Roll No.26 Div:D15C Name :Aditya kirtane

AIDS-I Assignment No: 2

Q.1: Use the following data set for question 1

82, 66, 70, 59, 90, 78, 76, 95, 99, 84, 88, 76, 82, 81, 91, 64, 79, 76, 85, 90

1. Find the Mean (10pts)
2. Find the Median (10pts)
3. Find the Mode (10pts)
4. Find the Interquartile range (20pts)

Solution:

Let's solve each part step by step using the given data set:

Data Set: 82, 66, 70, 59, 90, 78, 76, 95, 99, 84, 88, 76, 82, 81, 91, 64, 79, 76, 85, 90

---

1. Mean (Average)

To find the mean:

Mean=Sum of all valuesNumber of values\text{Mean} = \frac{\text{Sum of all values}}{\text{Number of values}}Mean=Number of valuesSum of all values Sum=82+66+70+59+90+78+76+95+99+84+88+76+82+81+91+64+79+76+85+90=1621\text{Sum} = 82 + 66 + 70 + 59 + 90 + 78 + 76 + 95 + 99 + 84 + 88 + 76 + 82 + 81 + 91 + 64 + 79 + 76 + 85 + 90 = 1621Sum=82+66+70+59+90+78+76+95+99+84+88+76+82+81+91+64+79+76+85+90=1621 Number of values=20\text{Number of values} = 20Number of values=20 Mean=162120=81.05\text{Mean} = \frac{1621}{20} = 81.05Mean=201621=81.05

☑ **Mean = 81.05**

---

2. Median

To find the median, we sort the data and find the middle value(s).

**Sorted data: 59, 64, 66, 70, 76, 76, 76, 78, 79, 81,
82, 82, 84, 85, 88, 90, 90, 91, 95, 99**

There are 20 numbers (even), so the median is the average of the 10th and 11th values.

10th = 81
11th = 82

Median=81+822=1632=81.5\text{Median} = \frac{81 + 82}{2} = \frac{163}{2} = 81.5Median=281+82=2163=81.5

☑ **Median = 81.5**

---

3. Mode

The mode is the number(s) that appear most frequently.

From the sorted list:

- **76 appears 3 times**
- **Others appear less frequently**

☑ **Mode = 76**

---

4. Interquartile Range (IQR)

To calculate the IQR, we need:

- **Q1 (First Quartile) = Median of the lower half**
- **Q3 (Third Quartile) = Median of the upper half**

Lower half (first 10 values):
59, 64, 66, 70, 76, 76, 76, 78, 79, 81
Q1 = Median of this half = average of 5th and 6th values = (76 + 76)/2 = 76

Upper half (last 10 values):
82, 82, 84, 85, 88, 90, 90, 91, 95, 99
Q3 = Median of this half = average of 5th and 6th values = (88 + 90)/2 = 89

IQR=Q3−Q1=89−76=13\text{IQR} = Q3 - Q1 = 89 - 76 = 13IQR=Q3−Q1=89−76=13

☑ **Interquartile Range = 13**

☑ **Final Answers:**

1. **Mean: 81.05**
2. **Median: 81.5**
3. **Mode: 76**
4. **Interquartile Range: 13**

Q.2 1) Machine Learning for Kids 2) Teachable Machine

1. For each tool listed above
    - identify the target audience
    - discuss the use of this tool by the target audience
    - identify the tool's benefits and drawbacks
2. From the two choices listed below, how would you describe each tool listed above? Why did you choose the answer?

    - Predictive analytic
    - Descriptive analytic

3. From the three choices listed below, how would you describe each tool listed above? Why did you choose the answer?

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Solution ;

1. Tool Analysis

A) Machine Learning for Kids

• Target Audience:

- Primarily **school students (ages 8–16)**, **educators**, and **beginners** in machine learning.

• Use by Target Audience:

- Students and teachers use it to:
    - Learn basic concepts of ML using a friendly interface.
    - Create simple text, image, and number-based ML models.
    - Train models with labeled data and test them using their own examples.
    - Integrate models into Scratch or Python projects for real-world simulation.

• **Benefits:**

- Free and easy to use.
- Designed for **education and conceptual understanding**.
- Integrates with Scratch and Python.
- No prior coding or ML experience required.

• **Drawbacks:**

- Limited to basic ML concepts; **not suitable for advanced learning**.
- Less control over complex algorithms or fine-tuning.
- Limited data processing capabilities.

