel('Pressure')
plt.show()
```

INPUT:

	sno	Tempetarure	Pressure
0	1	0	0.0002
1	2	20	0.0012
2	3	40	0.0060
3	4	60	0.0300
4	5	80	0.0900
5	6	100	0.2700

OUTPUT SCREEN:

```
enter first independent variable value: 110
value at 110.0 is [0.43295877]
```

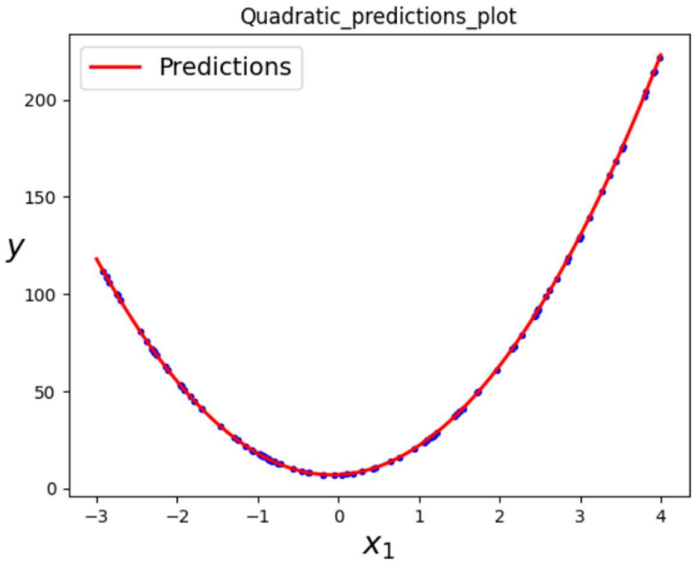


Q4. *IMPLEMENT POLYNOMIAL REGRESSION FOR NON-LINEAR RANDOM DATASET.*

Code:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score
x = np.array(7 * np.random.rand(100, 1) - 3)
x1 = x.reshape(-1, 1)
y = 13 * x*x + 2 * x + 7
regression_model = LinearRegression()
regression_model.fit(x1, y)
y_predicted = regression_model.predict(x1)
poly_features = PolynomialFeatures(degree = 2, include_bias = False)
x_poly = poly_features.fit_transform(x1)
lin_reg = LinearRegression()
lin_reg.fit(x_poly, y)
x_new = np.linspace(-3, 4, 100).reshape(100, 1)
x_new_poly = poly_features.transform(x_new)
y_new = lin_reg.predict(x_new_poly)
plt.plot(x, y, "b.")
plt.plot(x_new, y_new, "r-", linewidth = 2, label ="Predictions")
plt.xlabel("$x_1$", fontsize = 18)
plt.ylabel("$y$", rotation = 0, fontsize = 18)
plt.legend(loc ="upper left", fontsize = 14)
plt.title("Quadratic_predictions_plot")
plt.show()
```

OUTPUT SCREEN:



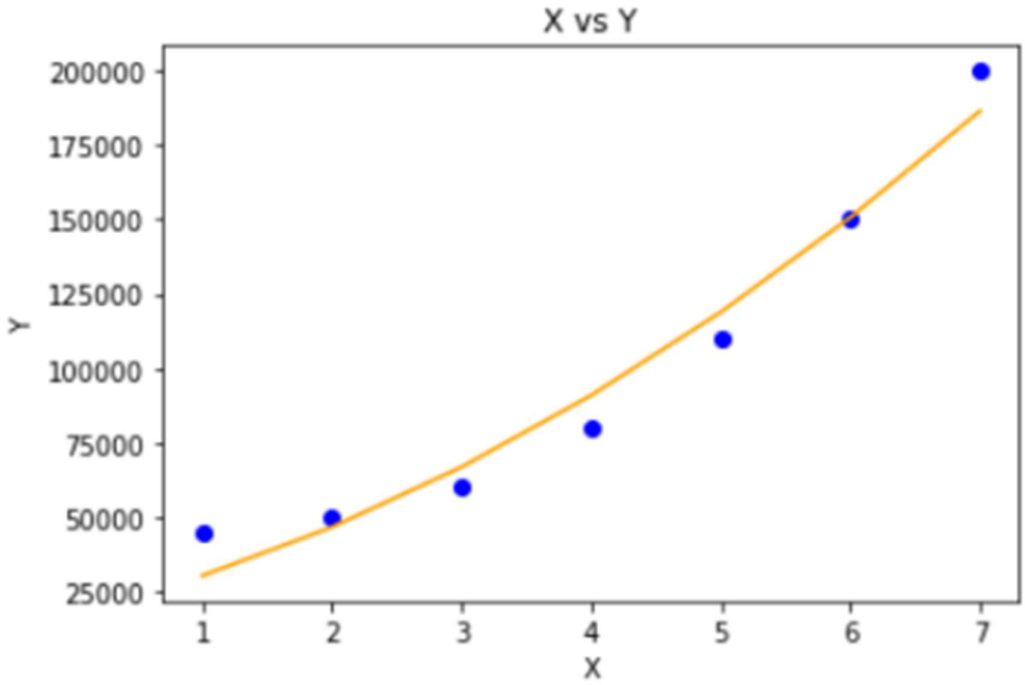
Q5. ***CALCULATE THE PERFORMANCE OF THE MODEL OBTAINED BY POLYNOMIAL REGRESSION USING PYTHON.***

