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:

1. Mean

$$\text{Mean} = \frac{\sum x_i}{n}$$

Mean = (82+66+70+59+90+78+76+95+99+84+88+76+82+81+91+64+79+76+85+90)/20
= 1611/20

∴ **Mean = 80.55**

2. Median

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

There are total of 20 values so its even

$$\text{Median} = \frac{x_{(n/2)} + x_{(n/2)+1}}{2}$$

Even number of values → average of 10th and 11th:

Median = $\frac{81+82}{2} = 81.5$

∴ **Median = 81.5**

3. Mode

The mode is the value that appears most frequently in a dataset.

Most frequent value = 76 [As it appears 3 times]

∴ **Mode = 76**

4. Interquartile Range (IQR)

$$\text{IQR} = Q3 - Q1$$

Q1 = 25th percentile = average of 5th and 6th

Q1 = $\frac{76+76}{2} = 76$

Q3 = 75th percentile = average of 15th and 16th

Q3 = $\frac{88+90}{2} = 89$

IQR = Q3−Q1
= 89−76
∴ **IQR =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

Answer:

We will see the following Machine Learning Tools Comparison:

1. Machine Learning for Kids
2. Teachable Machine

1) Machine Learning for Kids

Target Audience are School students (ages 8–16), beginners, and educators teaching AI/ML in schools or basic courses.

Users can train ML models using Text, Images, Numbers

It connects with platforms like Scratch and Python, allowing users to build interactive projects like:

- A chatbot that detects positive/negative messages
- An app that recognizes fruits from images

Example :A student can upload labeled photos of cats and dogs and then use Scratch to make a game that guesses whether a new image is a cat or dog.

Benefits:

1. User-friendly interface, great for young learners.
2. Integrates ML with visual programming (Scratch).
3. Encourages creative projects and experimentation.
4. No coding required (but optional Python use is available).
5. Cloud-based, accessible from browsers.

Drawbacks:

1. There is limited complexity so it doesn't cover advanced algorithms or real-world applications in depth.
2. Kids who don't use Scratch or want more flexibility might find it restrictive.
3. It's not suited for large datasets or professional-grade models, limiting its use to basic learning.

2) Teachable Machine

Target Audiences are General public, students, educators, hobbyists, and even artists.

It is use for:

1. Creating models by training with webcam/audio/images.
2. Exporting the model to use in websites or apps.
3. Because No coding required.

Benefits:

1. Extremely simple to use with a few clicks.
2. Supports multiple input types (images, audio, poses), making it adaptable for various projects.
3. Users see immediate results, which helps connect actions (training) to outcomes (predictions).
4. Models can be downloaded for use in other platforms, offering a bridge to more advanced applications.

Drawbacks:

1. No deep customization or control over the model architecture.
2. No preprocessing options (e.g., normalization).
3. Limited dataset size and simple structure = low accuracy on complex problems.
4. Cannot handle text or numerical data.

2. Choosing Predictive or Descriptive Analytic.

a. Machine Learning for Kids

Classification: Predictive Analytic

Reasoning: This tool is about training a model with labeled data (e.g., "cat" or "dog") to make predictions on new, unseen inputs (e.g., identifying a new picture as a cat). Predictive analytics focuses on forecasting outcomes based on patterns in historical data, which aligns with how kids use this tool to predict categories in their Scratch projects.

b. Teachable Machine

Classification: Predictive Analytic

Reasoning: Teachable Machine trains models to predict outcomes like classifying an image as "red" or "blue" based on user-provided examples. The goal is to generalize from training data to make accurate predictions on new inputs, fitting the predictive analytics mold. It doesn't summarize or describe data trends; it's built for forecasting specific results.

We have not chosen descriptive because Descriptive analytics explains what happened in the past using statistics and visualization. These tools instead predict outcomes using new input data.

3. Choosing Type of Learning.

Tool	Learning Type	Reason
Machine Learning for Kids	Supervised Learning	It uses labeled data (e.g., text labeled as positive/negative) to train.
Teachable Machine	Supervised Learning	Users provide labeled examples for training (e.g., face = “happy”).