B) Teachable Machine

• **Target Audience:**

- **General public**, including **students, hobbyists, educators**, and even **designers** or **non-programmers** interested in AI.

• **Use by Target Audience:**

- Users upload images, audio, or pose examples to train classification models.
- Real-time feedback allows users to understand how data affects model behavior.
- Trained models can be exported and used in web apps or TensorFlow projects.

• **Benefits:**

- Extremely easy and **interactive**.
- Fast results with **real-time testing**.
- No programming knowledge required.
- Models can be downloaded for use in custom projects.

• **Drawbacks:**

- Limited to **classification tasks** (image/audio/pose).
- Not ideal for large datasets or professional-level applications.
- Lack of transparency in model architecture.

---

2. Analytic Type Classification

A) Machine Learning for Kids: → Predictive Analytic

- **Why?** It teaches kids how to **train a model on labeled data** to make predictions (e.g., classify text as positive or negative, recognize images). This is a key trait of **predictive analytics**.

B) Teachable Machine: → Predictive Analytic

- **Why?** The tool is used to **train and test models that predict classes** based on new input (e.g., identifying whether an image belongs to Class A, B, or C). Hence, it falls under **predictive analytics**.

---

3. Learning Type Classification

A) Machine Learning for Kids: → Supervised Learning

- **Why?** Users must provide **labeled training data** (e.g., tag images or words). The model then learns the relationship between input and output. This is the essence of **supervised learning**.

B) Teachable Machine: → Supervised Learning

- **Why?** Users upload **examples with class labels** (e.g., pictures tagged as "cat" or "dog"). The system learns from these to predict the correct label for new data—definitely **supervised learning**

Summary Table

Tool	Target Audience	Type of Analytic	Type of Learning	Benefits	Drawbacks
Machine Learning for Kids	Students, Educators	Predictive Analytic	Supervised Learning	Easy to use, educational, Scratch integration	Limited scope, basic only
Teachable Machine	General public, Beginners	Predictive Analytic	Supervised Learning	Real-time testing, no coding needed	Only does classification, basic

Q.3 Data Visualization: Read the following two short articles:

- Read the article Kakande, Arthur. February 12. "What's in a chart? A Step-by-Step guide to Identifying Misinformation in Data Visualization." *Medium*
- Read the short web page Foley, Katherine Ellen. June 25, 2020. "How bad Covid-19 data visualizations mislead the public." *Quartz*
- Research a current event which highlights the results of misinformation based on data visualization. Explain how the data visualization method failed in presenting accurate information. Use newspaper articles, magazines, online news websites or any other legitimate and valid source to cite this example. Cite the news source that you found.

Solution :

Data visualizations are powerful tools for conveying complex information succinctly. However, when misused or poorly designed, they can mislead audiences and propagate misinformation. A pertinent example of this occurred in July 2024, involving misleading social media posts about sexually transmitted diseases (STDs) in Houston, Texas.Reuters

Case Study: Misleading STD Statistics in Houston

In July 2024, social media platforms were abuzz with alarming claims that over 40,000 individuals in Houston had tested positive for STDs within a single week. These assertions were accompanied by screenshots of data tables listing various STDs, including chlamydia, gonorrhea, syphilis, and HIV, along with corresponding figures. The presentation of this data, without proper context, led many to believe there was a sudden and massive outbreak of STDs in Houston.Reuters

How the Data Visualization Was Misleading

1. Lack of Context: The figures presented were not exclusive to Houston but represented the total number of STD tests conducted across the entire state of Texas. This crucial

detail was omitted, leading viewers to draw incorrect conclusions about the health situation in Houston.Reuters

2. Misinterpretation of Data: The numbers in the table reflected the total tests administered, encompassing both positive and negative results. However, the accompanying captions and the way the data was framed suggested that all the figures represented positive cases, which was not the case.Reuters

3. Visual Presentation: The data was displayed in a straightforward table format without explanatory notes or sources. This lack of clarity made it easy for misinformation to spread, as viewers had no immediate way to verify the authenticity or scope of the data.

Impact and Response

The misleading posts quickly gained traction, causing unnecessary panic and concern among Houston residents. In response, the Houston Health Department clarified that the numbers were being misrepresented and that no such surge in STD cases had occurred in the city. They emphasized the importance of interpreting data within its proper context and cautioned against the spread of unverified information. Additionally, the department investigated the misuse of their data and implemented measures to prevent similar incidents in the future. Reuters

Lessons Learned

This incident underscores the critical need for careful and responsible data visualization practices:

- Provide Clear Context: Always accompany data visualizations with explanations that define the scope, source, and meaning of the data presented.

