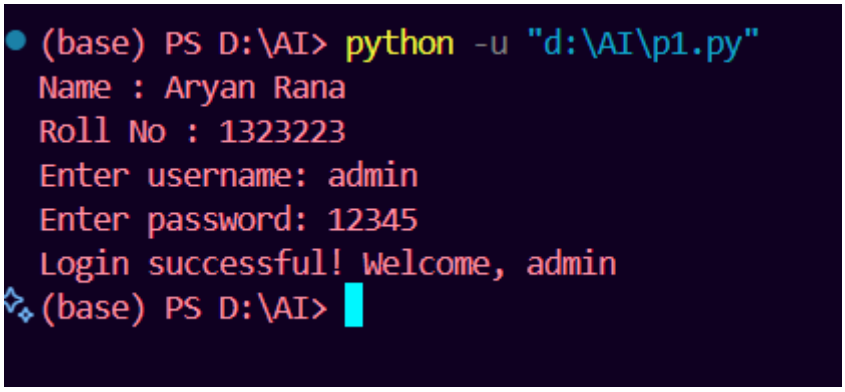
Practical 1 :

Write a Python program that checks if the entered username and password match predefined values.

```
print("Name : Aryan Rana \n Roll No : 1323223")

USERNAME = "admin"

PASSWORD = "12345"

entered_username = input("Enter username: ")

entered_password = input("Enter password: ")

if entered_username.lower() == USERNAME and entered_password.lower()
== PASSWORD:

    print(" Login successful! Welcome,", entered_username)

else:

    print(" Invalid username or password. Please try again.")
```

OUTPUT



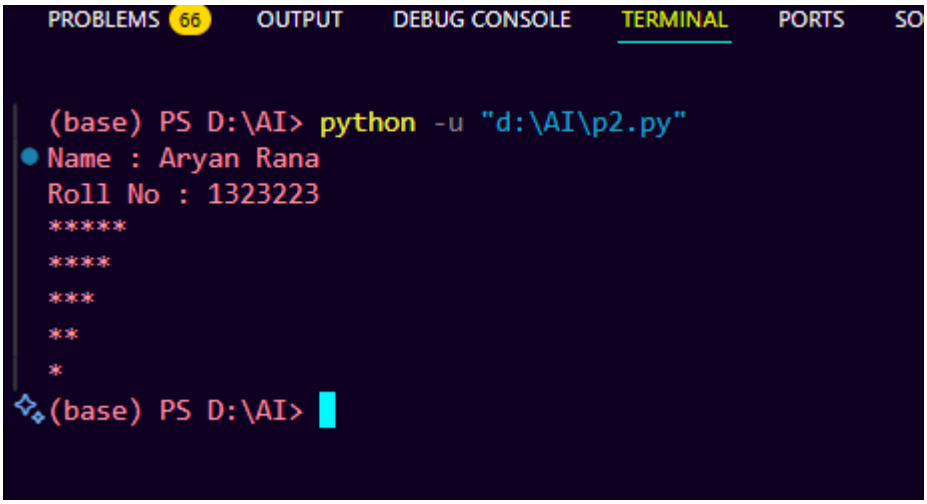
PRACTICAL 2 :

Write a program to print the given inverted right-angled triangle star pattern.

```
*****
****
***
**
*
```

```
print("Name : Aryan Rana \n Roll No : 1323223")

rows = 5  # Number of rows

for i in range(rows, 0, -1):

    print('*' * i)
```

OUTPUT

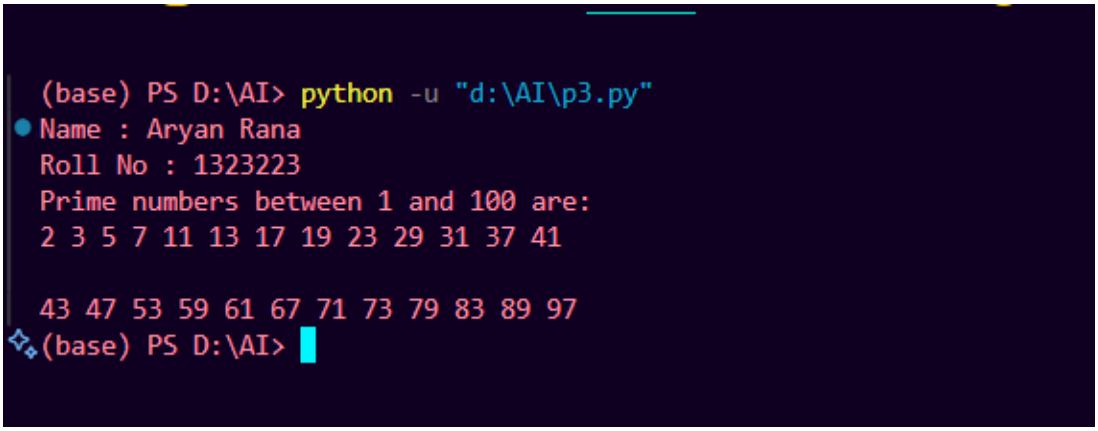


PRACTICAL 3 :

Write a program to print all prime numbers between 1 and 100.

```
print("Name : Aryan Rana \n Roll No : 1323223")
print("Prime numbers between 1 and 100 are:")
for num in range(2, 101):
    is_prime = True
    for i in range(2, int(num ** 0.5) + 1):
        if num % i == 0:
            is_prime = False
            break
    if is_prime:
        print(num, end=" ")
    if num == 41 :
        print("\n")
```

OUTPUT

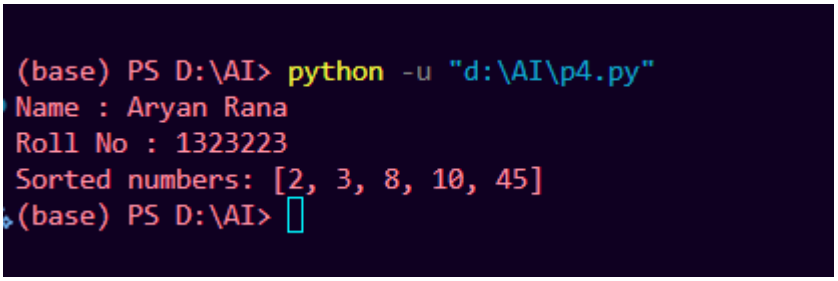


PRACTICAL 4 :

Convert a list of string numbers into integers and sort them in ascending order without using built-in sort functions.

```
print("Name : Aryan Rana \n Roll No : 1323223")
string_numbers = ["10", "3", "45", "2", "8"]
int_numbers = [int(num) for num in string_numbers]
n = len(int_numbers)
for i in range(n):
    for j in range(0, n - i - 1):
        if int_numbers[j] > int_numbers[j + 1]:
            int_numbers[j], int_numbers[j + 1] = int_numbers[j + 1], int_numbers[j]
print("Sorted numbers:", int_numbers)
```

