NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

AIDS - 1 Assignment 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)

ANS->

1. Mean

Formula:
Mean =Sum of all data points / Total number of data points

Calculation:
Sum = 82+66+70+ 59+ 90+ 78+ 76+ 95+ 99+ 84+ 88+ 76+ 82+ 81+ 91+ 64+ 79+ 76+ 85+ 90
Sum = 1621
Mean = 1621/20 = 81.05

Mean = 81.05

2. Median

Steps:
First, arrange the data in ascending order:
59, 64, 66, 70, 76, 76, 76, 78, 79, 81, 82, 82, 84, 85, 88, 90, 90, 91, 95, 99
Since the number of values (n = 20) is even, the median is the average of the 10th and 11th terms.

Median = 81 + 82 / 2 = 163/2= 81.5
Median = 81.5

3. Mode

Steps:

Look for the number(s) that occur most frequently.

From the sorted list:
76 occurs 3 times, while all other numbers occur once or twice.

Mode = 76

4. Interquartile Range

Steps:
The Interquartile Range (IQR) is defined as: IQR=Q3−Q1

From the ordered dataset:
59, 64, 66, 70, 76, 76, 76, 78, 79, 81, 82, 82, 84, 85, 88, 90, 90, 91, 95, 99

Q1 (First Quartile):
Lower half (first 10 numbers):
59, 64, 66, 70, 76, 76, 76, 78, 79, 81

Q1 is the median of the first half = average of 5th and 6th values:
Q1 = 76+76 / 2 =76

Q3 (Third Quartile):
Upper half (last 10 numbers):
82, 82, 84, 85, 88, 90, 90, 91, 95, 99

Q3 is the median of the second half = average of 5th and 6th values:
Q3= 88+90 / 2 = 89

IQR Calculation:

IQR=89−76=13
Interquartile Range = 13

Q.2: Comparison and Analysis of Machine Learning for Kids and Teachable Machine
Ans ->
1. For each tool listed below:

A. Machine Learning for Kids

- Target Audience: Primarily designed for school-aged students, beginners, and educators who want to introduce AI and machine learning concepts in an accessible, visual, and engaging way.
- Use by Target Audience: Students use the platform to train machine learning models using examples (text, numbers, images, or sound) and apply those models to interactive projects using Scratch or Python. Teachers utilize it as a classroom teaching tool to demonstrate AI concepts with hands-on projects.
- Benefits: Intuitive, beginner-friendly interface.
  Encourages experiential learning through integration with Scratch.
  Enables students to create real AI models using their own data.
  Free and web-based — no installation required.
- Drawbacks: Limited to basic functionalities — not suitable for advanced machine learning. Less control over model parameters or algorithms. Requires internet connection for use.

B. Teachable Machine

- Target Audience: Targeted towards students, educators, hobbyists, and non-programmers interested in quickly building and testing machine learning models, especially for image, audio, and pose detection.
- Use by Target Audience: Users can create classification models by uploading their own data (images, audio, or pose samples), train models quickly, and export them for use in websites, apps, or physical devices.
- Benefits: No coding required.
  Allows real-time training and testing of models.
  Provides export options to TensorFlow.js and TensorFlow Lite for deployment.
  Highly engaging and interactive interface.
- Drawbacks: Not suitable for complex or large-scale projects. Limited model customization. May not perform well with small or unbalanced datasets.

2. Classification of Tools: Predictive or Descriptive Analytics

Tool	Classification	Reason
Machine Learning for Kids	Predictive Analytic	It uses training data to build models that predict outcomes based on new input (e.g., classify images or text).

Teachable Machine	Predictive Analytic	It allows users to train models that predict or classify new inputs, such as recognizing gestures, faces, or sounds.

3. Classification of Tools: Type of Machine Learning

Tool	Learning Type	Reason
Machine Learning for Kids	Supervised Learning	Users train models using labeled data, where the input data is tagged with the correct output.
Teachable Machine	Supervised Learning	It also uses labeled data (e.g., tagging images with class names) to train models for classification.

Q3. 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.

Ans ->

1. **Key Insights from Assigned Readings**

a. "What's in a Chart? A Step-by-Step Guide to Identifying Misinformation in Data Visualization" by Arthur Kakande.

Arthur Kakande emphasizes the importance of critically evaluating data visualizations to detect misinformation. Key takeaways include:

- Assessing Data Sources: Verify the credibility and origin of the data presented.
- Examining Axes and Scales: Check for manipulated scales that can exaggerate or downplay trends.
- Identifying Omissions: Be alert to missing data or context that could alter interpretation.
- Recognizing Misleading Chart Types: Some chart types, like pie charts, can be easily misinterpreted if not used appropriately.

By applying these principles, readers can better discern the accuracy and intent behind data visualizations.

b. "How Bad Covid-19 Data Visualizations Mislead the Public" by Katherine Ellen Foley
Katherine Ellen Foley discusses specific examples from the COVID-19 pandemic where data visualizations misled the public:

- Snapshot Data Without Trends: Presenting single-day data without historical context can misrepresent the severity or improvement of situations.
- Overuse of Pie Charts: Complex data shown in pie charts can confuse readers, especially when depicting parts of a whole that don't sum intuitively.
- Missing Axes and Labels: Omitting axes or labels deprives viewers of necessary reference points, leading to potential misinterpretation.

These examples underscore the necessity for clarity, context, and completeness in data visualizations to ensure accurate public understanding.

2. **Case Study: Misleading Data Visualization in COVID-19 Vaccine Efficacy Reporting.**

Incident Overview

In March 2024, social media platforms circulated claims suggesting that COVID-19 vaccines were causing more harm than good. These assertions were based on data showing higher absolute numbers of deaths among vaccinated individuals compared to unvaccinated ones between July 2021 and May 2023. The data was presented in bar charts highlighting the total number of deaths in each group without considering the proportion of vaccinated versus unvaccinated individuals in the population.