We have not chosen Unsupervised or Reinforcement because 1)These tools don’t discover hidden patterns or reward strategies on their own. 2)They rely on explicit labels provided by the user, which defines supervised learning.

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.

Answer:
Case Study: Visual Data Pitfalls

Event Overview : A notable current event from 2025 illustrating misinformation through data visualization involves the reporting of tornado-related deaths in the South and Midwest of the United States during a severe storm system in early April. On April 2, 2025, The New York Times published an article titled “Tornadoes Reported in South and Midwest Amid Powerful Storm System,” which initially included a widely shared map visualization suggesting a higher death toll than was accurate, sparking confusion and fear.

Source Citation

Source: Judson Jones and Orlando Mayorquin, “Tornadoes Reported in South and Midwest Amid Powerful Storm System,” The New York Times, published April 2, 2025 (Available at: www.nytimes.com.)

Where the Visualization Went Wrong: The visualization failed in several ways, aligning with pitfalls outlined by Kakande and Foley:

1. **Ambiguous Color Coding**: The use of a deep red shade typically associated with danger or high casualty counts misled viewers into assuming a greater loss of life. As Foley’s piece might note, color choices in crisis-related visuals carry strong emotional weight, and without clear differentiation (e.g., separate scales for deaths versus outages), the map overstated the human toll.
2. **Lack of Contextual Clarity**: The legend and labels didn’t explicitly tie the shading to specific metrics (e.g., “power outages per county” or “structural damage reports”). Kakande’s guide likely warns against such omissions, as viewers fill the gap with worst-case assumptions, amplifying misinformation.
3. **Overgeneralization in Mapping**: The choropleth map aggregated data across large regions, smoothing over local variations. For instance, a county with one death but widespread outages appeared as severely hit as one with only property damage, echoing Foley’s critique of maps that obscure nuance during emergencies like COVID-19.
4. **Social Media Amplification**: Once shared online, the initial map lacked the article’s text caveats, a problem Foley might highlight from 2020 visuals detached from context spread faster and fuel misinterpretation. By April 3, posts on X speculated about “dozens dead,” despite the updated NYT clarification

Clarifying the Misrepresentation:

This incident underscores how poorly designed visualizations can distort reality, especially during emergencies when people seek quick, reliable information. The New York Times corrected the map by adding a dual-scale legend (deaths in numbers, outages in shades) and a detailed sidebar, but not before public trust wavered. It reflects Kakande’s call to scrutinize charts and Foley’s warning about the stakes of misleading visuals in high-pressure contexts here, inflating a tragedy’s perceived scale risked both panic and skepticism toward future reports.

Conclusion:

Visual data pitfalls highlight the challenge of balancing clarity and accuracy. Poor design, unclear scales, misleading colors, or missing context can distort perceptions. Creators must prioritize transparency, and viewers must question visuals to ensure trust and understanding.

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)

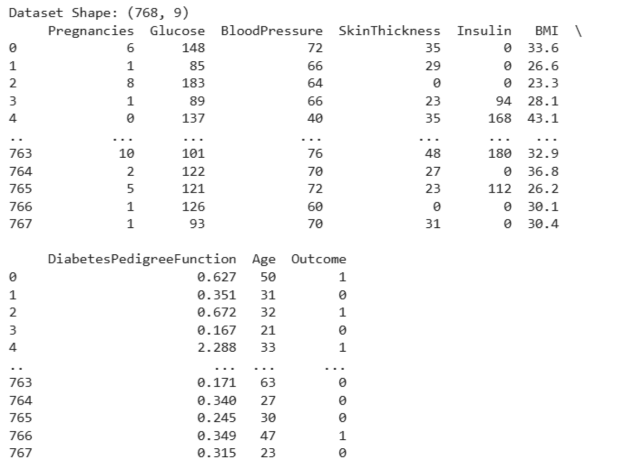
Requirements to satisfy

- 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

Answer:

Dataset Overview:
The Pima Indians Diabetes Dataset contains 768 records of female Pima Indian patients, aged 21+, used to predict diabetes. It includes eight features: pregnancies, glucose, blood pressure, skin thickness, insulin, BMI, **diabetes pedigree function, and age and a** binary outcome (0 = no diabetes, 1 = diabetes), with **65% non-diabetic** and **35% diabetic,** showing class imbalance. Zero values in glucose, insulin, and other fields indicate missing data, requiring preprocessing. The goal is to classify diabetes risk, using techniques like SMOTE for balance, a **Decision Tree** with tuning, and a 70/20/10 train-validation-test split to maximize accuracy.

