DAY 5 LAB EXPERIMENTS

Name: M.SANTHOSH

Reg.no: 192324014

Dept: B.Tech AI & DS

Scenario: You are a scientist conducting research on rare elements found in a specific region. Your goal is to estimate the average concentration of a rare element in the region using a random sample of measurements. You will use the NumPy library to perform point estimation and calculate confidence intervals for the population mean.The rare element concentration data is stored in a CSV file named "rare_elements.csv," where each row contains a single measurement of the concentration.

Question: Write a Python program that allows the user to input the sample size, confidence level, and desired level of precision.

Solution:

```
import numpy as np

import pandas as pd

from scipy import stats

import math

data = pd.read_csv("/content/rare_elements.csv")

sample_size = int(input("Enter sample size: "))

confidence_level = float(input("Enter confidence level (e.g., 0.95): "))

precision = float(input("Enter desired level of precision (E): "))

sample = data.sample(n=sample_size, random_state=1)

sample_mean = np.mean(sample["concentration"])

sample_std = np.std(sample["concentration"], ddof=1)

z_score = stats.norm.ppf((1 + confidence_level) / 2)

margin_of_error = z_score * (sample_std / math.sqrt(sample_size))

lower_bound = sample_mean - margin_of_error

upper_bound = sample_mean + margin_of_error

print("\n--- Estimation Results ---")

print(f"Sample Mean (Point Estimate): {sample_mean:.3f}")

print(f"Confidence Level: {confidence_level * 100:.1f}%")

print(f"Margin of Error: {margin_of_error:.3f}")

print(f"Confidence Interval: ({lower_bound:.3f}, {upper_bound:.3f})")
```

```
--- Estimation Results ---

Sample Mean (Point Estimate): 50.328

Confidence Level: 99.0%

Margin of Error: 3.486
```

Confidence Interval: (46.842, 53.814)

if margin_of_error <= precision:

  print("✔ Desired precision achieved.")

else:

  print("✖ Desired precision NOT achieved. Increase sample size.")

```
data = pd.read_csv("/content/rare_elements.csv")
sample_size = int(input("Enter sample size: "))
confidence_level = float(input("Enter confidence level (e.g., 0.95): "))
precision = float(input("Enter desired level of precision (E): "))
```

```
Enter sample size: 10
Enter confidence level (e.g., 0.95): 0.99
Enter desired level of precision (E): 80
```

```
print("\n--- Estimation Results ---")
print(f"Sample Mean (Point Estimate): {sample_mean:.3f}")
print(f"Confidence Level: {confidence_level * 100:.1f}%")
print(f"Margin of Error: {margin_of_error:.3f}")
print(f"Confidence Interval: ({lower_bound:.3f}, {upper_bound:.3f})")
```

```
--- Estimation Results ---
Sample Mean (Point Estimate): 50.328
Confidence Level: 99.0%
Margin of Error: 3.486
Confidence Interval: (46.842, 53.814)
```

```
if margin_of_error <= precision:
    print("✓ Desired precision achieved.")
else:
    print("✗ Desired precision NOT achieved. Increase sample size.")
```

```
✓ Desired precision achieved.
```

Scenario: Imagine you are an analyst for a popular online shopping website. Your task is to analyze customer reviews and provide insights on the average rating and customer satisfaction level for a specific product category.

Question: You will use the pandas library to calculate confidence intervals to estimate the true population mean rating. You have been provided with a CSV file named "customer_reviews.csv," which contains customer ratings for products in the chosen category.

Solution:

```
import pandas as pd

import numpy as np

from scipy import stats

import math

df = pd.read_csv("/content/customer_reviews.csv")

confidence_level = 0.95

sample_mean = df["rating"].mean()

sample_std = df["rating"].std()

sample_size = df["rating"].count()

z_score = stats.norm.ppf((1 + confidence_level) / 2)

margin_of_error = z_score * (sample_std / math.sqrt(sample_size))

lower_bound = sample_mean - margin_of_error

upper_bound = sample_mean + margin_of_error

print("Average Rating (Point Estimate):", round(sample_mean, 2))

print("Confidence Level:", confidence_level * 100, "%")

print("Confidence Interval:", (round(lower_bound, 2), round(upper_bound, 2)))
```

```
print("Average Rating (Point Estimate):", round(sample_mean, 2))
print("Confidence Level:", confidence_level * 100, "%")
print("Confidence Interval:", (round(lower_bound, 2), round(upper_bound, 2)))
```

```
Average Rating (Point Estimate): 4.27
Confidence Level: 95.0 %
Confidence Interval: (np.float64(4.02), np.float64(4.51))
```

Question: You are a real estate analyst trying to predict housing prices based on various features of the houses, such as area, number of bedrooms, and location. You have collected a dataset of houses with their respective prices. Write a Python program that allows the user to input the features (area, number of bedrooms, etc.) of a new house. The program should use linear regression from scikit-learn to predict the price of the new house based on the input features.