Analysis of the Misleading Visualization

- Lack of Proportional Context: The visualizations displayed raw death counts without accounting for the fact that a significantly larger portion of the population was vaccinated.

Without normalizing the data (e.g., deaths per 100,000 individuals), the charts misleadingly suggested that vaccination increased mortality.

- Omission of Confounding Variables: Factors such as age, underlying health conditions, and exposure risks were not represented in the visualizations. These variables are critical in understanding mortality rates and the effectiveness of vaccines.

- Absence of Time Trends: The charts did not show how death rates changed over time in relation to vaccination rollouts, missing an opportunity to illustrate potential correlations between vaccination and mortality reductions.

Impact and Consequences

The misleading visualizations contributed to vaccine hesitancy among the public, undermining efforts to achieve widespread immunity. Public health officials had to invest additional resources to counteract the misinformation, emphasizing the importance of accurate data interpretation and presentation.

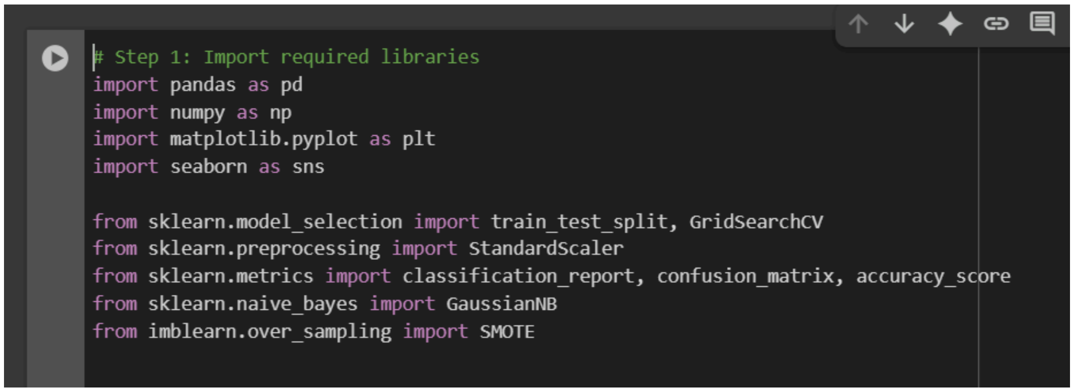
Conclusion

This case study highlights the critical importance of responsible data visualization. Misrepresenting data, whether intentionally or inadvertently, can have significant real-world consequences, particularly in public health. It is imperative for data communicators to present information transparently, ensuring that visualizations accurately reflect the underlying data and context.

Q4. Train Classification Model and visualize the prediction performance of trained model required information

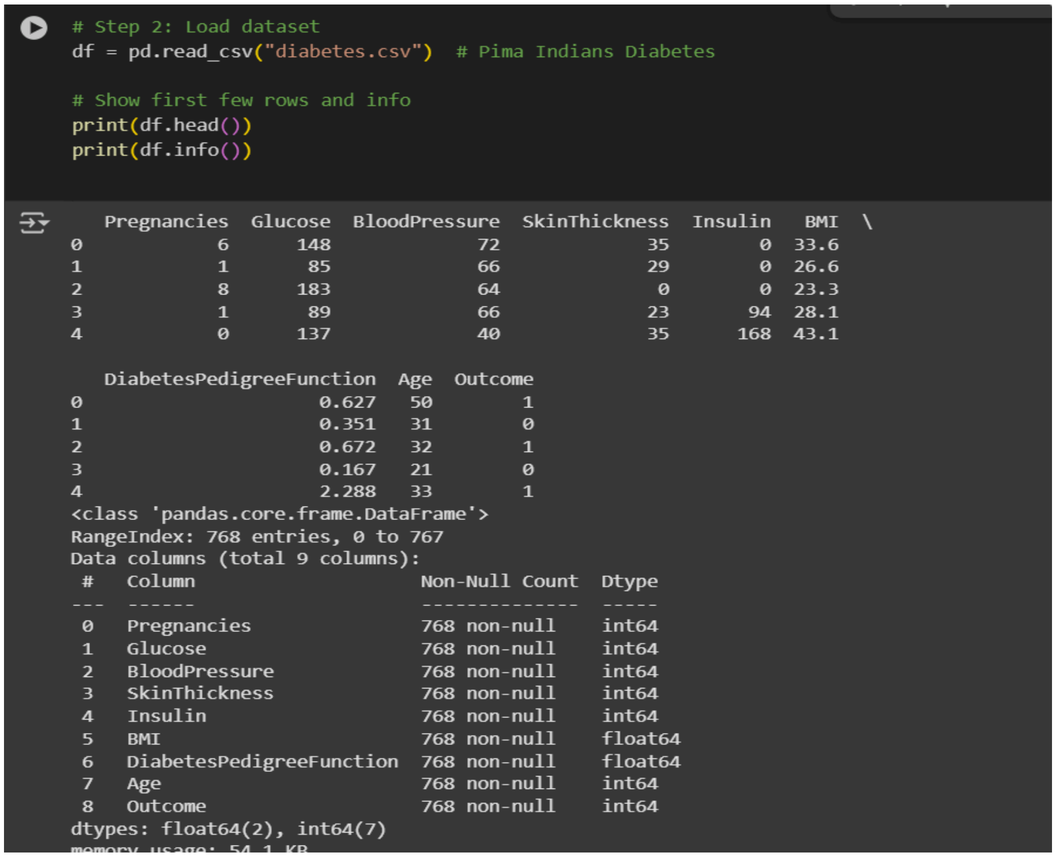
Ans ->

Dataset Used: Pima Indians Diabetes Database



```python
# Step 1: Import required 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
from sklearn.naive_bayes import GaussianNB
from imblearn.over_sampling import SMOTE
```