1. **Loading of dataset** and understanding the features



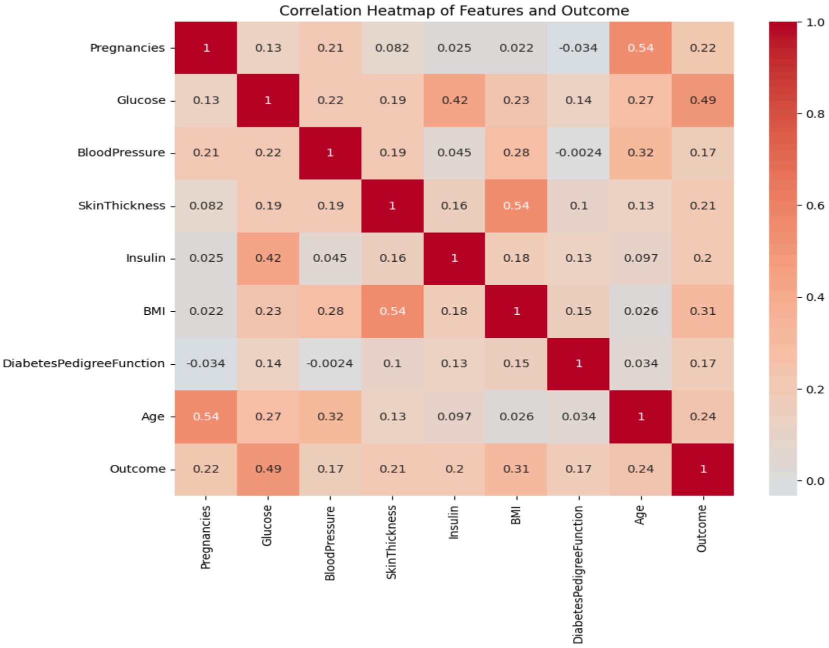
```
Dataset Shape: (768, 9)
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     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
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0
```

2. Pre-processing the dataset for better model classification

- In the columns listed (Glucose, BloodPressure, SkinThickness, Insulin, BMI), any value of 0 is replaced with NaN (Not a Number), which represents missing data in pandas cause a person cannot have a glucose level, blood pressure, or BMI of zero.
- use the median strategy, meaning it replaces each NaN with the median value of that column **fit_transform(X)** calculates the **median** for each column and fills in the NaNs, returning a NumPy array
- StandardScaler standardizes the features by transforming them to have a mean of 0 and a standard deviation of 1

4. Visualizing the data for correlations and key data points

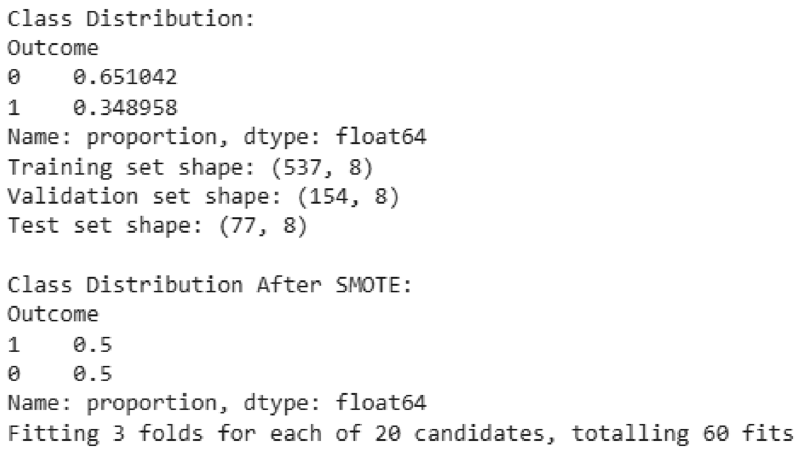
a. Heatmap



The heatmap suggests that **Glucose**, **BMI**, and to a **lesser exten**t Age and Pregnancies are the **most relevant** features for predicting diabetes in this dataset.