Solution:

```
import pandas as pd

from sklearn.linear_model import LinearRegression

df = pd.read_excel("House_Prediction.xlsx")

X = df[["beds", "baths", "size", "zip_code"]]

y = df["price"]

model = LinearRegression()

model.fit(X, y)

beds = float(input("Number of bedrooms: "))

baths = float(input("Number of bathrooms: "))

size = float(input("House size (in sq ft): "))

zip_code = float(input("ZIP code: "))

new_house = pd.DataFrame([[beds, baths, size, zip_code]],

                         columns=["beds", "baths", "size", "zip_code"])

predicted_price = model.predict(new_house)

print("Predicted House Price:", round(predicted_price[0], 2))
```

```
beds = float(input("Number of bedrooms: "))
size = float(input("House size (in sq ft): "))
zip_code = float(input("ZIP code: "))
```

```
Number of bedrooms: 3
House size (in sq ft): 2000
ZIP code: 98144
```

```
new_house = pd.DataFrame([[beds, size, zip_code]],
                         columns=["beds", "size", "zip_code"])
predicted_price = model.predict(new_house)
```

```
print("Predicted House Price:", round(predicted_price[0], 2))
```

```
Predicted House Price: 1032902.3
```

Question: You are working for a telecommunications company, and you want to predict whether a customer will churn (leave the company) based on their usage patterns and demographic data. You have collected a dataset of past customers with their churn status (0 for not churned, 1 for churned) and various features. Write a Python program that allows the user to input the features (e.g., usage minutes, contract duration) of a new customer. The program should use logistic regression from scikit-learn to predict whether the new customer will churn or not based on the input features.

Solution:

```
import pandas as pd

from sklearn.linear_model import LogisticRegression

df = pd.read_csv("customer_churn.csv")

X = df[["usage_minutes", "contract_months", "monthly_charges"]]

y = df["churn"]

model = LogisticRegression()

model.fit(X, y)

usage = float(input("Enter usage minutes: "))

contract = float(input("Enter contract duration (months): "))

charges = float(input("Enter monthly charges: "))

new_customer = pd.DataFrame([[usage, contract, charges]],

                columns=["usage_minutes", "contract_months", "monthly_charges"])

prediction = model.predict(new_customer)

if prediction[0] == 1:

   print("Customer is likely to churn")

else:

   print("Customer is not likely to churn")
```

```
usage = float(input("Enter usage minutes: "))
contract = float(input("Enter contract duration (months): "))
charges = float(input("Enter monthly charges: "))

Enter usage minutes: 20
Enter contract duration (months): 4
Enter monthly charges: 500

new_customer = pd.DataFrame([[usage, contract, charges]],
                            columns=["usage_minutes", "contract_months", "monthly_charges"])

prediction = model.predict(new_customer)

if prediction[0] == 1:
    print("Customer is likely to churn")
else:
    print("Customer is not likely to churn")

Customer is likely to churn
```

Scenario: You work as a data scientist for an automobile company that sells various car models. The company has collected data on different car attributes, such as engine size,horsepower, fuel efficiency, and more, along with their corresponding prices. The marketing team wants to build a predictive model to estimate the price of cars based on their features.

Question: Your task is write a Python program that perform linear regression modeling to predict car prices based on a selected set of features from the dataset. Additionally, you need to evaluate the model's performance and provide insights to the marketing team to understand the most influential factors affecting car prices.

Solution:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import r2_score, mean_absolute_error


df = pd.read_csv("Fuel_Efficiency datasets.csv")


X = df[["cylinders", "displacement", "highway_mpg", "year"]]

y = df["city_mpg"]


X_train, X_test, y_train, y_test = train_test_split(

   X, y, test_size=0.2, random_state=42

)


model = LinearRegression()

model.fit(X_train, y_train)


y_pred = model.predict(X_test)


print("R2 Score:", round(r2_score(y_test, y_pred), 3))

print("Mean Absolute Error:", round(mean_absolute_error(y_test, y_pred), 2))


cylinders = float(input("Enter cylinders: "))

displacement = float(input("Enter displacement: "))

highway_mpg = float(input("Enter highway mpg: "))
```

```
year = int(input("Enter model year: "))


new_car = pd.DataFrame(
    [[cylinders, displacement, highway_mpg, year]],
    columns=["cylinders", "displacement", "highway_mpg", "year"]
)


prediction = model.predict(new_car)
print("Predicted City MPG:", round(prediction[0], 2))
```

```
print("R2 Score:", round(r2_score(y_test, y_pred), 3))
print("Mean Absolute Error:", round(mean_absolute_error(y_test, y_pred), 2))
```

```
R2 Score: -1.902
Mean Absolute Error: 0.69
```

```
cylinders = float(input("Enter cylinders: "))
displacement = float(input("Enter displacement: "))
highway_mpg = float(input("Enter highway mpg: "))
year = int(input("Enter model year: "))
new_car = pd.DataFrame(
    [[cylinders, displacement, highway_mpg, year]],
    columns=["cylinders", "displacement", "highway_mpg", "year"]
)
```

```
Enter cylinders: 4
Enter displacement: 2
Enter highway mpg: 37
Enter model year: 2021
```

```
prediction = model.predict(new_car)
print("Predicted City MPG:", round(prediction[0], 2))
```

```
Predicted City MPG: 27.71
```