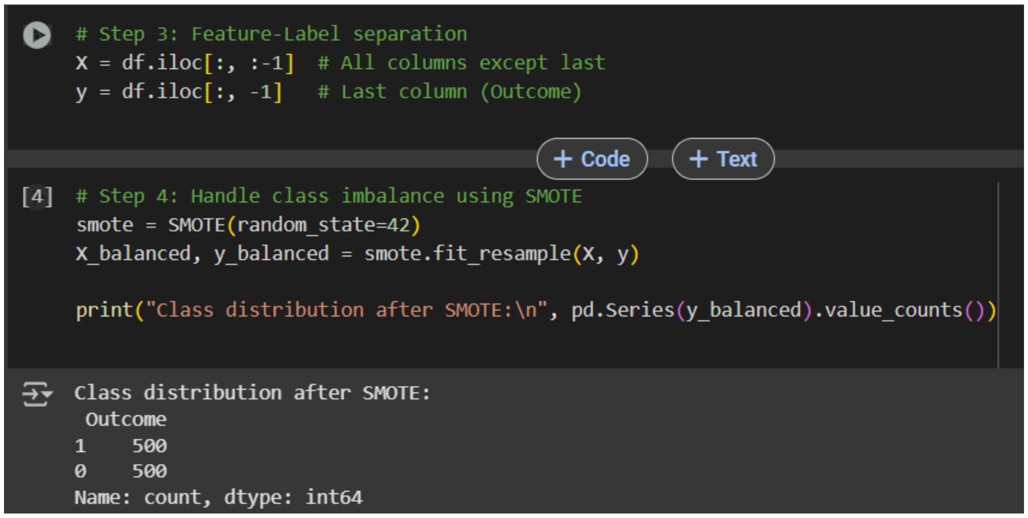
- Import all the required libraries



```
# Step 2: Load dataset
df = pd.read_csv("diabetes.csv")  # Pima Indians Diabetes

# Show first few rows and info
print(df.head())
print(df.info())
```

```
   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
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
```

- In this step, the Pima Indians Diabetes dataset was loaded using pandas. The dataset contains 768 rows and 9 columns, including medical features like Glucose, BMI, Age, and the target variable Outcome. Using head() and info(), it was confirmed that there are no missing values and all columns are numeric. The dataset is clean and ready for further preprocessing and modeling.



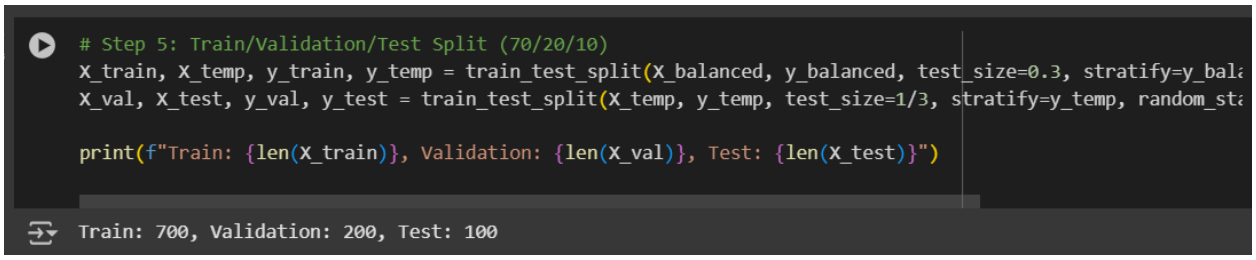
```
# Step 3: Feature-Label separation
X = df.iloc[:, :-1]  # All columns except last
y = df.iloc[:, -1]   # Last column (Outcome)
```

```
[4] # Step 4: Handle class imbalance using SMOTE
    smote = SMOTE(random_state=42)
    X_balanced, y_balanced = smote.fit_resample(X, y)

    print("Class distribution after SMOTE:\n", pd.Series(y_balanced).value_counts())
```

```
Class distribution after SMOTE:
 Outcome
1    500
0    500
Name: count, dtype: int64
```

- In this step, the dataset was split into features (X) and the target variable (y) using .iloc. To address the issue of class imbalance in the target variable Outcome, the SMOTE (Synthetic

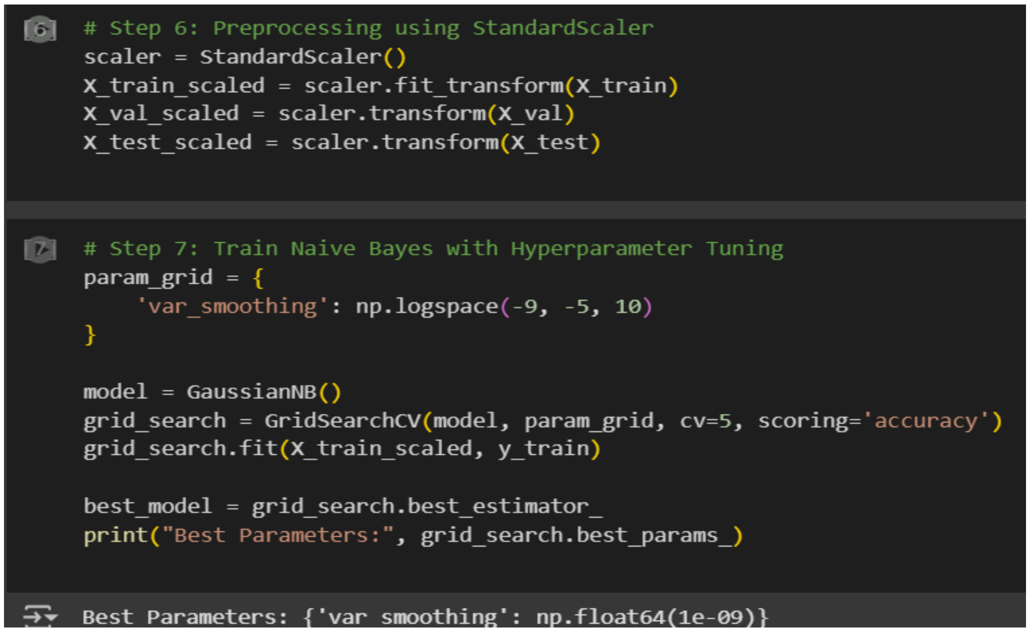
Minority Over-sampling Technique) method was applied. After applying SMOTE, the class distribution was balanced to 500 samples for each class (0 and 1), ensuring that the model does not become biased toward the majority class during training.



```
# Step 5: Train/Validation/Test Split (70/20/10)
X_train, X_temp, y_train, y_temp = train_test_split(X_balanced, y_balanced, test_size=0.3, stratify=y_bala
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=1/3, stratify=y_temp, random_sta

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