Code:

```
import numpy as np
import math
import matplotlib.pyplot as plt
class PolynomailRegression() :
    def __init__( self, degree, learning_rate, iterations ) :
        self.degree = degree
        self.learning_rate = learning_rate
        self.iterations = iterations
    def transform( self, X ) :
        X_transform = np.ones( ( self.m, 1 ) )
        j = 0
        for j in range( self.degree + 1 ) :
            if j != 0 :
                x_pow = np.power( X, j )
                X_transform = np.append( X_transform, x_pow.reshape( -1, 1 ), axis = 1 )
            return X_transform
    def normalize( self, X ) :
        X[:, 1:] = ( X[:, 1:] - np.mean( X[:, 1:], axis = 0 ) ) / np.std(X[:, 1:], axis = 0 )
        return X
    def fit( self, X, Y ) :
        self.X = X
        self.Y = Y
        self.m, self.n = self.X.shape
        self.W = np.zeros( self.degree + 1 )
        X_transform = self.transform( self.X )
        X_normalize = self.normalize( X_transform )
        for i in range( self.iterations ) :
            h = self.predict( self.X )
            error = h - self.Y
            self.W = self.W - self.learning_rate * ( 1 / self.m ) * np.dot( X_normalize.T, error )
        return self
    def predict( self, X ) :
        X_transform = self.transform( X )
        X_normalize = self.normalize( X_transform )
        return np.dot( X_transform, self.W )
def main() :
 X = np.array( [ [1], [2], [3], [4], [5], [6], [7] ] )
 Y = np.array( [ 45000, 50000, 60000, 80000, 110000, 150000, 200000 ])
 model = PolynomailRegression( degree = 2, learning_rate = 0.01, iterations = 500 )
```

```
model.fit( X, Y )
Y_pred = model.predict( X )
plt.scatter( X, Y, color = 'blue' )
plt.plot( X, Y_pred, color = 'orange' )
plt.title( 'X vs Y' )
plt.xlabel( 'X' )
plt.ylabel( 'Y' )
plt.show()

if __name__ == "__main__" :
   main()
```

OUTPUT SCREEN:



Q6. *IMPLEMENT SIMPLE LINEAR REGRESSION USING PYHTON FOR THE FOLLOWING DATA POINT.*

Code:

