Write a program to demonstrate the working of the Simple Linear Regression.

```
import numpy as np

def simple_linear_regression(x,y):
    n = len(x)
    x_mean = np.mean(x)
    y_mean = np.mean(y)
# Clculate slope (m) & intercept (b) using least suares method
    num = sum((x[i] - x_mean) * (y[i] - y_mean) for i in range(n))
    den = sum((x[i] - x_mean) ** 2 for i in range(n))
    slope = num/den
    intercept = y_mean - slope * x_mean
    return slope,intercept

# Data

x = np.array([1,2,3,4,5])
y = np.array([2,3,4,5,6])
slope,intercept = simple_linear_regression(x,y)
print("Slope(m) :", slope)
print("Intercept(b) :", intercept)
```

```
Slope(m) : 1.0
Intercept(b) : 1.0
```

Ans

Write a program to demonstrate the working of Multiple Linear Regression.

```
Click here to ask Blackbox to help you code faster
import numpy as np
import pandas as pd
import statsmodels.api as sm

# Dataset

data = {
    'X1':[1,2,3,4,5],
    'X2':[2,3,4,5,6],
    'X3':[3,4,5,6,7],
    'Y':[10,20,30,40,50]
}
# Create Dataframe
df = pd.DataFrame(data)

# Define the independent varible X & dependent variable Y
X = df[['X1','X2','X3']]
Y = df['Y']

# Add a constrint
X = sm.add_constant(X)

# Fit the multiple regression
model = sm.OLS(Y , X).fit()

# Print the model summary
print(model.summary())
```

Write a program to demonstrate the working of the K-means Clustering.

```
Click here to ask Blackbox to help you code faster
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
# Generate a synthetic dataset with 3 clusters
X, y = make_blobs(n_samples=300, centers=4, random_state=0, cluster_std=0.60)

# Plot the dataset
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='viridis')

# Initialize KMeans with 3 clusters
kmeans = KMeans(n_clusters=4)

# Fit the model to the data
kmeans.fit(X)

# Get the cluster assignments for each data point
y_pred = kmeans.predict(X)

# Plot the cluster assignments
plt.scatter(X[:, 0], X[:, 1], c=y_pred, s=50, cmap='viridis')

# Plot the cluster centers
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=200, c='black')
plt.title("K-Means Clustering")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.legend()
plt.grid(True)
plt.show()
```

WAP to demonstrate the Usage of Haet Map & Confusion Matrix and it Interpretation

```
Click here to ask Blackbox to help you code faster
# Importing necessary libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
# Load the dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/00225/Indian%20Liver%20Patient%20Dataset%20(ILPD).csv"
column_names = ['Age', 'Gender', 'Total_Bilirubin', 'Direct_Bilirubin', 'Alkaline_Phosphotase','Alamine_Aminotransferase',
'Aspartate_Aminotransferase', 'Total_Protiens', 'Albumin', 'Albumin_and_Globulin_Ratio', 'Liver_disease']
indian_liver_df = pd.read_csv(url, names=column_names)

# Data preprocessing
indian_liver_df['Gender'] = indian_liver_df['Gender'].map({'Female': 0, 'Male': 1})
indian_liver_df['Albumin_and_Globulin_Ratio'].fillna(indian_liver_df['Albumin_and_Globulin_Ratio'].mean(), inplace=True)

# Splitting features and target variable
X = indian_liver_df.drop('Liver_disease', axis=1)
y = indian_liver_df['Liver_disease']

# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# Initialize the KNN classifier
k = 5 # Number of neighbors
knn_classifier = KNeighborsClassifier(n_neighbors=k)

# Train the classifier
knn_classifier.fit(X_train_scaled, y_train)

# Make predictions on the test data
y_pred = knn_classifier.predict(X_test_scaled)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print()
print("Classification Report:")
print(classification_report(y_test, y_pred))
print()
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```