```
Train: 700, Validation: 200, Test: 100
```

- In this step, the dataset was split into training, validation, and testing sets using a 70/20/10 ratio. Initially, 70% of the data was allocated to the training set, and the remaining 30% was further split into validation and test sets in a 2:1 ratio. The train_test_split function was used with stratification to preserve the class distribution. As a result, the final data was divided into 700 training samples, 200 validation samples, and 100 testing samples, ensuring a balanced and representative dataset for model training and evaluation.

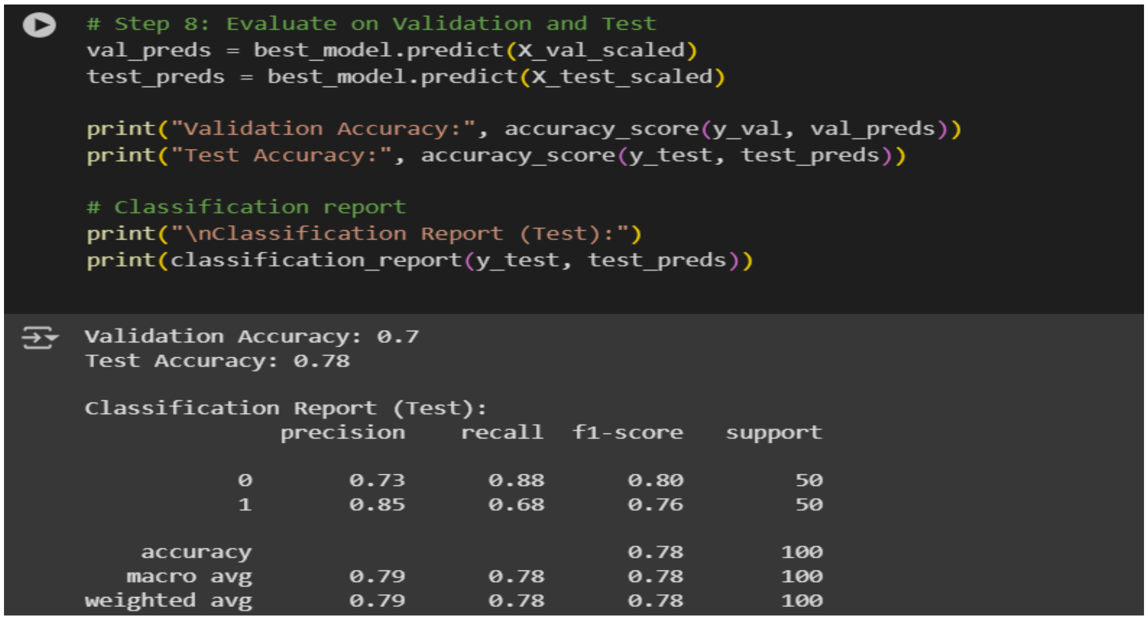


```
# Step 6: Preprocessing using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
X_test_scaled = scaler.transform(X_test)
```

```
# Step 7: Train Naive Bayes with Hyperparameter Tuning
param_grid = {
    'var_smoothing': np.logspace(-9, -5, 10)
}

model = GaussianNB()
grid_search = GridSearchCV(model, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train_scaled, y_train)

best_model = grid_search.best_estimator_
print("Best Parameters:", grid_search.best_params_)
```

```
Best Parameters: {'var_smoothing': np.float64(1e-09)}
```

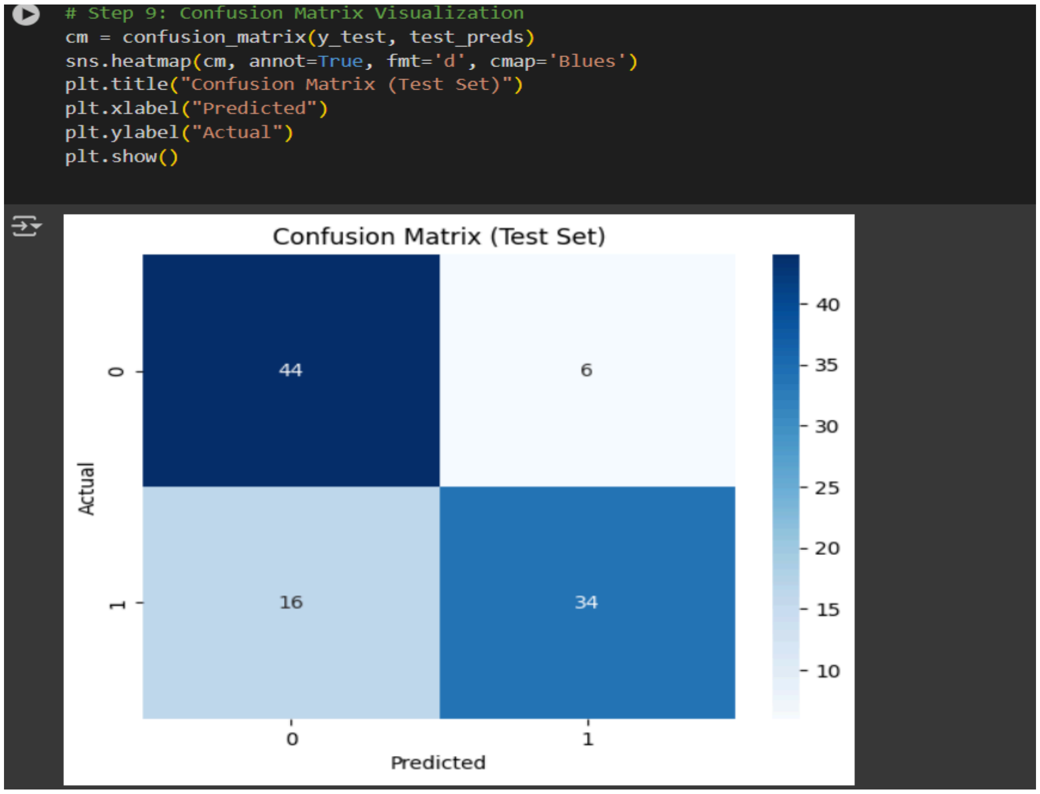
- In this step, data preprocessing was performed using StandardScaler to standardize the features, ensuring they have a mean of 0 and standard deviation of 1, which helps improve model performance. After scaling, a Naive Bayes classifier was trained using GridSearchCV to tune the var_smoothing parameter over a logarithmic range. The best parameter found was var_smoothing = 1e-09, which helps control numerical stability in the model and improves classification accuracy.