OUTPUT

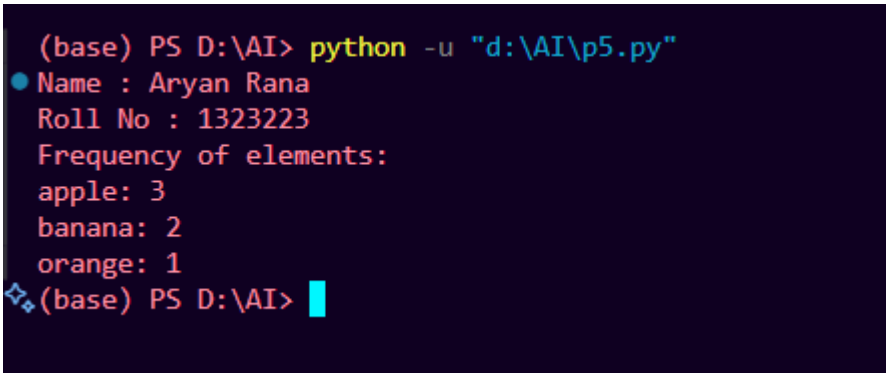


PRACTICAL 5 :

Use a dictionary to count the frequency of elements in a list.

```
print("Name : Aryan Rana \n Roll No : 1323223")
my_list = ["apple", "banana", "apple", "orange", "banana", "apple"]
frequency = {}
for item in my_list:
    if item in frequency:
        frequency[item] += 1
    else:
        frequency[item] = 1
print("Frequency of elements:")
for key, value in frequency.items():
    print(f"{key}: {value}")
```

OUTPUT

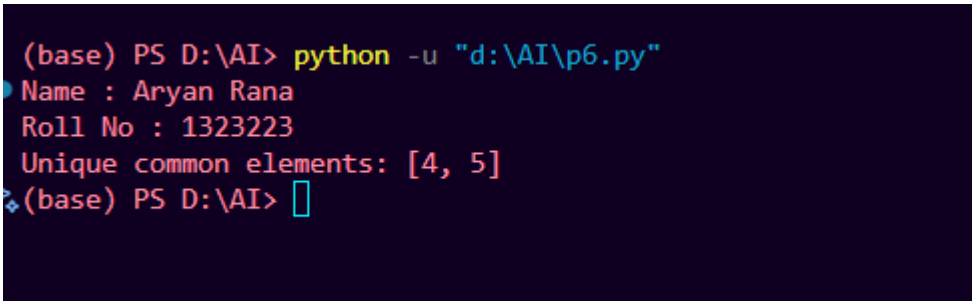


PRACTICAL 6 :

 # Two sample lists

```
print("Name : Aryan Rana \n Roll No : 1323223")
list1 = [1, 2, 3, 4, 5, 5, 2]
list2 = [4, 5, 6, 7, 5, 4]
common_elements = list(set(list1) & set(list2))
print("Unique common elements:", common_elements)
```

OUTPUT

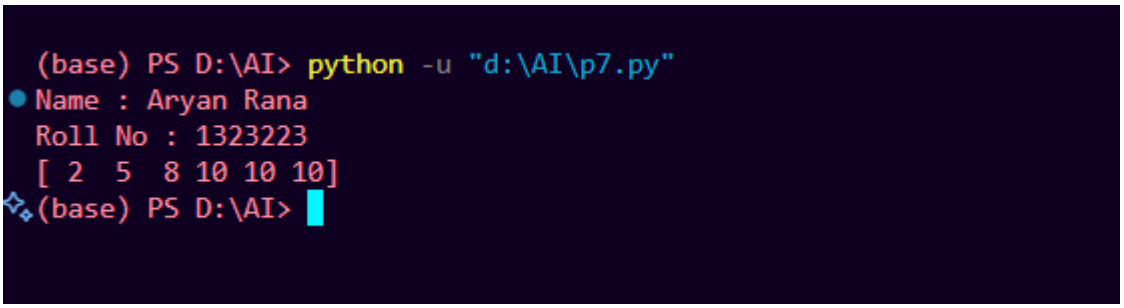


PRACTICAL 7 :

Replace all elements in a NumPy array greater than a threshold with that threshold

```
import numpy as np

print("Name : Aryan Rana \n Roll No : 1323223")

arr = np.array([2, 5, 8, 10, 12, 15])

threshold = 10

arr = np.where(arr > threshold, threshold, arr)

print(arr)
```

OUTPUT



PRACTICAL 8 :

Replace missing values in a Pandas DataFrame with the mean of their columns.

```
import pandas as pd

import numpy as np

print("Name : Aryan Rana \n Roll No : 1323223")

data = {

   'A': [1, 2, np.nan, 4],

   'B': [5, np.nan, 7, 8],

   'C': [np.nan, 11, 12, 13]

}

df = pd.DataFrame(data)

print("Before filling missing values:\n", df)

df.fillna(df.mean(numeric_only=True), inplace=True)

print("\nAfter filling missing values:\n", df)
```

OUTPUT

```
(base) PS D:\AI> python -u "d:\AI\p8.py"
Name : Aryan Rana
Roll No : 1323223
Before filling missing values:
     A    B     C
0  1.0  5.0   NaN
1  2.0  NaN  11.0
2  NaN  7.0  12.0
3  4.0  8.0  13.0

After filling missing values:
          A         B     C
0  1.000000  5.000000  12.0
1  2.000000  6.666667  11.0
2  2.333333  7.000000  12.0
3  4.000000  8.000000  13.0
(base) PS D:\AI>
```

PRACTICAL 9 :

Load the Myntra dataset, perform **info, shape, describe,** and check for null values and calculate Total revenue and discount.

```
import pandas as pd
print("Name : Aryan Rana \n Roll No : 1323223")
df = pd.read_csv("myntra.csv")
print(df.head(1), "\n")
print("=== Dataset Info ===")
print(df.info(), "\n")
print("=== Shape (Rows, Columns) ===")
print(df.shape, "\n")
print("=== Statistical Summary ===")
print(df.describe(), "\n")
print("=== Null Values ===")
print(df.isnull().sum(), "\n")
df["Revenue"] = df["discounted_price"]
df["Discount"] = df["marked_price"] - df["discounted_price"]
total_revenue = df["Revenue"].sum()
total_discount = df["Discount"].sum()
print("=== Totals ===")
print(f" Total Revenue: ₹{total_revenue:,.2f}")
print(f" Total Discount Given: ₹{total_discount:,.2f}")
```