- Avoid Ambiguity: Ensure that visual elements do not lend themselves to multiple interpretations. Use labels, legends, and notes to guide the audience toward accurate understanding.

- Verify Before Sharing: Before disseminating data visualizations, especially on public platforms, verify the accuracy and context of the information to prevent the spread of misinformation.

By adhering to these principles, communicators can maintain the integrity of information and foster

Q. 4 Train Classification Model and visualize the prediction performance of trained model required information

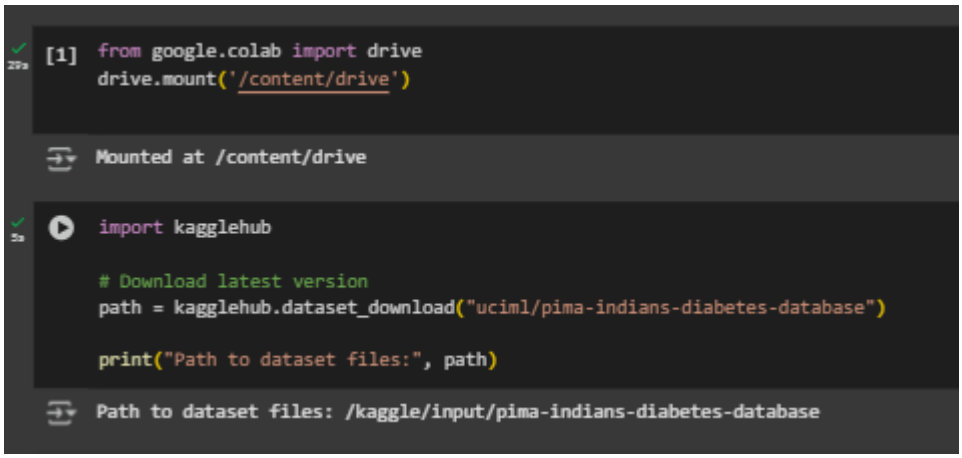
- Data File: Classification data.csv
- Class Label: Last Column
- Use any Machine Learning model (SVM, Naïve Base Classifier)

Requirements to satisfy

- Programming Language: Python
- Class imbalance should be resolved
- Data Pre-processing must be used
- Hyper parameter tuning must be used
- Train, Validation and Test Split should be 70/20/10
- Train and Test split must be randomly done
- Classification Accuracy should be maximized
- Use any Python library to present the accuracy measures of trained model

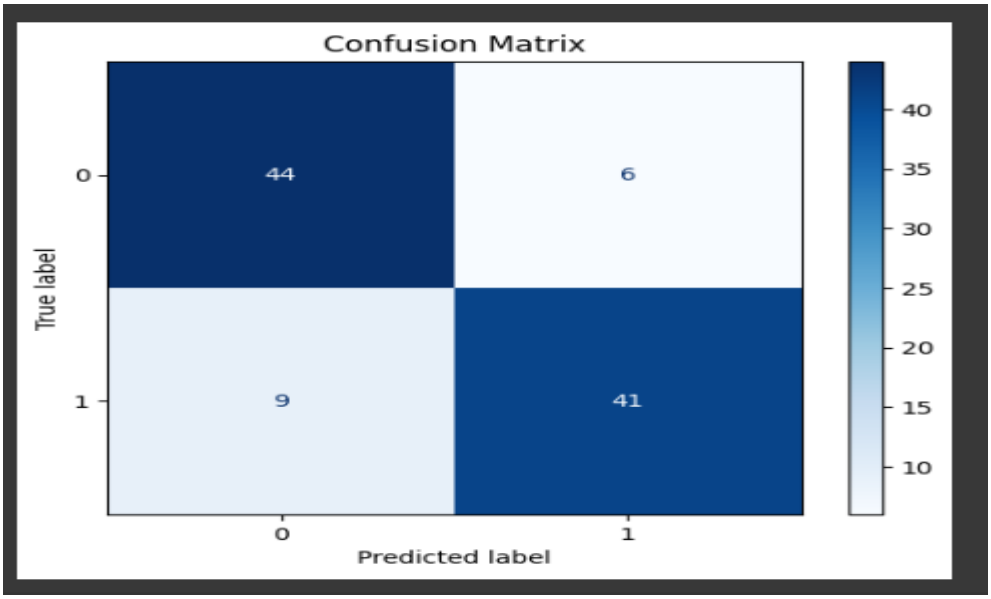
[Pima Indians Diabetes Database]