```
# Step 8: Evaluate on Validation and Test
val_preds = best_model.predict(X_val_scaled)
test_preds = best_model.predict(X_test_scaled)

print("Validation Accuracy:", accuracy_score(y_val, val_preds))
print("Test Accuracy:", accuracy_score(y_test, test_preds))

# Classification report
print("\nClassification Report (Test):")
print(classification_report(y_test, test_preds))
```

```
Validation Accuracy: 0.7
Test Accuracy: 0.78

Classification Report (Test):
              precision    recall  f1-score   support

           0       0.73      0.88      0.80        50
           1       0.85      0.68      0.76        50

    accuracy                           0.78       100
   macro avg       0.79      0.78      0.78       100
weighted avg       0.79      0.78      0.78       100
```

- In the final step, the trained Naive Bayes model was evaluated on the validation and test sets. The validation accuracy was 70%, while the test accuracy reached 78%. The classification report for the test set showed a good balance between precision and recall, with an overall F1-score of 0.78. This indicates that the model performs reasonably well on unseen data.



```
# Step 9: Confusion Matrix Visualization
cm = confusion_matrix(y_test, test_preds)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix (Test Set)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

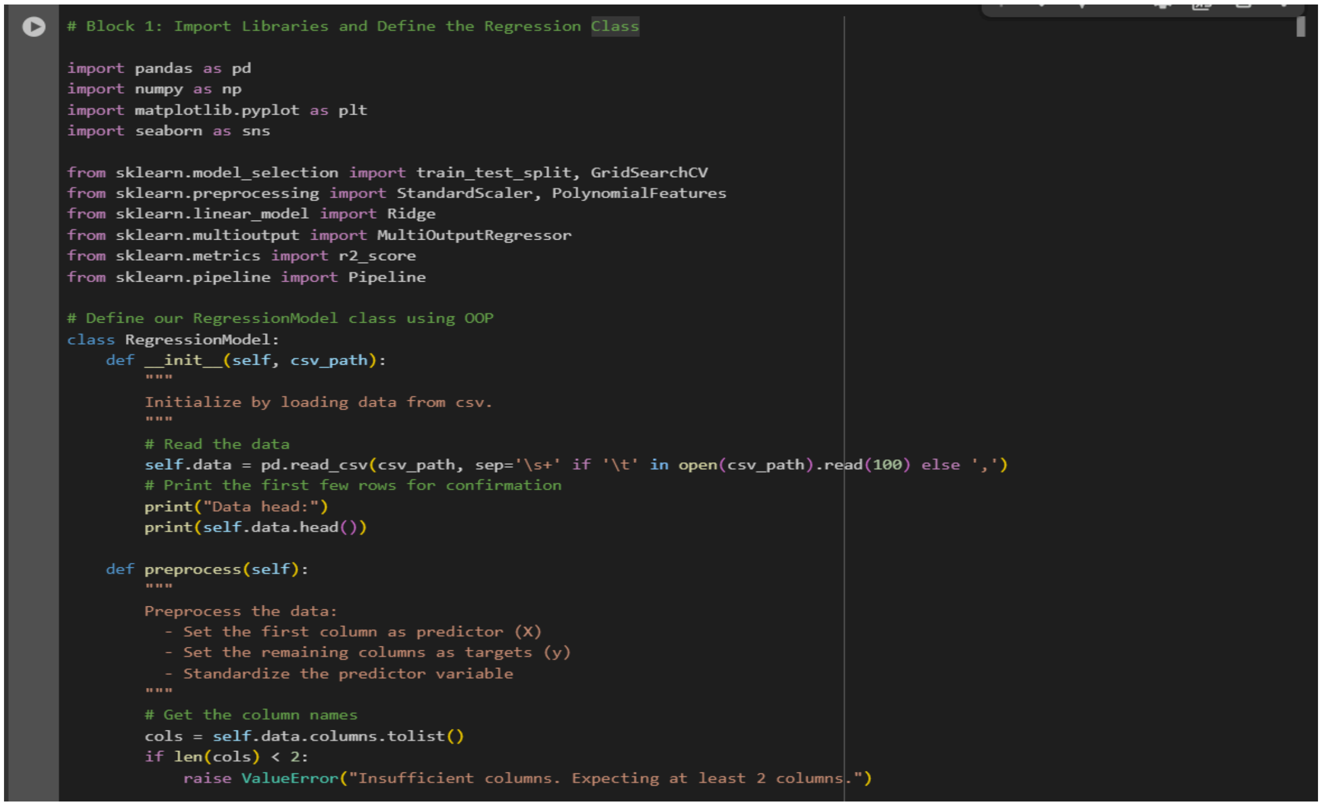
NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