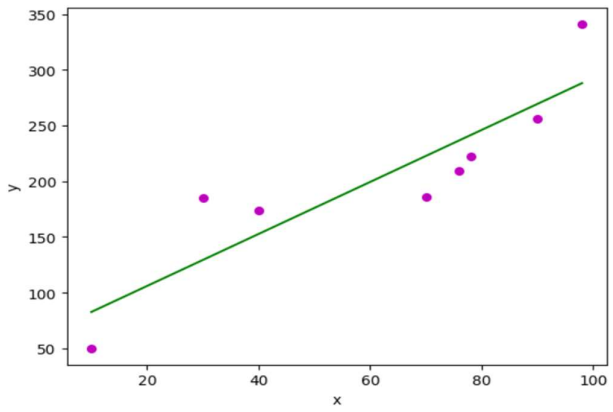
```
import numpy as np
import matplotlib.pyplot as plt
def plot_regression_line(x, y, b):
   plt.scatter(x, y, color="m",
           marker="o", s=30)
   y_pred = b[0] + b[1]*x
   plt.plot(x, y_pred, color="g")
   plt.xlabel('x')
   plt.ylabel('y')
   plt.show()
x = np.array([10, 30, 40, 70, 76, 78, 90, 98])
y = np.array([50, 185, 174, 186, 209, 222, 256, 341])
n = np.size(x)
x_mean = np.mean(x)
y_mean = np.mean(y)
Sxy = np.sum(x*y) - n*x_mean*y_mean
Sxx = np.sum(x*x) - n*x_mean*x_mean
b1 = Sxy/Sxx
b0 = y_mean-b1*x_mean
p = float(input("enter the value of x to predict y: "))
y_pred = b1 * p + b0
print("value at", p, "is", y_pred)
plot_regression_line(x, y, [b0, b1])
```

DATA POINT:

X	10	30	40	70	76	78	90	98
Y	50	185	174	186	209	222	256	341

OUTPUT SCREEN:

```
enter the value of x to predict y: 92
value at 92.0 is 274.0095503967088
```



Q7. *IMPLEMENT USING PYTHON PROGRAM TO LOGISTIC REGRESSION USING PYTHON FOR THE GIVEN DATA SET.*

Code:

```
from sklearn import linear_model
from sklearn.model_selection import train_test_split
import pandas as pd

datas = pd.read_csv('data.csv')
X = datas.iloc[:, 2:4].values
y = datas.iloc[:, 4].values
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.1)

model = linear_model.LogisticRegression()
model.fit(x_train, y_train)

p = float(input("enter first independent variable value: "))
q = float(input("enter second independent variable value: "))
pre = model.predict([[p, q]])
print("predicted value is: ", pre)
```

DATA POINTS:

User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0
15570769	Female	26	80000	0
15606274	Female	26	52000	0
15746139	Male	20	86000	0
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1
15617482	Male	45	26000	1
15704583	Male	46	28000	1

OUTPUT SCREEN:

```
enter first independent variable value: 47
enter second independent variable value: 25000
predicted value is:  [1]
```

Q8. *IMPLEMENT OF KNN ALGORITHM OF THE FLOWING DATASET USING PYTHON.*

Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix

df = pd.read_csv("prostate2.csv")
scaler = StandardScaler()
scaler.fit(df.drop('target', axis=1))
scaled_features = scaler.transform(df.drop('target', axis=1))
df_feat = pd.DataFrame(scaled_features, columns=df.columns[:-1])
X_train, X_test, y_train, y_test = train_test_split(scaled_features, df['target'], test_size=0.30)
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
pred = knn.predict(X_test)
print(confusion_matrix(y_test, pred))
print(classification_report(y_test, pred))
error_rate = []
for i in range(1, 40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
plt.figure(figsize=(10, 6))
plt.plot(range(1, 40), error_rate, color='blue', linestyle='dashed', marker='o', markerfacecolor='red',
markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
plt.show()
```

DATA POINTS:

	lcavol	lweight	age	lbph	svi	lcp	gleason	pgg45	lpsa
0	-0.579818	2.769459	50	-1.386294	0	-1.386294	6	0	-0.430783
1	-0.994252	3.319626	58	-1.386294	0	-1.386294	6	0	-0.162519
2	-0.510826	2.691243	74	-1.386294	0	-1.386294	7	20	-0.162519
3	-1.203973	3.282789	58	-1.386294	0	-1.386294	6	0	-0.162519
4	0.751416	3.432373	62	-1.386294	0	-1.386294	6	0	0.371564

OUTPUT SCREEN:

```
[[22  3]
 [ 2  3]]
              precision    recall  f1-score   support

           0       0.92      0.88      0.90        25
           1       0.50      0.60      0.55         5

    accuracy                           0.83        30
   macro avg       0.71      0.74      0.72        30
weighted avg       0.85      0.83      0.84        30
```