OUTPUT

```
(base) PS D:\AI> python -u "d:\AI\p9.py"
Name : Aryan Rana
Roll No : 1323223
                   product_name  ...                                       product_link
0  Croc Textured Two Fold Wallet  ...  wallets/lino-perros/lino-perros-women-peach-co...

[1 rows x 7 columns]

=== Dataset Info ===
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 168029 entries, 0 to 168028
Data columns (total 7 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   product_name     168029 non-null  object
 1   brand_name       168029 non-null  object
 2   rating           168029 non-null  float64
 3   rating_count     168029 non-null  int64
 4   marked_price     168029 non-null  int64
 5   discounted_price 168029 non-null  int64
 6   product_link     168029 non-null  object
dtypes: float64(1), int64(3), object(3)
memory usage: 9.0+ MB
None

=== Shape (Rows, Columns) ===
(168029, 7)

=== Statistical Summary ===
              rating   rating_count  marked_price  discounted_price
count  168029.000000  168029.000000  168029.00000     168029.000000
mean        2.264556     114.668658    2509.07457       1515.168757
std         2.101402     800.207065    2402.10918       1800.460291
min         0.000000       0.000000      50.00000         49.000000
25%         0.000000       0.000000    1295.00000        674.000000
50%         3.500000       5.000000    1998.00000        999.000000
75%         4.300000      31.000000    2999.00000       1739.000000
max         5.000000   55900.000000  113999.00000      45900.000000

=== Null Values ===
product_name        0
brand_name          0
rating              0
rating_count        0
marked_price        0
discounted_price    0
product_link        0
dtype: int64

=== Totals ===
💰 Total Revenue: ₹254,592,291.00
😲 Total Discount Given: ₹167,005,000.00
(base) PS D:\AI>
```

PRACTICAL 10 :

From the Myntra dataset, filter products priced above 2000 and belonging to the category “tshirts”.

```
import pandas as pd

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

df['category'] = df['product_link'].apply(lambda x: x.split('/')[0])

filtered_df = df[(df['discounted_price'] > 2000) & (df['category'] == 'tshirts')]

print(filtered_df)
```

OUTPUT

```
(base) PS D:\AI> python -u "d:\AI\p10.py"
Name : Aryan Rana
Roll No : 1323223
                        product_name        brand_name  rating  ...  discounted_price                                       product_link  category
80                   Men Solid T-shirt         MANGO MAN     0.0  ...              2390  tshirts/mango-man/mango-man-blue-solid-t-shirt...   tshirts
486     Men Printed Polo Collar T-shirt    Louis Philippe     0.0  ...              2499  tshirts/louis-philippe/louis-philippe-men-gree...   tshirts
936      Men Abstract Print Pure Cotton             Levis     4.1  ...              2209  tshirts/levis/levis-men-white--beige-abstract-...   tshirts
1085          Slim Fit HEAT.RDY T-shirt            ADIDAS     0.0  ...              3059  tshirts/adidas/adidas-men-white-brand-logo-pri...   tshirts
1199    Men Striped Polo Collar T-shirt        Van Heusen     0.0  ...              2124  tshirts/van-heusen/van-heusen-men-navy-blue-st...   tshirts
...                                 ...               ...     ...  ...               ...                                                ...       ...
166559  Men Striped Polo Collar T-shirt  ORIGIN BY ZALORA     0.0  ...              2799  tshirts/origin-by-zalora/origin-by-zalora-men-...   tshirts
166590  Men Striped Polo Collar T-shirt  ORIGIN BY ZALORA     0.0  ...              2869  tshirts/origin-by-zalora/origin-by-zalora-men-...   tshirts
167278         Men Polo Collar T-shirt  ORIGIN BY ZALORA     0.0  ...              2729  tshirts/origin-by-zalora/origin-by-zalora-men-...   tshirts
167312      Men Mandarin Collar T-shirt     ZALORA ACTIVE     0.0  ...              2174  tshirts/zalora-active/zalora-active-men-black-...   tshirts
167779  Men Striped Polo Collar T-shirt  ORIGIN BY ZALORA     0.0  ...              2799  tshirts/origin-by-zalora/origin-by-zalora-men-...   tshirts

[774 rows x 8 columns]
(base) PS D:\AI>
```

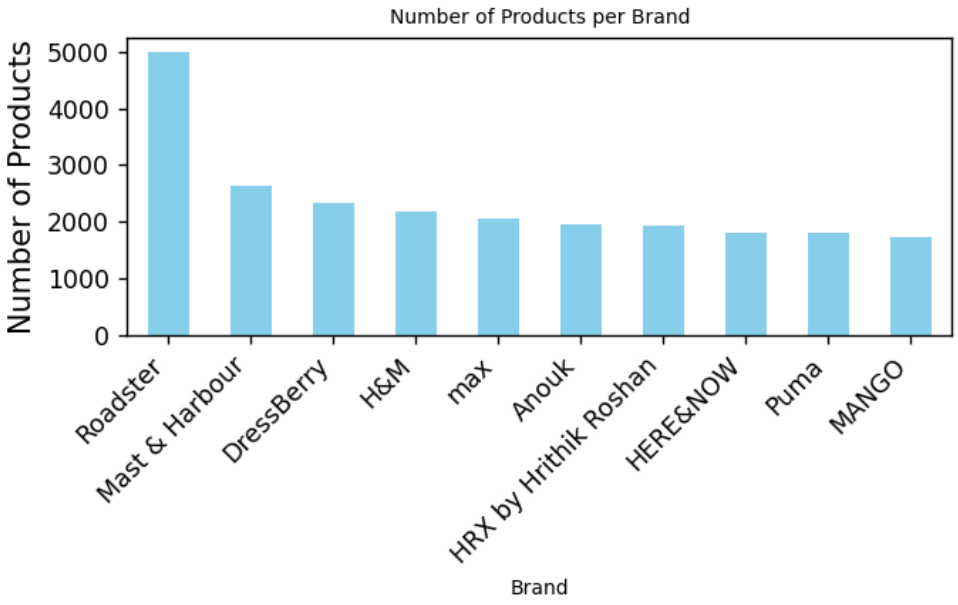
PRACTICAL 11 :

Plot a bar chart showing the number of products per brand

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("myntra.csv")
brand_counts = df['brand_name'].value_counts()
plt.figure(figsize=(12, 6))
brand_counts.plot(kind='bar', color='skyblue')
plt.title('Number of Products per Brand', fontsize=16)
plt.xlabel('Brand', fontsize=12)
plt.ylabel('Number of Products', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

OUTPUT

```
(base) PS D:\AI> python -u "d:\AI\p11.py"
Name : Aryan Rana
Roll No : 1323223
(base) PS D:\AI>
```



PRACTICAL 12 :

Find the average price of products grouped by category.

```
import pandas as pd
df = pd.read_csv("myntra.csv")
df['category'] = df['product_link'].apply(lambda x: x.split('/')[0])
avg_price = df.groupby('category')['discounted_price'].mean()
print("Average price of products per category:\n")
print(avg_price)
```