- The confusion matrix visualization in Step 9 offers a deeper insight into the model's performance:

- True Positives (1 predicted as 1): 34
- True Negatives (0 predicted as 0): 44
- False Positives (0 predicted as 1): 6
- False Negatives (1 predicted as 0): 16

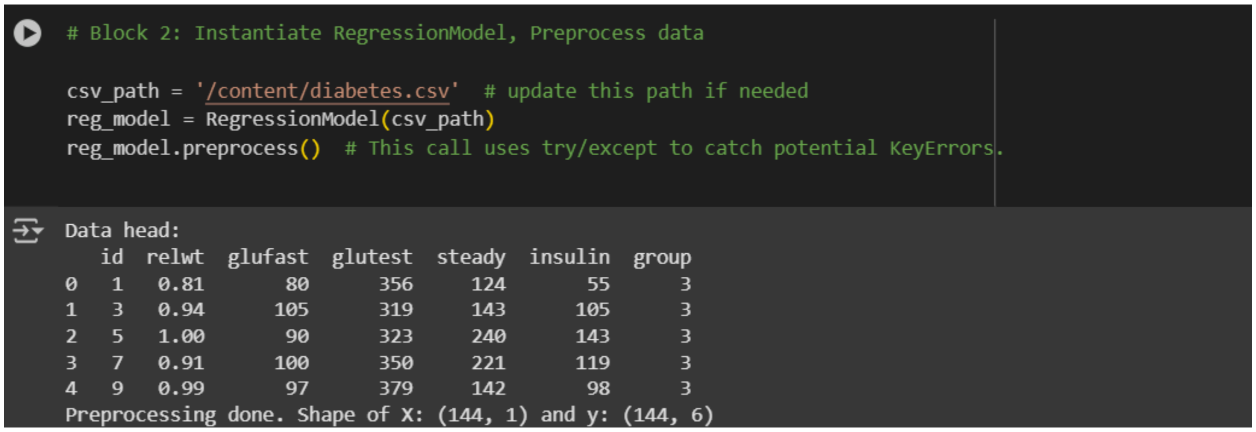
- This shows that the model correctly identified a majority of both classes but is slightly better at predicting class 0. The higher number of false negatives (16) suggests that the model sometimes struggles to detect class 1 instances correctly, which aligns with the recall score for class 1 being lower than that of class 0. Overall, the confusion matrix confirms the classification report and highlights areas for potential improvement, such as reducing false negatives.

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



```
# Block 1: Import Libraries and Define the Regression Class

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, PolynomialFeatures
from sklearn.linear_model import Ridge
from sklearn.multioutput import MultiOutputRegressor
from sklearn.metrics import r2_score
from sklearn.pipeline import Pipeline

# Define our RegressionModel class using OOP
class RegressionModel:
    def __init__(self, csv_path):
        """
        Initialize by loading data from csv.
        """
        # Read the data
        self.data = pd.read_csv(csv_path, sep='\s+' if '\t' in open(csv_path).read(100) else ',')
        # Print the first few rows for confirmation
        print("Data head:")
        print(self.data.head())

    def preprocess(self):
        """
        Preprocess the data:
          - Set the first column as predictor (X)
          - Set the remaining columns as targets (y)
          - Standardize the predictor variable
        """
        # Get the column names
        cols = self.data.columns.tolist()
        if len(cols) < 2:
            raise ValueError("Insufficient columns. Expecting at least 2 columns.")
```

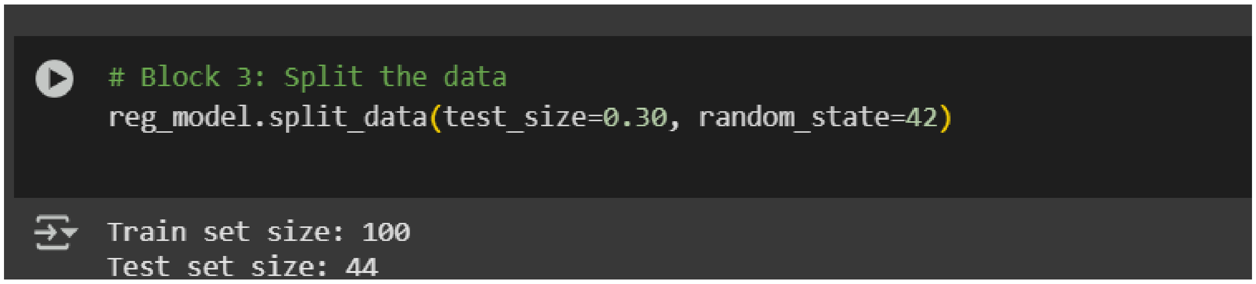
- This code defines a regression pipeline using object-oriented programming. It loads and preprocesses data, splits it into training and test sets, and applies polynomial regression with Ridge regularization. Hyperparameters are tuned using GridSearchCV, and the model handles multiple outputs using MultiOutputRegressor. Finally, it evaluates performance with R^2 and adjusted R^2 scores and prints regression equations for each target variable.



```
# Block 2: Instantiate RegressionModel, Preprocess data

csv_path = '/content/diabetes.csv'  # update this path if needed
reg_model = RegressionModel(csv_path)
reg_model.preprocess()  # This call uses try/except to catch potential KeyErrors.
```

```
Data head:
   id  relwt  glufast  glutest  steady  insulin  group
0   1   0.81       80      356     124       55      3
1   3   0.94      105      319     143      105      3
2   5   1.00       90      323     240      143      3
3   7   0.91      100      350     221      119      3
4   9   0.99       97      379     142       98      3
Preprocessing done. Shape of X: (144, 1) and y: (144, 6)
```