Solution :



```python
# STEP 1: Install required libraries (only if not already installed)
!pip install -q kagglehub imbalanced-learn

# STEP 2: Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, ConfusionMatrixDisplay
from sklearn.svm import SVC
from imblearn.over_sampling import SMOTE
import kagglehub

import warnings
warnings.filterwarnings("ignore")

# STEP 3: Download dataset using kagglehub
path = kagglehub.dataset_download("uciml/pima-indians-diabetes-database")
print("Path to dataset files:", path)

# Load the diabetes.csv file
df = pd.read_csv(f"{path}/diabetes.csv")
print("Dataset loaded successfully.")
print(df.head())

# STEP 4: Preprocess
print("Missing values:\n", df.isnull().sum())
df.dropna(inplace=True)  # Optionally impute if needed

# Separate features and target
X = df.iloc[:, :-1]
y = df.iloc[:, -1]

# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# STEP 5: Fix class imbalance using SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_scaled, y)

# STEP 6: Split data (Train 70%, Validation 20%, Test 10%)
X_train, X_temp, y_train, y_temp = train_test_split(
    X_resampled, y_resampled, test_size=0.30, random_state=42, stratify=y_resampled)

X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=1/3, random_state=42, stratify=y_temp)

print(f"Train: {len(X_train)}, Validation: {len(X_val)}, Test: {len(X_test)}")

# STEP 7: Train model (SVM) with hyperparameter tuning
param_grid = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf', 'poly'],
    'gamma': ['scale', 'auto']
}

svc = SVC()
grid = GridSearchCV(svc, param_grid, cv=5, scoring='accuracy')
grid.fit(X_train, y_train)

print("Best Parameters:", grid.best_params_)

# Validation accuracy
val_preds = grid.predict(X_val)
print("Validation Accuracy:", accuracy_score(y_val, val_preds))

# STEP 8: Evaluate on Test Set
test_preds = grid.predict(X_test)
print("\nTest Accuracy:", accuracy_score(y_test, test_preds))
print("\nClassification Report:\n", classification_report(y_test, test_preds))

# Confusion Matrix
cm = confusion_matrix(y_test, test_preds)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap='Blues')
plt.title("Confusion Matrix")
plt.show()
```

```
Path to dataset files: /kaggle/input/pima-indians-diabetes-database
Dataset loaded successfully.
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
Missing values:
 Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
Train: 700, Validation: 200, Test: 100
Best Parameters: {'C': 10, 'gamma': 'auto', 'kernel': 'rbf'}
Validation Accuracy: 0.8

Test Accuracy: 0.85

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.88      0.85        50
           1       0.87      0.82      0.85        50

    accuracy                           0.85       100
   macro avg       0.85      0.85      0.85       100
weighted avg       0.85      0.85      0.85       100
```



Q.5 Train Regression Model and visualize the prediction performance of trained model

- Data File: Regression data.csv
- Independent Variable: 1st Column
- Dependent variables: Column 2 to 5

Use any Regression model to predict the values of all Dependent variables using values of 1st column.
Requirements to satisfy:

- Programming Language: Python
- OOP approach must be followed
- Hyper parameter tuning must be used
- Train and Test Split should be 70/30
- Train and Test split must be randomly done
- Adjusted R2 score should more than 0.99
- Use any Python library to present the accuracy measures of trained model