OUTPUT

PRACTICAL 13:

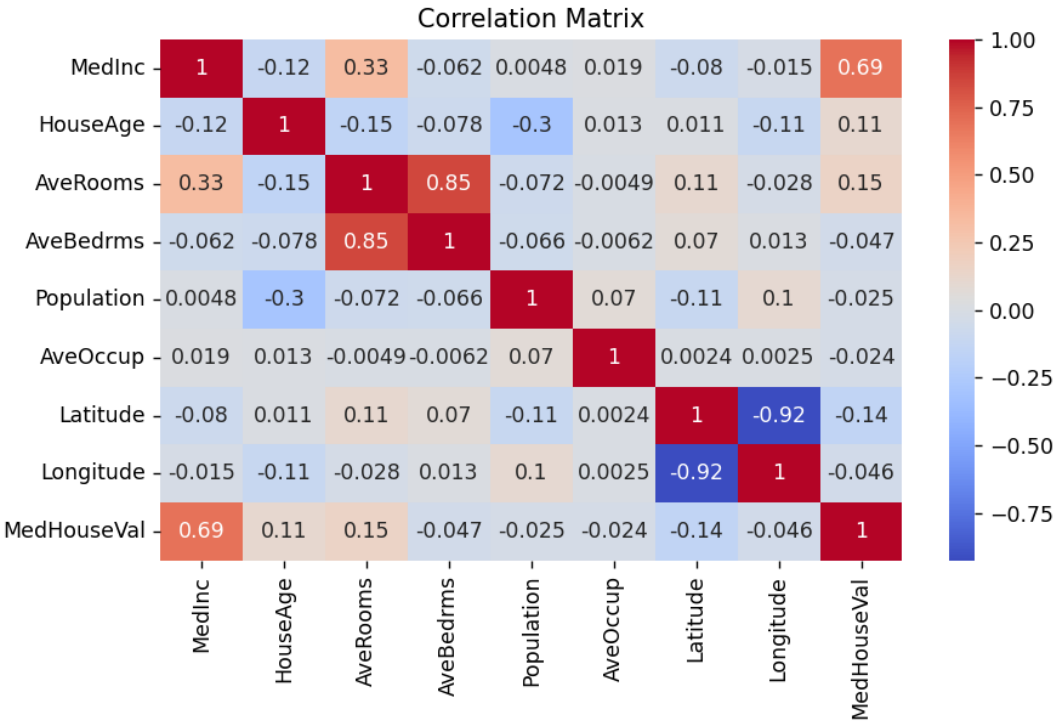
Load the Housing dataset, clean it, visualize correlation, encode categorical values, split (70:30), train Linear Regression, evaluate with R^2 and MSE..

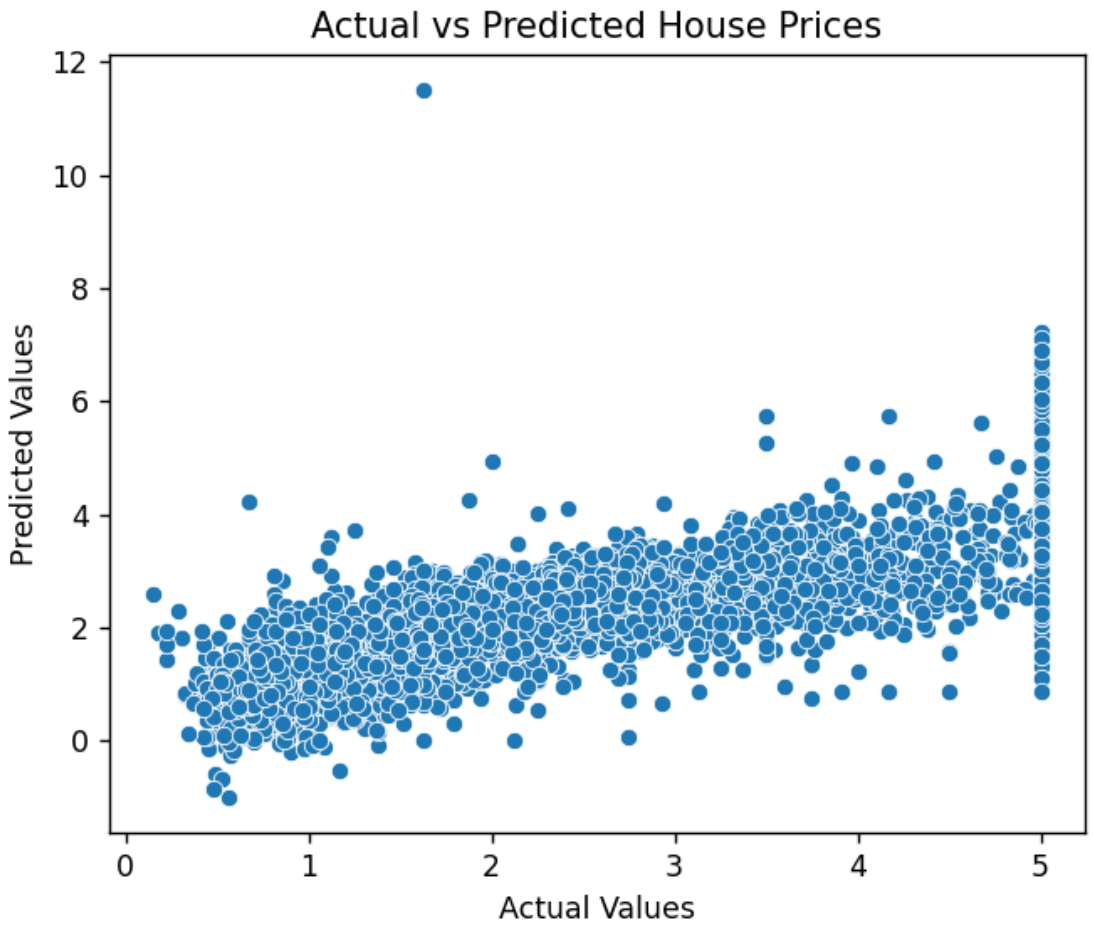
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
print("Name : Aryan Rana \nRoll No : 1323223")
housing = fetch_california_housing(as_frame=True)
df = housing.frame
print("Shape:", df.shape)
print(df.head())
print("\nMissing values:\n", df.isnull().sum())
df = df.drop_duplicates()
plt.figure(figsize=(10, 6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()
print("\nData types:\n", df.dtypes)
X = df.drop('MedHouseVal', axis=1)
y = df['MedHouseVal']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
print("\nModel Performance:")
print(f"R² Score: {r2:.4f}")
print(f"Mean Squared Error: {mse:.4f}")
plt.figure(figsize=(6, 6))
sns.scatterplot(x=y_test, y=y_pred)
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Actual vs Predicted House Prices")
plt.show()
```

OUTPUT