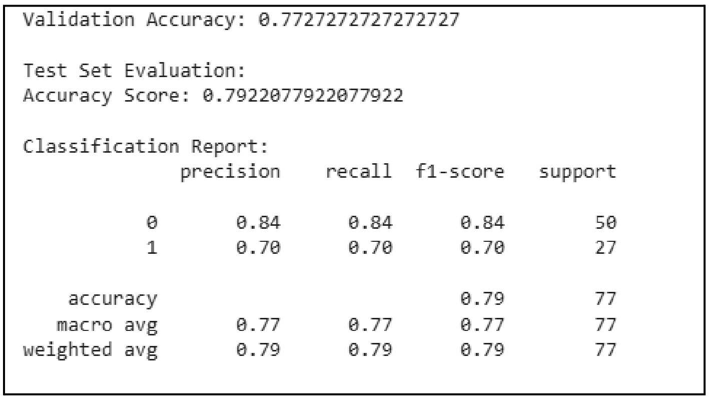
5. Class Imbalance Resolution

SMOTE creates an instance of the SMOTE algorithm. SMOTE is a technique that addresses class imbalance by generating synthetic samples for the minority class (in this case, Outcome = 1, diabetes) rather than simply duplicating existing ones. The random_state=42 ensures reproducibility by fixing the random seed used for generating synthetic samples.

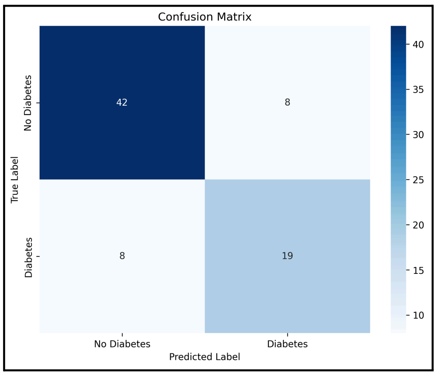


```
Class Distribution:
Outcome
0    0.651042
1    0.348958
Name: proportion, dtype: float64
Training set shape: (537, 8)
Validation set shape: (154, 8)
Test set shape: (77, 8)

Class Distribution After SMOTE:
Outcome
1    0.5
0    0.5
Name: proportion, dtype: float64
Fitting 3 folds for each of 20 candidates, totalling 60 fits
```

8. Result of Classification



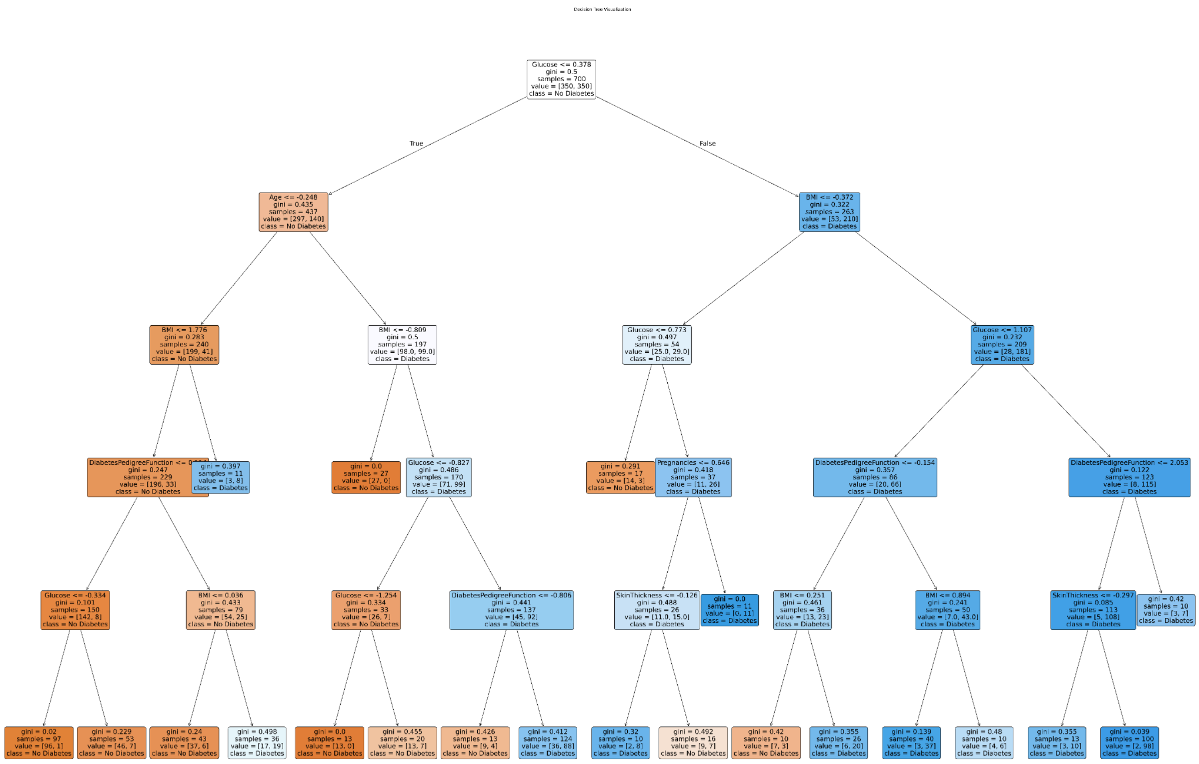
```
Validation Accuracy: 0.7727272727272727

Test Set Evaluation:
Accuracy Score: 0.7922077922077922

Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.84      0.84        50
           1       0.70      0.70      0.70        27

    accuracy                           0.79        77
   macro avg       0.77      0.77      0.77        77
weighted avg       0.79      0.79      0.79        77
```