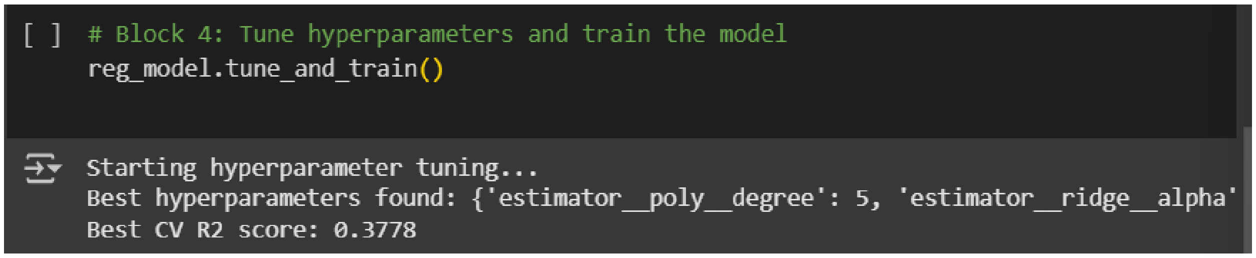
- This block initializes the regression model using the diabetes.csv file and preprocesses the data. The first column (relwt) is used as the predictor (X), and the remaining six columns (glufast, glutest, steady, insulin, group, etc.) are treated as target variables (y). The predictor is also standardized. The output confirms that the data has 144 samples, with X shaped as (144, 1) and y as (144, 6).



```
# Block 3: Split the data
reg_model.split_data(test_size=0.30, random_state=42)
```

```
Train set size: 100
Test set size: 44
```

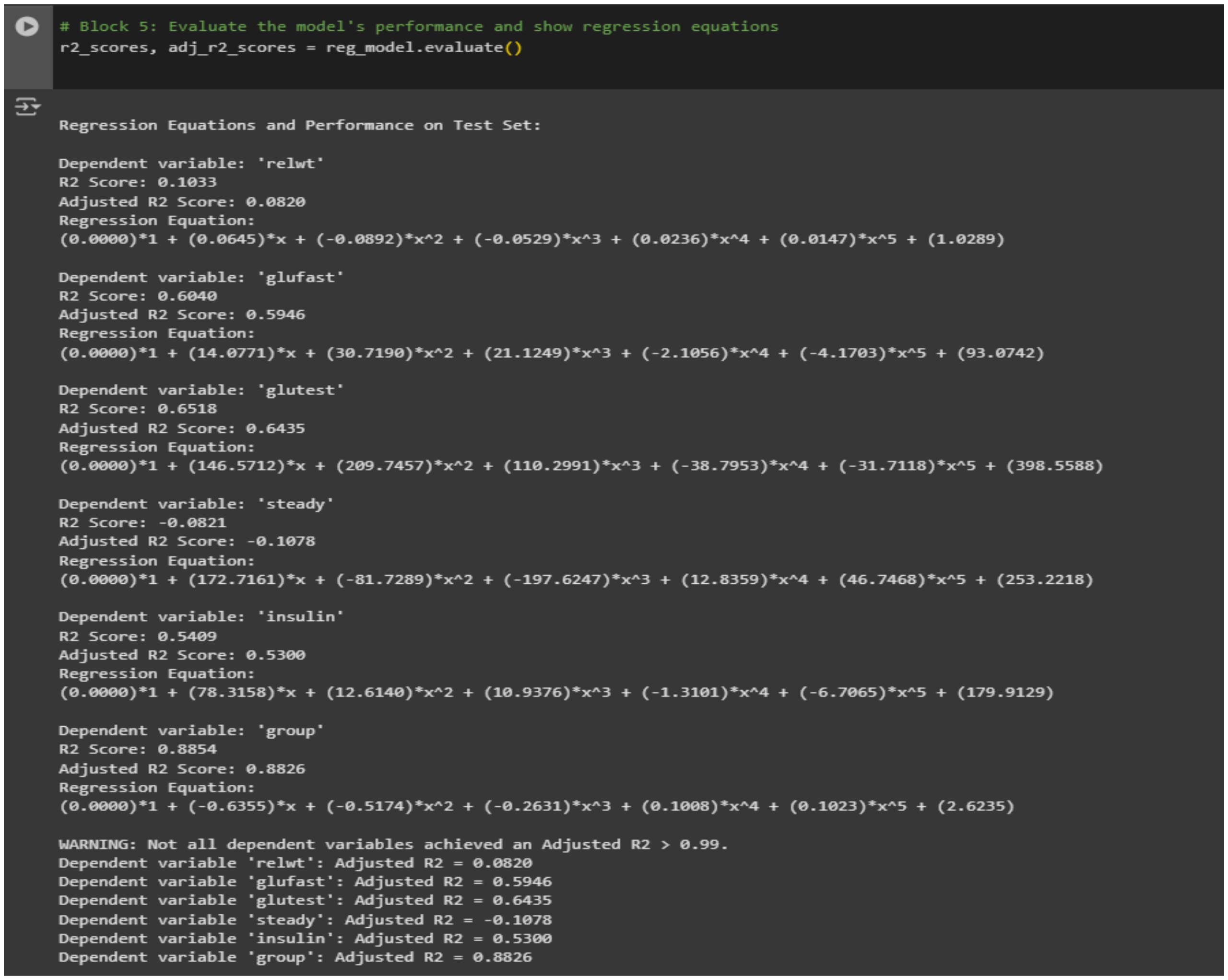
- This block splits the preprocessed data into training and testing sets using a 70-30 ratio. Out of 144 total records, 100 samples are used for training and 44 for testing. The random_state=42 ensures reproducibility of the split.



```
[ ] # Block 4: Tune hyperparameters and train the model
    reg_model.tune_and_train()
```

```
Starting hyperparameter tuning...
Best hyperparameters found: {'estimator__poly__degree': 5, 'estimator__ridge__alpha'
Best CV R2 score: 0.3778
```