```
(base) PS D:\AI> python -u "d:\AI\p13.py"
Name : Aryan Rana
Roll No : 1323223
Shape: (20640, 9)
   MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  Longitude  MedHouseVal
0  8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88    -122.23        4.526
1  8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86    -122.22        3.585
2  7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85    -122.24        3.521
3  5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85    -122.25        3.413
4  3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85    -122.25        3.422

Missing values:
 MedInc         0
HouseAge        0
AveRooms        0
AveBedrms       0
Population      0
AveOccup        0
Latitude        0
Longitude       0
MedHouseVal     0
dtype: int64
```


Correlation Matrix
MedInc 1 -0.12 0.33 -0.062 0.0048 0.019 -0.08 -0.015 0.69
HouseAge -0.12 1 -0.15 -0.078 -0.3 0.013 0.011 -0.11 0.11
AveRooms 0.33 -0.15 1 0.85 -0.072 -0.0049 0.11 -0.028 0.15
AveBedrms -0.062 -0.078 0.85 1 -0.066 -0.0062 0.07 0.013 -0.047
Population 0.0048 -0.3 -0.072 -0.066 1 0.07 -0.11 0.1 -0.025
AveOccup 0.019 0.013 -0.0049 -0.0062 0.07 1 0.0024 0.0025 -0.024
Latitude -0.08 0.011 0.11 0.07 -0.11 0.0024 1 -0.92 -0.14
Longitude -0.015 -0.11 -0.028 0.013 0.1 0.0025 -0.92 1 -0.046
MedHouseVal 0.69 0.11 0.15 -0.047 -0.025 -0.024 -0.14 -0.046 1
MedInc
HouseAge
AveRooms
AveBedrms
Population
AveOccup
Latitude
Longitude
MedHouseVal
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75


Actual vs Predicted House Prices
Predicted Values
Actual Values
0
2
4
6
8
10
12
0
1
2
3
4
5

PRACTICAL 14:

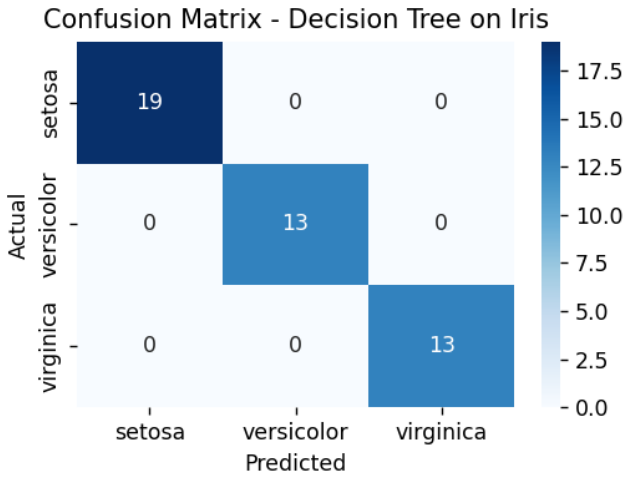
Train a Decision Tree classifier on the Iris dataset and evaluate **accuracy.**

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
print("Name : Aryan Rana \nRoll No : 1323223")
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['target'] = iris.target
print("Dataset shape:", df.shape)
print(df.head())
X = df.drop('target', axis=1)
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"\n✅ Model Accuracy: {accuracy:.4f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=iris.target_names))
cm = confusion_matrix(y_test, y_pred)
```

```
plt.figure(figsize=(5, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=iris.target_names,
            yticklabels=iris.target_names)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Decision Tree on Iris")
plt.show()
```

OUTPUT

```
(base) PS D:\AI> python -u "d:\AI\p14.py"
Name : Aryan Rana
Roll No : 1323223
Dataset shape: (150, 5)
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  target
0                5.1               3.5                1.4               0.2       0
1                4.9               3.0                1.4               0.2       0
2                4.7               3.2                1.3               0.2       0
3                4.6               3.1                1.5               0.2       0
4                5.0               3.6                1.4               0.2       0

✅ Model Accuracy: 1.0000

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        19
  versicolor       1.00      1.00      1.00        13
   virginica       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45

(base) PS D:\AI>
```



PRACTICAL 15:

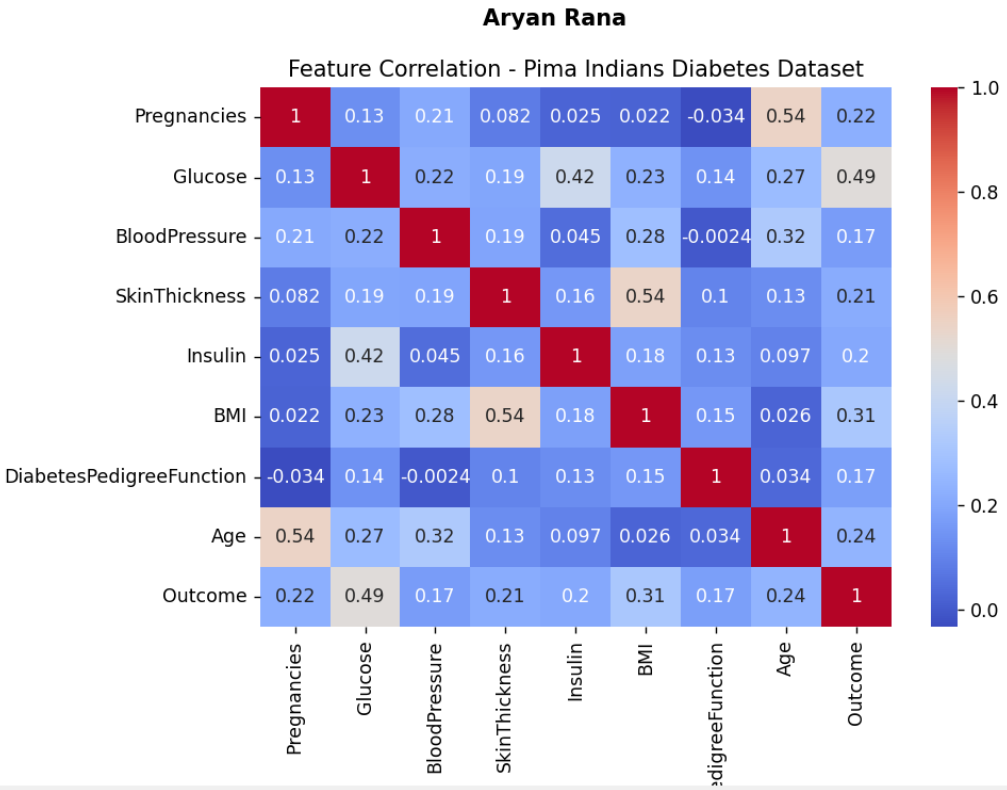
Train a **Random Forest** on Pima Indian Diabetes dataset and compare results with other algorithms using **LazyPredict.**