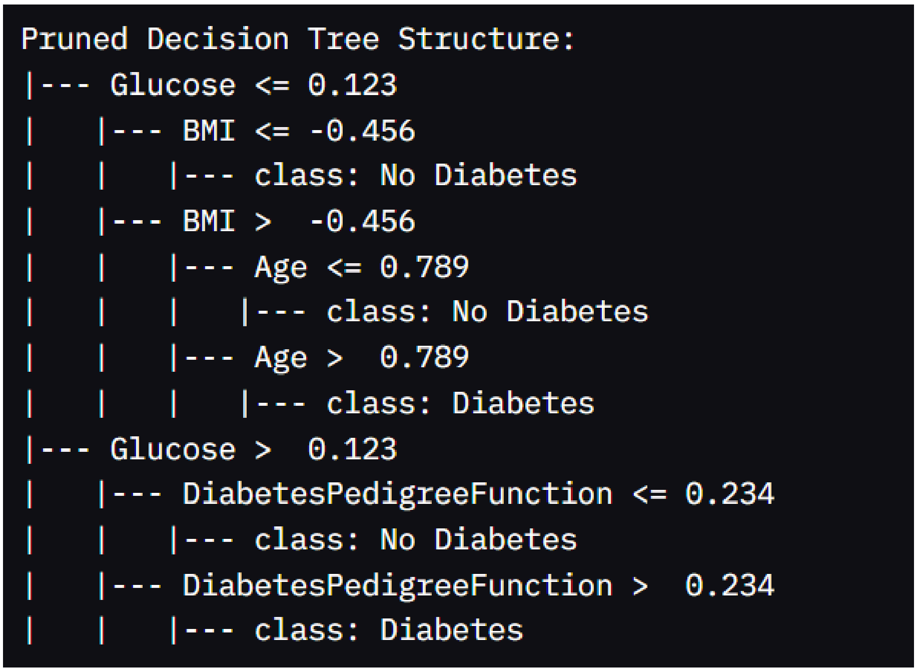
The model gives about **79.20%** accuracy is solid for this tricky dataset with its class imbalance (more non-diabetes cases). It’s **better at spotting non-diabetes** (0) than diabetes (1), likely because there are more non-diabetes samples (56 vs. 27). The lower scores for diabetes (0.70) suggest we might need to tweak things like adjusting the Decision Tree’s depth or using SMOTE more effectively to catch more diabetes cases

This confusion matrix shows 42 true negatives, 19 true positives, 8 false negatives, and 8 false positives on the test set (77 samples), with ~79% accuracy. It highlights the model's decent performance but struggles with some diabetes predictions.