https://github.com/Sutanoy/Public-Regression-Datasets

https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv

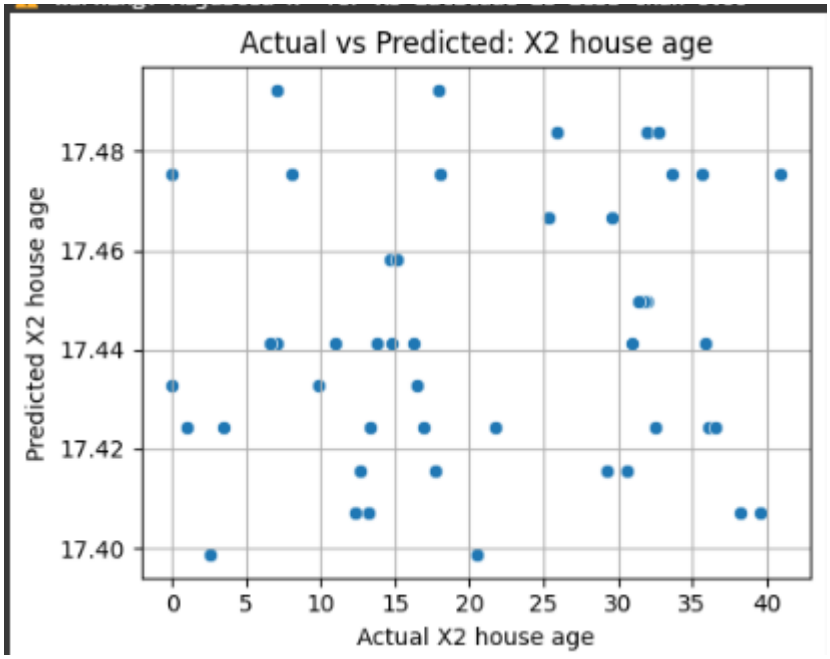
- URL: https://archive.ics.uci.edu/ml/machine-learning-databases/00477/Real%20estate%20valuation%20data%20set.xlsx

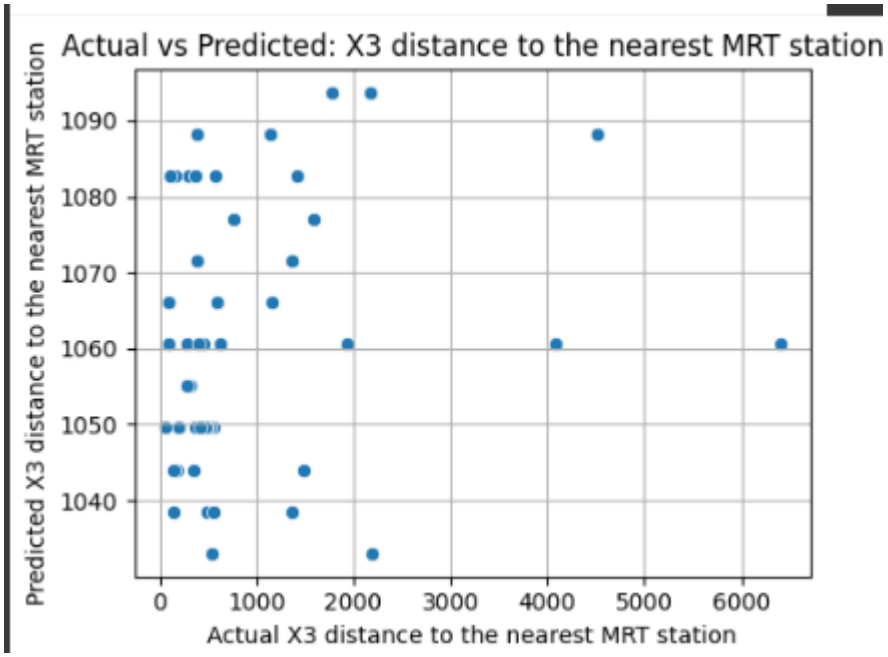
(Refer any one)

Solution :