- This block performs hyperparameter tuning using GridSearchCV. It finds the best combination of polynomial degree and Ridge alpha for the regression model. The best parameters selected are degree 5 and alpha 10, with a cross-validation R^2 score of 0.3778.



```
# Block 5: Evaluate the model's performance and show regression equations
r2_scores, adj_r2_scores = reg_model.evaluate()
```

```
Regression Equations and Performance on Test Set:

Dependent variable: 'relwt'
R2 Score: 0.1033
Adjusted R2 Score: 0.0820
Regression Equation:
(0.0000)*1 + (0.0645)*x + (-0.0892)*x^2 + (-0.0529)*x^3 + (0.0236)*x^4 + (0.0147)*x^5 + (1.0289)

Dependent variable: 'glufast'
R2 Score: 0.6040
Adjusted R2 Score: 0.5946
Regression Equation:
(0.0000)*1 + (14.0771)*x + (30.7190)*x^2 + (21.1249)*x^3 + (-2.1056)*x^4 + (-4.1703)*x^5 + (93.0742)

Dependent variable: 'glutest'
R2 Score: 0.6518
Adjusted R2 Score: 0.6435
Regression Equation:
(0.0000)*1 + (146.5712)*x + (209.7457)*x^2 + (110.2991)*x^3 + (-38.7953)*x^4 + (-31.7118)*x^5 + (398.5588)

Dependent variable: 'steady'
R2 Score: -0.0821
Adjusted R2 Score: -0.1078
Regression Equation:
(0.0000)*1 + (172.7161)*x + (-81.7289)*x^2 + (-197.6247)*x^3 + (12.8359)*x^4 + (46.7468)*x^5 + (253.2218)

Dependent variable: 'insulin'
R2 Score: 0.5409
Adjusted R2 Score: 0.5300
Regression Equation:
(0.0000)*1 + (78.3158)*x + (12.6140)*x^2 + (10.9376)*x^3 + (-1.3101)*x^4 + (-6.7065)*x^5 + (179.9129)

Dependent variable: 'group'
R2 Score: 0.8854
Adjusted R2 Score: 0.8826
Regression Equation:
(0.0000)*1 + (-0.6355)*x + (-0.5174)*x^2 + (-0.2631)*x^3 + (0.1008)*x^4 + (0.1023)*x^5 + (2.6235)

WARNING: Not all dependent variables achieved an Adjusted R2 > 0.99.
Dependent variable 'relwt': Adjusted R2 = 0.0820
Dependent variable 'glufast': Adjusted R2 = 0.5946
Dependent variable 'glutest': Adjusted R2 = 0.6435
Dependent variable 'steady': Adjusted R2 = -0.1078
Dependent variable 'insulin': Adjusted R2 = 0.5300
Dependent variable 'group': Adjusted R2 = 0.8826
```

This block evaluates the model's performance on the test set for each dependent variable using R^2 and Adjusted R^2 scores. Here's a short summary:

- Best model performance was for the group variable (Adjusted R^2: 0.8262).
- Weakest performance was for steady (Adjusted R^2: -0.1078).
- All models used a 5-degree polynomial regression.

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).
Ans ->

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

Key Features of the Wine Quality Dataset

The Wine Quality Dataset contains physicochemical properties of red or white wine samples. Each row in the dataset is a different wine sample, and the target variable is the quality score (rated from 0 to 10).

Feature Name	Description
fixed acidity	Non-volatile acids that do not evaporate during fermentation.
volatile acidity	Amount of acetic acid; high levels can result in an unpleasant taste.
citric acid	Adds freshness and flavor; small quantities are usually favorable.
residual sugar	Sugar remaining after fermentation; affects sweetness.
chlorides	Salt content; affects taste and stability of wine.
free sulfur dioxide	Protects wine from microbial growth and oxidation.
total sulfur dioxide	Sum of free and bound forms; high levels can affect taste and health.
density	Affected by sugar and alcohol content; used to monitor fermentation.
pH	Indicates acidity level; affects taste and microbial stability.
sulphates	Used as an antioxidant and antimicrobial agent.
alcohol	Alcohol content of wine; typically has a strong positive correlation with quality.
quality (target)	Score assigned to the wine sample (0–10) based on sensory data.

Importance of Features in Predicting Wine Quality

- Alcohol: Positively correlated with quality. Higher alcohol usually improves flavor.
- Volatile Acidity: Negative impact; too much leads to vinegar-like taste.
- Sulphates & Citric Acid: Moderate to high correlation; help in preservation and flavor enhancement.
- pH & Chlorides: Can influence taste, but usually show lower correlation with quality.
- Residual Sugar: Important for sweetness but has lower impact unless extreme.
- Fixed Acidity, Density: Help in fermentation monitoring; moderate importance.

Feature importance can be confirmed through correlation analysis and model-based feature selection techniques (e.g., feature importance from Random Forest).

Handling Missing Data During Feature Engineering

In the original Kaggle dataset, missing values are rare or often not present. However, if missing data does exist, it can be handled in several ways:

Steps Taken:

NAME: VEDANT DHOKE CLASS/ROLL NO: D15C/ 9

1.Data Exploration:
Checked for NaN or missing values using df.isnull().sum()

2.Visual Inspection:
Used heatmaps (e.g., Seaborn) to visualize missing data patterns.

3. Applied Imputation Techniques:
For numerical features, used mean/median imputation.
For non-linear relationships, used K-Nearest Neighbors (KNN) imputation.

Advantages and Disadvantages of Imputation Techniques

Technique	Advantages	Disadvantages
Mean/Median Imputation	Simple, fast, works well with symmetric distributions	Can reduce variance; not suitable for skewed or non-normal distributions
Mode Imputation	Works well for categorical data	Not useful for continuous/numerical features
KNN Imputation	Uses data similarity; retains data variance	Computationally expensive, sensitive to outliers
Regression Imputation	Uses correlations between features to predict missing values	Can introduce bias if model is overfit or assumptions are incorrect
Multiple Imputation	Accounts for uncertainty; gives multiple datasets for better estimation	Complex and computationally intensive
Dropping Rows	Easy and clean	Can lead to data loss if many rows are missing