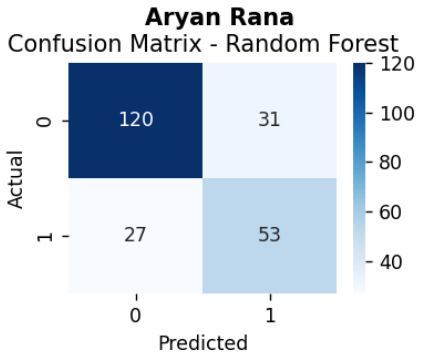
```
#  Import libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from lazypredict.Supervised import LazyClassifier

print("Name : Aryan Rana \nRoll No : 1323223")
df = pd.read_csv("diabetes.csv")  # <-- ensure data.csv is in the same folder
print("Dataset shape:", df.shape)
print(df.head())
# Check for missing values
print("\nMissing values before cleaning:\n", df.isnull().sum())

# Replace 0 with NaN in certain medical features (invalid values)
cols_with_zero = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']
df[cols_with_zero] = df[cols_with_zero].replace(0, np.nan)

# Fill missing with median values
df.fillna(df.median(), inplace=True)
print("\nMissing values after cleaning:\n", df.isnull().sum())
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
```

```
plt.title("Feature Correlation - Pima Indians Diabetes Dataset")
plt.suptitle("Aryan Rana", fontsize=12, fontweight='bold')
plt.show()
X = df.drop('Outcome', axis=1)
y = df['Outcome']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"\n Random Forest Accuracy: {accuracy:.4f}")
print("\nClassification Report:\n", classification_report(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.suptitle("Aryan Rana", fontsize=12, fontweight='bold')
plt.title('Confusion Matrix - Random Forest')
plt.show()
clf = LazyClassifier(verbose=0, ignore_warnings=True, custom_metric=None)
models, predictions = clf.fit(X_train, X_test, y_train, y_test)
print("\nModel Comparison Results:\n")
print(models.head(10))  # Show top 10 models
```

OUTPUT

```
(base) PS D:\AI> python -u "d:\AI\p15.py"
Name : Aryan Rana
Roll No : 1323223
Dataset shape: (768, 9)
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  DiabetesPedigreeFunction  Age  Outcome
0            6      148             72             35        0 33.60                     0.63   50        1
1            1       85             66             29        0 26.60                     0.35   31        0
2            8      183             64              0        0 23.30                     0.67   32        1
3            1       89             66             23       94 28.10                     0.17   21        0
4            0      137             40             35      168 43.10                     2.29   33        1

Missing values before cleaning:
 Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64

Missing values after cleaning:
 Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64

✅ Random Forest Accuracy: 0.7489

Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.79      0.81       151
           1       0.63      0.66      0.65        80
```



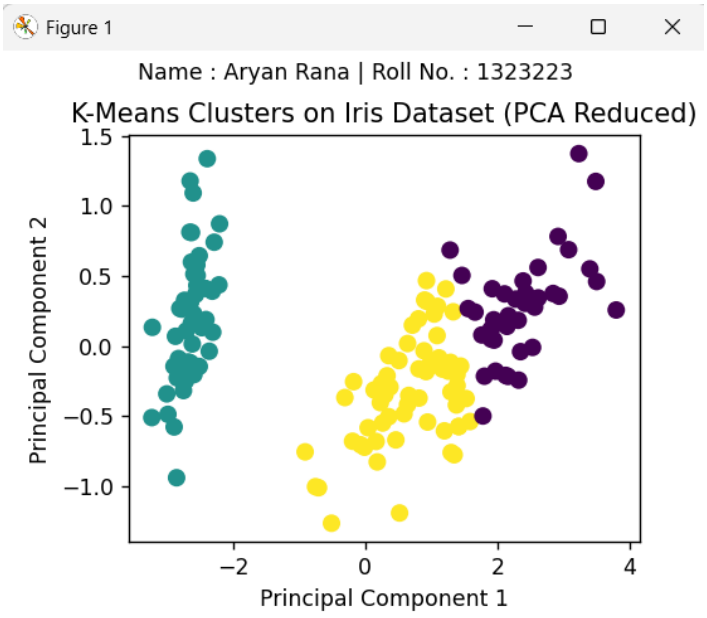



```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
100%|                                                                                | 32/32 [00:08<00:00,  3.78it/s]

Model Comparison Results:

                               Accuracy  Balanced Accuracy  ROC AUC  F1 Score  Time Taken
Model
RandomForestClassifier             0.75               0.73     0.73      0.75        0.24
AdaBoostClassifier                 0.76               0.73     0.73      0.76        0.19
LGBMClassifier                     0.74               0.73     0.73      0.74        6.82
BaggingClassifier                  0.76               0.73     0.73      0.76        0.09
GaussianNB                         0.74               0.72     0.72      0.74        0.02
XGBClassifier                      0.73               0.71     0.71      0.73        0.35
QuadraticDiscriminantAnalysis      0.75               0.71     0.71      0.75        0.02
CalibratedClassifierCV             0.74               0.71     0.71      0.74        0.08
RidgeClassifier                    0.74               0.71     0.71      0.74        0.02
SVC                                0.74               0.70     0.70      0.74        0.04
(base) PS D:\AI>
```

Q16. Perform K-Means on Iris dataset and visualize clusters.

```
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import pandas as pd
print("Namer : Aryan Rana \nRoll No. : 1323223")
iris = load_iris()
X = iris.data
y = iris.target
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)
labels = kmeans.labels_
pca = PCA(n_components=2)
reduced_X = pca.fit_transform(X)
df = pd.DataFrame(reduced_X, columns=["PC1", "PC2"])
df["Cluster"] = labels
plt.figure(figsize=(8, 6))
plt.scatter(df["PC1"], df["PC2"], c=df["Cluster"], cmap="viridis", s=50)
plt.title("K-Means Clusters on Iris Dataset (PCA Reduced)")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()
print(df.head())
```

OUTPUT



Q17. Use Tensorflow to build a feedforward neural network for **MNIST digit classification.**