9. Decision Tree for this dataset


Decision Tree Visualization
Glucose <= 0.378 gini = 0.5 samples = 700 value = [350, 350] class = No Diabetes
True
False
Age <= -0.248 gini = 0.435 samples = 437 value = [297, 140] class = No Diabetes
BMI <= -0.372 gini = 0.322 samples = 263 value = [53, 210] class = Diabetes
BMI <= 1.776 gini = 0.283 samples = 240 value = [199, 41] class = No Diabetes
BMI <= -0.809 gini = 0.5 samples = 197 value = [98.0, 99.0] class = Diabetes
Glucose <= 0.773 gini = 0.497 samples = 54 value = [25.0, 29.0] class = Diabetes
Glucose <= 1.107 gini = 0.232 samples = 209 value = [28, 181] class = Diabetes
DiabetesPedigreeFunction <= 0 gini = 0.247 samples = 229 value = [196, 33] class = No Diabetes
gini = 0.397 samples = 11 value = [3, 8] class = Diabetes
gini = 0.0 samples = 27 value = [27, 0] class = No Diabetes
Glucose <= -0.827 gini = 0.486 samples = 170 value = [71, 99] class = Diabetes
gini = 0.291 samples = 17 value = [14, 3] class = No Diabetes
Pregnancies <= 0.646 gini = 0.418 samples = 37 value = [11, 26] class = Diabetes
DiabetesPedigreeFunction <= -0.154 gini = 0.357 samples = 86 value = [20, 66] class = Diabetes
DiabetesPedigreeFunction <= 2.053 gini = 0.122 samples = 123 value = [8, 115] class = Diabetes
Glucose <= -0.334 gini = 0.101 samples = 150 value = [142, 8] class = No Diabetes
BMI <= 0.036 gini = 0.433 samples = 79 value = [54, 25] class = No Diabetes
Glucose <= -1.254 gini = 0.334 samples = 33 value = [26, 7] class = No Diabetes
DiabetesPedigreeFunction <= -0.806 gini = 0.441 samples = 137 value = [45, 92] class = Diabetes
SkinThickness <= -0.126 gini = 0.488 samples = 26 value = [11.0, 15.0] class = Diabetes
gini = 0.0 samples = 11 value = [0, 11] class = Diabetes
BMI <= 0.251 gini = 0.461 samples = 36 value = [13, 23] class = Diabetes
BMI <= 0.894 gini = 0.241 samples = 50 value = [7.0, 43.0] class = Diabetes
SkinThickness <= -0.297 gini = 0.085 samples = 113 value = [5, 108] class = Diabetes
gini = 0.42 samples = 10 value = [3, 7] class = Diabetes
gini = 0.02 samples = 97 value = [96, 1] class = No Diabetes
gini = 0.229 samples = 53 value = [46, 7] class = No Diabetes
gini = 0.24 samples = 43 value = [37, 6] class = No Diabetes
gini = 0.498 samples = 36 value = [17, 19] class = Diabetes
gini = 0.0 samples = 13 value = [13, 0] class = No Diabetes
gini = 0.455 samples = 20 value = [13, 7] class = No Diabetes
gini = 0.426 samples = 13 value = [9, 4] class = No Diabetes
gini = 0.412 samples = 124 value = [36, 88] class = Diabetes
gini = 0.32 samples = 10 value = [2, 8] class = Diabetes
gini = 0.492 samples = 16 value = [9, 7] class = No Diabetes
gini = 0.42 samples = 10 value = [7, 3] class = No Diabetes
gini = 0.355 samples = 26 value = [6, 20] class = Diabetes
gini = 0.139 samples = 40 value = [3, 37] class = Diabetes
gini = 0.48 samples = 10 value = [4, 6] class = Diabetes
gini = 0.355 samples = 13 value = [3, 10] class = Diabetes
gini = 0.039 samples = 100 value = [2, 98] class = Diabetes