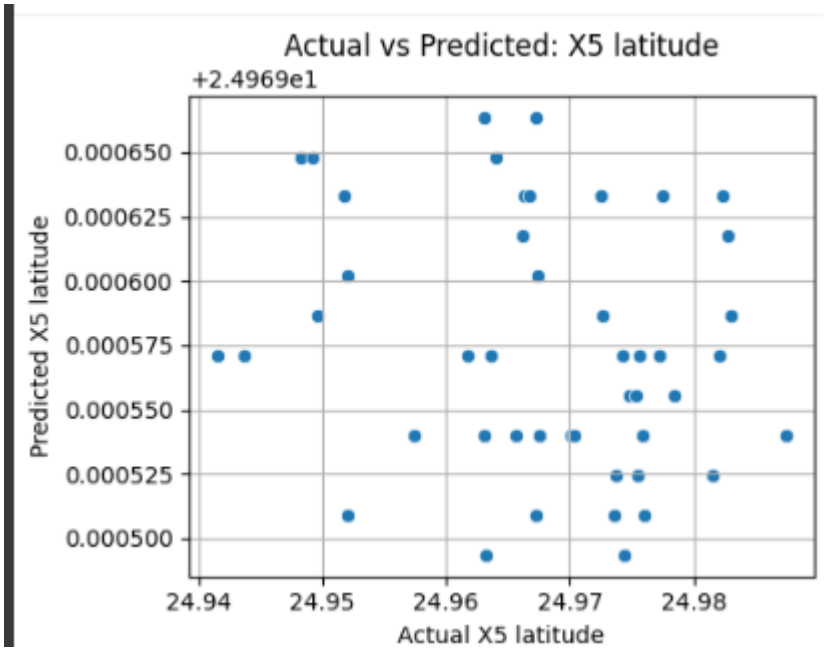
```python
# STEP 1: Install required libraries
!pip install -q openpyxl scikit-learn matplotlib seaborn

# STEP 2: Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import Ridge
from sklearn.metrics import r2_score

import warnings
warnings.filterwarnings("ignore")

# STEP 3: Load dataset from UCI (Excel file)
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/00477/Real%20estate%20valuation%20data%20set.xlsx"
df = pd.read_excel(url)

# Drop 'No' column if it exists
if "No" in df.columns:
    df.drop("No", axis=1, inplace=True)

print("Dataset Loaded:")
print(df.head())

# STEP 4: Extract input and outputs
X = df.iloc[:, [0]]   # Independent variable: 1st column
Y = df.iloc[:, 1:5]   # Dependent variables: columns 2-5

# STEP 5: Define OOP-based Multi-Output Regression Model
class MultiOutputRegressor:
    def __init__(self, model):
        self.base_model = model
        self.models = {}

    def train(self, X_train, Y_train, params=None):
        for column in Y_train.columns:
            grid = GridSearchCV(self.base_model, param_grid=params, cv=5, scoring='r2')
            grid.fit(X_train, Y_train[column])
            self.models[column] = grid.best_estimator_
            print(f"✅ Best Params for {column}: {grid.best_params_}")

    def predict(self, X):
        predictions = pd.DataFrame()
        for column, model in self.models.items():
            predictions[column] = model.predict(X)
        return predictions

    def evaluate(self, X_test, Y_test):
        preds = self.predict(X_test)
        for col in Y_test.columns:
            r2 = r2_score(Y_test[col], preds[col])
            n = len(Y_test)
            p = 1  # only 1 feature
            adj_r2 = 1 - (1 - r2) * (n - 1) / (n - p - 1)
            print(f"🔍 {col}: Adjusted R² = {adj_r2:.5f} | R² = {r2:.5f}")
            if adj_r2 < 0.99:
                print(f"⚠️ Warning: Adjusted R² for {col} is less than 0.99")
        return preds

    def plot_results(self, Y_test, Y_pred):
        for col in Y_test.columns:
            plt.figure(figsize=(5, 4))
            sns.scatterplot(x=Y_test[col], y=Y_pred[col])
            plt.xlabel(f"Actual {col}")
            plt.ylabel(f"Predicted {col}")
            plt.title(f"Actual vs Predicted: {col}")
            plt.grid(True)
            plt.tight_layout()
            plt.show()

# STEP 6: Train-Test Split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.30, random_state=42)

# STEP 7: Train the model with Ridge Regression and tuning
regressor = MultiOutputRegressor(Ridge())

param_grid = {
    'alpha': [0.01, 0.1, 1, 10, 100]
}

regressor.train(X_train, Y_train, param_grid)

# STEP 8: Evaluate and plot
Y_pred = regressor.evaluate(X_test, Y_test)
regressor.plot_results(Y_test, Y_pred)
```

```
Dataset Loaded:
   X1 transaction date  X2 house age  X3 distance to the nearest MRT station  \
0          2012.916667          32.0                                84.87882
1          2012.916667          19.5                               306.59470
2          2013.583333          13.3                               561.98450
3          2013.500000          13.3                               561.98450
4          2012.833333           5.0                               390.56840

   X4 number of convenience stores  X5 latitude  X6 longitude  \
0                               10     24.98298     121.54024
1                                9     24.98034     121.53951
2                                5     24.98746     121.54391
3                                5     24.98746     121.54391
4                                5     24.97937     121.54245

   Y house price of unit area
0                        37.9
1                        42.2
2                        47.3
3                        54.8
4                        43.1
✅ Best Params for X2 house age: {'alpha': 100}
✅ Best Params for X3 distance to the nearest MRT station: {'alpha': 100}
✅ Best Params for X4 number of convenience stores: {'alpha': 100}
✅ Best Params for X5 latitude: {'alpha': 100}
🔍 X2 house age: Adjusted R² = -0.01397 | R² = -0.00579
⚠️ Warning: Adjusted R² for X2 house age is less than 0.99
🔍 X3 distance to the nearest MRT station: Adjusted R² = -0.01003 | R² = -0.00188
⚠️ Warning: Adjusted R² for X3 distance to the nearest MRT station is less than 0.99
🔍 X4 number of convenience stores: Adjusted R² = -0.01866 | R² = -0.01045
⚠️ Warning: Adjusted R² for X4 number of convenience stores is less than 0.99
🔍 X5 latitude: Adjusted R² = -0.03105 | R² = -0.02274
⚠️ Warning: Adjusted R² for X5 latitude is less than 0.99
```