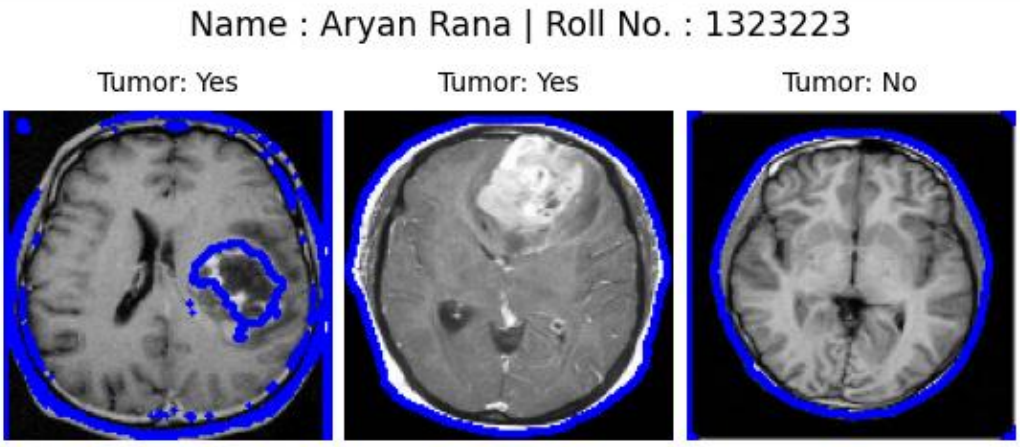
```
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
import numpy as np
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
x_train = x_train.reshape(-1, 28*28)
x_test = x_test.reshape(-1, 28*28)
model = models.Sequential([
    layers.Dense(128, activation='relu', input_shape=(784,)),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5, batch_size=128, validation_split=0.1)
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {test_acc:.4f}")
predictions = model.predict(x_test[:10])
predicted_labels = np.argmax(predictions, axis=1)
plt.figure(figsize=(10, 4))
for i in range(2):
    plt.subplot(2, 5, i + 1)
    plt.imshow(x_test[i].reshape(28, 28), cmap='gray')
    plt.title(f"Pred: {predicted_labels[i]}, True: {y_test[i]}")
    plt.axis('off')
plt.tight_layout()
plt.suptitle("Name : Aryan Rana | Roll No. : 1323223", fontsize=10)
plt.show()
```

OUTPUT

Q18. Train a **CNN** on a Brain Tumor Detection dataset to predict if an image has a tumor or not.

```
import os
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import cv2
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers, models
import random
data_dir = r"D:\AI\archive\brain_tumor_dataset"
img_size = (128, 128)
datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
train = datagen.flow_from_directory(data_dir, target_size=img_size, subset='training',
class_mode='binary')
val = datagen.flow_from_directory(data_dir, target_size=img_size,
subset='validation', class_mode='binary')
model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=img_size+(3,)),
    layers.MaxPooling2D(2,2),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D(2,2),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(train, validation_data=val, epochs=5)
sample_images = random.sample(val.filepaths, 3)
plt.figure(figsize=(12, 4))  # wide figure for single-row display
for i, path in enumerate(sample_images):
    img = cv2.imread(path)
    img_resized = cv2.resize(img, img_size)
    gray = cv2.cvtColor(img_resized, cv2.COLOR_BGR2GRAY)
    _, mask = cv2.threshold(gray, 180, 255, cv2.THRESH_BINARY)
    contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    marked = img_resized.copy()
    cv2.drawContours(marked, contours, -1, (255, 0, 0), 2)
    plt.subplot(1, 3, i + 1)
    plt.imshow(cv2.cvtColor(marked, cv2.COLOR_BGR2RGB))
    plt.title(f"Tumor: {'Yes' if 'yes' in path else 'No'}", fontsize=8)
    plt.suptitle("Name : Aryan Rana | Roll No. : 1323223", fontsize=10)
    plt.axis('off')
plt.tight_layout()
plt.show()
```

OUTPUT



Q19. Train another **feedforward neural network** on MNIST to practice **hyperparameter tuning.**

```
import tensorflow as tf
from tensorflow.keras import layers, models, optimizers
from tensorflow.keras.datasets import mnist
import numpy as np
import matplotlib.pyplot as plt
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
x_train = x_train.reshape(-1, 28*28)
x_test = x_test.reshape(-1, 28*28)
def build_model(hidden_units=128, learning_rate=0.001):
    model = models.Sequential([
        layers.Dense(hidden_units, activation='relu', input_shape=(784,)),
        layers.Dense(hidden_units // 2, activation='relu'),
        layers.Dense(10, activation='softmax')
    ])
    optimizer = optimizers.Adam(learning_rate=learning_rate)
    model.compile(optimizer=optimizer,
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    return model
hidden_units_list = [64, 128, 256]
learning_rates = [0.01, 0.001]
batch_sizes = [64, 128]
best_acc = 0
best_config = None
for hidden_units in hidden_units_list:
    for lr in learning_rates:
        for batch in batch_sizes:
            print(f"\nTraining model: hidden_units={hidden_units}, lr={lr}, batch={batch}")
            model = build_model(hidden_units, lr)
            history = model.fit(x_train, y_train, epochs=3, batch_size=batch,
validation_split=0.1, verbose=0)
            val_acc = history.history['val_accuracy'][-1]
            print(f"Validation Accuracy: {val_acc:.4f}")
            if val_acc > best_acc:
                best_acc = val_acc
                best_config = (hidden_units, lr, batch)
print(f"\nBest configuration -> Hidden Units: {best_config[0]}, Learning Rate:
{best_config[1]}, Batch Size: {best_config[2]}")
final_model = build_model(best_config[0], best_config[1])
final_model.fit(x_train, y_train, epochs=5, batch_size=best_config[2],
validation_split=0.1, verbose=1)
test_loss, test_acc = final_model.evaluate(x_test, y_test)
print(f"Test Accuracy (Best Model): {test_acc:.4f}")
predictions = final_model.predict(x_test[:10])
predicted_labels = np.argmax(predictions, axis=1)
plt.figure(figsize=(8, 4))
```

```
for i in range(2):
    plt.suptitle("Name : Aryan Rana | Roll No. : 1323223", fontsize=8)
    plt.subplot(1, 2, i + 1)
    plt.imshow(x_test[i].reshape(28, 28), cmap='gray')
    plt.title(f"Pred: {predicted_labels[i]}, True: {y_test[i]}")
    plt.axis('off')
plt.tight_layout()
plt.show()
```