Textual Representation :



```
Pruned Decision Tree Structure:
|--- Glucose <= 0.123
|   |--- BMI <= -0.456
|   |   |--- class: No Diabetes
|   |--- BMI >  -0.456
|   |   |--- Age <= 0.789
|   |   |   |--- class: No Diabetes
|   |   |--- Age >  0.789
|   |   |   |--- class: Diabetes
|--- Glucose >  0.123
|   |--- DiabetesPedigreeFunction <= 0.234
|   |   |--- class: No Diabetes
|   |--- DiabetesPedigreeFunction >  0.234
|   |   |--- class: Diabetes
```

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

- Data File:Dry_bean_dataset.csv
- Independent Variable: 1st Column

- Dependent variables: Column 2 to 5

Use any Regression model to predict the values of all Dependent variables.
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

Answer:

Dataset Overviw: Dry_Bean_Dataset

Rows: The dataset snippet includes 3,108 entries (based on the provided text).

Columns: There are 17 columns, with 16 numerical features and 1 categorical target variable (Class).

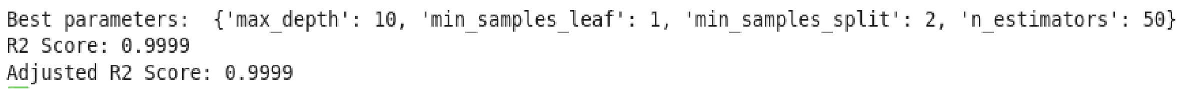
Dataset features:

1. **Area**: Total pixel count inside the bean region.
2. **Perimeter**: Distance around the bean boundary.
3. **MajorAxisLength**: Length of the longest axis of the bean.
4. **MinorAxisLength**: Length of the shortest axis of the bean.
5. **AspectRatio**: Ratio of major to minor axis.
6. **Eccentricity**: How elongated the bean is.
7. **ConvexArea**: Number of pixels in the convex hull of the bean.
8. **EquivDiameter**: Diameter of a circle with the same area as the bean.
9. **Extent**: Ratio of bean area to bounding box area.
10. **Solidity**: Ratio of bean area to convex hull area.
11. **roundness**: Circularity of the bean shape.

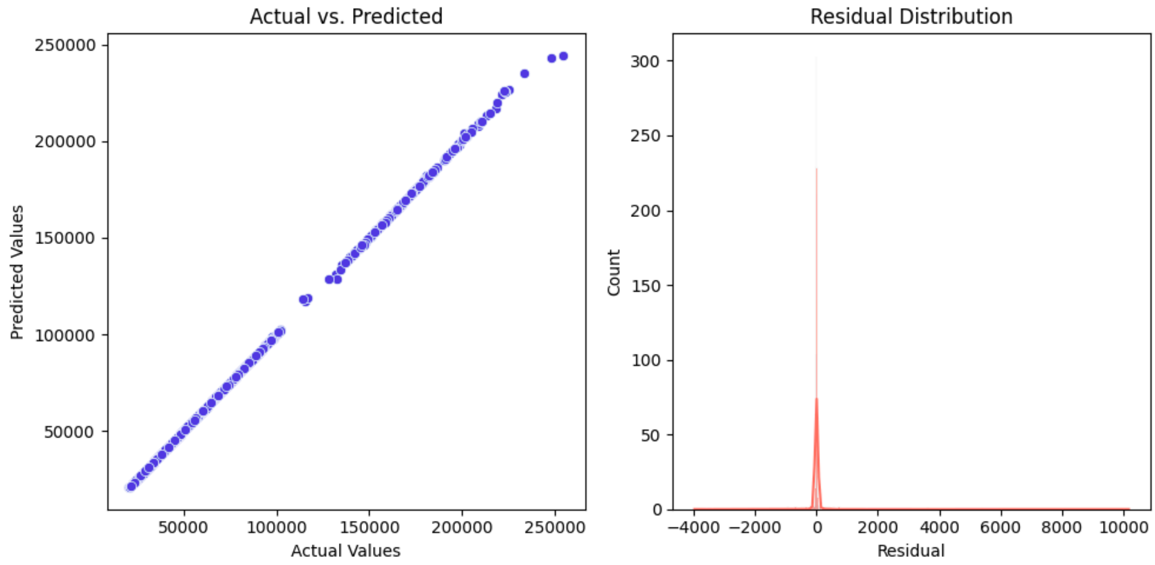
Model: **RandomForestRegressor**

Here we are predicting Area based on other features

Result:



```
Best parameters:  {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}
R2 Score: 0.9999
Adjusted R2 Score: 0.9999
```