Actual vs Predicted: X3 distance to the nearest MRT station
Predicted X3 distance to the nearest MRT station
1090
1080
1070
1060
1050
1040
0
1000
2000
3000
4000
5000
6000
Actual X3 distance to the nearest MRT station


Actual vs Predicted: X4 number of convenience stores
Predicted X4 number of convenience stores
4.20
4.19
4.18
4.17
4.16
0
2
4
6
8
10
Actual X4 number of convenience stores



Q.6 What are the key features of the wine quality data set? Discuss the importance of each feature in predicting the quality of wine? How did you handle missing data in the wine quality data set during the feature engineering process? Discuss the advantages and disadvantages of different imputation techniques. (Refer dataset from Kaggle).

Solution:

Key Features of the Wine Quality Dataset

Feature	Description
fixed acidity	Tartaric acid concentration; contributes to wine's stability and taste
volatile acidity	Acetic acid; high levels lead to unpleasant vinegar taste
citric acid	Adds freshness and flavor; can enhance wine quality
residual sugar	Sugar left after fermentation; affects sweetness
chlorides	Salt concentration; high levels negatively impact quality
free sulfur dioxide	Prevents microbial growth; excessive use can harm flavor
total sulfur dioxide	Includes bound + free SO_2; excessive levels affect health and aroma
density	Affects mouthfeel; high density often means higher sugar or alcohol
pH	Acidity level; lower pH means more acidic
sulphates	Add to wine's antimicrobial stability and taste profile
alcohol	Strongest predictor of quality; higher alcohol often means better quality
quality *(target)*	Integer score between 0 and 10, based on sensory data

Importance of Each Feature in Predicting Wine Quality

1. **Alcohol**: Positively correlated with quality – stronger wines are generally preferred.
2. **Volatile Acidity**: Negative impact – higher levels usually reduce quality.
3. **Sulphates**: Moderate positive impact – improves preservation and taste.
4. **Citric Acid**: Enhances freshness and contributes to a crisp taste.
5. **Residual Sugar**: Small influence – only certain wine types benefit from high sugar.
6. **Total/Free SO_2**: Affects preservation but excessive values degrade taste.
7. **Fixed Acidity & pH**: Interact with other acids; control wine's sharpness and stability.

📊 Feature importance can be quantitatively analyzed using feature importance plots (e.g., using Random Forest or SHAP values).

Handling Missing Data During Feature Engineering

Method	Description	Advantages	Disadvantages
Mean/Median/Mode Imputation	Replace missing values with mean/median/mode of the column	Simple, fast	Can distort variance; doesn't preserve relationships
KNN Imputation	Uses K-Nearest Neighbors to impute missing values based on similarity	Preserves data patterns	Slow with large datasets; sensitive to outliers
Multivariate Imputation (e.g., MICE)	Uses regression models to estimate missing values	Captures inter-feature relationships	Computationally intensive
Model-Based (e.g., Regression)	Predict missing values using another regression model	More accurate if correlations are strong	Risk of overfitting
Drop Rows	Remove rows with missing data	Simple	Can lose valuable data and reduce sample size

Best Practice (if data was missing):

- Use **median** for skewed data
- Use **KNN or MICE** for more advanced, correlated datasets
- Always compare the results **before vs. after imputatio**

Summary

- The wine quality dataset contains **11 key physicochemical features**.
- **Alcohol, volatile acidity, and sulphates** have the most predictive power.
- If missing values exist, choose an **imputation method** based on data size, missing rate, and feature relationships.
- Proper **feature engineering and imputation** directly impact model performance.