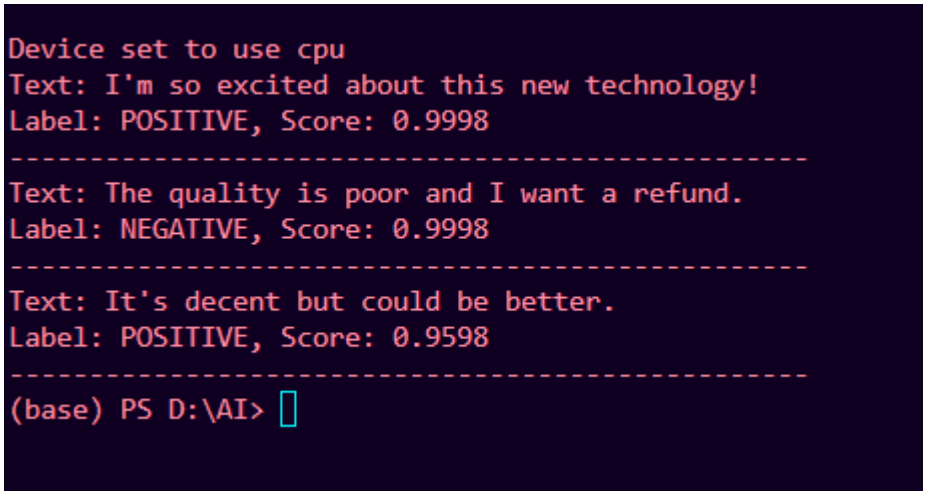
OUTPUT


```
Validation Accuracy: 0.9670

Training model: hidden_units=64, lr=0.01, batch=128
Validation Accuracy: 0.9643

Training model: hidden_units=64, lr=0.001, batch=64
Validation Accuracy: 0.9695

Training model: hidden_units=64, lr=0.001, batch=128
Validation Accuracy: 0.9672

Training model: hidden_units=128, lr=0.01, batch=64
Validation Accuracy: 0.9643

Training model: hidden_units=128, lr=0.01, batch=128
Validation Accuracy: 0.9717

Training model: hidden_units=128, lr=0.001, batch=64
Validation Accuracy: 0.9752

Training model: hidden_units=128, lr=0.001, batch=128
Validation Accuracy: 0.9738

Training model: hidden_units=256, lr=0.01, batch=64
Validation Accuracy: 0.9632

Training model: hidden_units=256, lr=0.01, batch=128
Validation Accuracy: 0.9753

Training model: hidden_units=256, lr=0.001, batch=64
Validation Accuracy: 0.9775

Training model: hidden_units=256, lr=0.001, batch=128
Validation Accuracy: 0.9785

Best configuration -> Hidden Units: 256, Learning Rate: 0.001, Batch Size: 128
```

Q20. Use a pre-trained **DistilBERT model** for sentiment classification of text.

```
from transformers import pipeline
sentiment_pipeline = pipeline(
    "sentiment-analysis",
    model="distilbert-base-uncased-finetuned-sst-2-english",
    tokenizer="distilbert-base-uncased-finetuned-sst-2-english"
)
texts = [
    "I'm so excited about this new technology!",
    "The quality is poor and I want a refund.",
    "It's decent but could be better."
]

results = sentiment_pipeline(texts)

for text, result in zip(texts, results):
    print(f"Text: {text}")
    print(f"Label: {result['label']}, Score: {result['score']:.4f}")
    print("-" * 50)
```

OUTPUT



Q21. Use Hugging Face pipeline("text-generation") with GPT-2 to generate text from a prompt.

```
from transformers import pipeline, AutoTokenizer, AutoModelForCausalLM
tokenizer = AutoTokenizer.from_pretrained("gpt2")
tokenizer.pad_token = tokenizer.eos_token
model = AutoModelForCausalLM.from_pretrained("gpt2")
generator = pipeline(
    "text-generation",
    model=model,
    tokenizer=tokenizer
)
def gpt(prompt):
    result = generator(
        prompt,
        max_new_tokens=60,
        temperature=0.7,
        top_p=0.9,
        truncation=True,
        pad_token_id=tokenizer.eos_token_id
    )
    return result[0]["generated_text"]
print("Type 'exit' or 'quit' to end the program.\n")
while True:
    user_input = input("You: ").strip()
    if user_input.lower() in ("exit", "quit", "bye"):
        print("GPT: Goodbye!")
        break
    elif user_input in ("hii", "hello", "hi","hey"):
        print("GPT: Hello! How can I assist you today?")
        continue
    else:
        response = gpt(user_input)
        print("GPT:", response)
```

OUTPUT

```
Device set to use cpu
Type 'exit' or 'quit' to end the program.

You: hello
GPT: Hello! How can I assist you today?
You: Tell me about AI
GPT: Tell me about AI and how it's affecting your life.

I'm starting to think that if I were to give you the chance to ask me what I thin
k about AI, I'd be able to give you the best answer. And I think that this is imp
ortant to understand. I'm a big believer that
You: bye
GPT: Goodbye!
(base) PS D:\AI>
```

Q22. Use **Hugging Face Diffusers** (Stable Diffusion) to generate images from text prompts.

```
import torch
from diffusers import DiffusionPipeline
pipeline = DiffusionPipeline.from_pretrained(
    "stabilityai/stable-diffusion-xl-base-1.0",
    torch_dtype=torch.float16,
    variant="fp16"
)
if torch.cuda.is_available():
    pipeline.to("cuda")
else:
    print("CUDA not available. Running on CPU (this will be very slow).")
prompt = "An astronaut riding a horse on Mars, high quality, digital art"
negative_prompt = "blurry, low quality, bad anatomy, text, watermark"
print("Generating image...")
image = pipeline(
    prompt=prompt,
    negative_prompt=negative_prompt,
    num_inference_steps=30,
    guidance_scale=7.5
).images[0]
image.save("astronaut_on_mars.png")
print("Image saved as 'astronaut_on_mars.png'")
```

OUTPUT

```
Loading pipeline components...: 100% 7/7 [00:02<00:00, 2.57it/s]
Generating image...
100% 30/30 [00:21<00:00, 1.39it/s]
Image saved as 'astronaut_on_mars.png'
```

astronaut_on_mars.png

Q23. Use Google open source gemini api key and create a suitable promptings to build a personalize text summarizer

```
import os
from dotenv import load_dotenv
import google.generativeai as genai
load_dotenv()
api_key = os.getenv("GEMINI_API_KEY")
if not api_key:
    raise ValueError("GEMINI_API_KEY is missing in the .env file.")
genai.configure(api_key=api_key)
PERSONAL_SUMMARY_PROMPT = """
You are a personalized text-summarizer.
Summarize the given text with the following rules:
1. Output must be factual and concise.
2. Remove repetitive, irrelevant, or emotional filler.
3. Preserve important technical details, data, and definitions.
4. Maintain a professional tone.
5. Format the output using short paragraphs or bullet points.
6. Do not add assumptions or opinions.
Text to summarize:
{input_text}
Generate the best possible summary.
"""
model = genai.GenerativeModel("gemini-2.5-flash")
def summarize_text(text):
    prompt = PERSONAL_SUMMARY_PROMPT.format(input_text=text)
    try:
        response = model.generate_content(prompt)
        return response.text
    except Exception as e:
        return f"An error occurred during summarization: {e}"
input_text = """
Artificial Intelligence is transforming industries by automating tasks,
enhancing decision-making, and increasing operational efficiency.
However, concerns such as job displacement, ethical risks, and privacy
issues are rising as AI adoption grows.
"""
print(summarize_text(input_text))
```

OUTPUT