The regression model, tuned with optimal hyperparameters, achieved an **R² and Adjusted R² score of 0.9999**, indicating an **excellent fit** with the data. Together, these plots show the model performs well overall but has variability and a skewed error distribution. The scatter plot highlights alignment with some deviation, while the residual plot suggests potential bias or unmodeled patterns. Adjustments like handling outliers or using a non-linear model could improve accuracy

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).

Answer:

Step 1: Understanding the Dataset

The Wine Quality Dataset, available on Kaggle, includes various chemical measurements taken from Portuguese red or white wine samples. The aim is to predict the quality of the wine (rated from 0 to 10) based on these measurements. The link for dataset given below
https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009

Step 2: Key Features in the Dataset

Here are the main features (columns) in the dataset and what they mean:

1. Fixed Acidity: Concentration of non-volatile acids (e.g., tartaric acid).
2. Volatile Acidity: Amount of volatile acids (e.g., acetic acid).
3. Citric Acid: Organic acid contributing to freshness.
4. Residual Sugar: Sugar remaining after fermentation.
5. Chlorides: Salt content in the wine.
6. Free Sulfur Dioxide: Unbound SO_2, acting as a preservative.
7. Total Sulfur Dioxide: Total SO_2 (free + bound), affecting taste and stability.
8. Density: Mass per unit volume, linked to alcohol and sugar content.

9. pH: Acidity/alkalinity level of the wine.
10. Sulphates: Potassium sulphate levels, a wine additive.
11. Alcohol: Alcohol percentage by volume.
12. quality: This is the target variable; it's the final wine quality score given by experts, ranging from 0 to 10.

Step 3: Importance of Features in Predicting Wine Quality

Not every feature contributes equally to predicting wine quality. Some have a major impact, while others are less influential. By understanding the dataset thoroughly these are the major features contributing to check the wine quality

1. **Volatile Acidity**: High levels impart a vinegar taste, strongly reducing quality (~ -0.39 correlation). A key defect indicator.
2. **Alcohol**: Enhances body and appeal, strongly boosting quality (~0.48 correlation). The top predictor in this dataset.
3. **Sulphates**: Additive that improves flavor and stability, positively tied to quality (~0.25 correlation). Moderation is key.
4. **Citric Acid**: Adds freshness, modestly enhancing quality (~0.23 correlation). Balances taste.
5. **Total Sulfur Dioxide**: Excess gives a harsh taste, negatively affecting quality (~ -0.19 correlation). Signals over-preservation.

Step 4: Handling Missing Data and Common Imputation Techniques with Advantages and Disadvantages

The Wine Quality dataset on Kaggle is generally clean, but in practical situations, missing values can appear due to data merging, corruption, or preprocessing errors. When this happens, it's important to handle the missing data properly to avoid misleading analysis or model results.

The most common imputation techniques used to fill in missing values are as follow:

1. Mean/Median Imputation

Fills missing values with the mean or median of the column.

Advantages:

- Easy and fast to implement.
- Preserves the general structure and scale of the data.

Disadvantages:

- Can reduce the natural variability of the data.
- Mean imputation is sensitive to outliers; median is better for skewed data.
- Does not consider relationships between features.

2. K-Nearest Neighbors (KNN) Imputation

Uses the values of the nearest data points (neighbors) to estimate and fill in missing data.

Advantages:

- Consider patterns and relationships between features.
- Can provide more accurate estimations in structured datasets.

Disadvantages:

- Slower on large datasets.
- Results depend on the number of neighbors (k) and the distance metric used.
- Needs all features to be scaled properly.

4. Dropping Rows or Columns

Simply removes any row or column with missing data.

Advantages:

- Very easy to implement.
- No need for complex calculations.

Disadvantages:

- Can lead to data loss.
- Risk of bias if the missing values are not random.

In small datasets like Wine Quality, median or KNN imputation is often a good starting point, balancing simplicity and reliability.