

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)**

Answer:

1. Mean : Mean is the sum of all numbers divided by the total count.

Total Sum = $82 + 66 + 70 + 59 + 90 + 78 + 76 + 95 + 99 + 84 + 88 + 76 + 82 + 81 + 91 + 64 + 79 + 76 + 85 + 90$

Total Sum = 1621

N = 20

Mean = Total sum / N = $1621 / 20 = 81.05$

Mean = 81.05

2. Median: Median is the middle value in an ordered list.

Sort the data:

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

Since n = 20,

Therefore, median = average of 10th and 11th number

Median = $(81 + 82) / 2 = 81.5$

Median = 81.5

3. Mode: Mode is the number that appears most frequently.

76 appears 3 times that is maximum than other

Mode = 76

4. Interquartile Range (IQR):

Step 1: Sort the data:

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

Divide it into 2 halves

Step 2: First half (1st to 10th):

59, 64, 66, 70, 76, 76, 76, 78, 79, 81

Median of the first half:

5th and 6th values: 76 and 76

$$Q1 = (76 + 76)/2 = 76$$

Step 3: Second half (11th to 20th):

82, 82, 84, 85, 88, 90, 90, 91, 95, 99

Median of the second half:

5th and 6th values: 88 and 90

$$Q3 = (88 + 90)/2 = 89$$

Step 4:

$$IQR = Q3 - Q1 = 89 - 76 = 13$$

Interquartile Range (IQR) = 13

Q.2 1) Machine Learning for Kids

Target Audience:

- Students (typically in middle or high school).
- Teachers and educators introducing artificial intelligence (AI) and machine learning (ML) in classrooms.
- Beginners with little or no coding background.

Use by Target Audience:

- Students upload or input examples of text, numbers, or images and assign labels (e.g., happy/sad, dog/cat).
- The tool uses these examples to train a machine learning model.
- Students can then test the model and see how well it predicts new data.
- They can build fun projects using Scratch or Python that include their ML models — like games that respond to emotional messages or quizzes that react to images.

Benefits:

- Kid-friendly interface using blocks (Scratch) and easy menus.
- Makes ML interactive, fun, and creative for young learners.
- No need to install anything — it runs on the web.
- Encourages logical thinking and digital creativity.

Drawbacks:

- Only covers basic concepts — can't handle complex data science tasks.
- Performance is limited — not suitable for large datasets or real-world deployment.
- Relies on a working internet connection.

Predictive or Descriptive Analytic:**I will choose predictive analytics and reasons for this are:**

- The tool learns from past labeled data and predicts a label for new input (e.g., predicting the emotion of a sentence).
- Example: "I am so excited!" → Model predicts: Happy.
- This prediction of future or unknown input classifies it as predictive, not just summarizing past data.

Supervised / Unsupervised / Reinforcement Learning:**I will chose Supervised Learning because:**

- Users provide labeled training data (e.g., "I love ice cream" → Happy).
- The model learns based on those labels.
- Example: After training, the model is asked to predict the label for a new sentence — and it uses its supervised learning to respond.

2) Teachable Machine**Target Audience:**

- Students, hobbyists, and beginners interested in building AI projects quickly.
- Teachers and presenters who want to demonstrate AI concepts in an interactive way.
- Content creators who want to integrate AI into web or app projects without writing code.

Use by Target Audience:

- Users can train models using images, sounds, or poses by providing multiple labeled examples.
- The tool then creates a machine learning model based on these examples.
- Users can test the model live (e.g., showing their face or speaking) and download/export the model for use in apps, websites, or games.

Benefits:

- No coding needed — just click, record, and train.
- Great for visual learners — everything happens in real-time and is easy to understand.
- Supports exporting to TensorFlow.js, Unity, or other platforms for real projects.
- Works well for quick experiments and demos.

Drawbacks:

- Limited control over how the model is trained (can't tweak algorithms or settings).
- Performance decreases with too many classes or very complex data.
- Requires webcam, mic, and internet access for most features.

Predictive or Descriptive Analytic:**I will choose predictive analytics and reasons for this are:**

- It predicts what a user is doing or saying based on trained data.
- For example: If the model sees you doing a thumbs-up, it predicts "Like".
- The focus is on forecasting/classifying new data, not just summarizing — hence, Predictive Analytic.

Supervised / Unsupervised / Reinforcement Learning:**I will chose Supervised Learning because:**

- The user supplies labeled data (each image or sound is tagged).
- The model learns from these examples to make predictions.
- Example: You label 10 "Hello" gestures and 10 "Stop" gestures — the system uses this to recognize them later. That's 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.

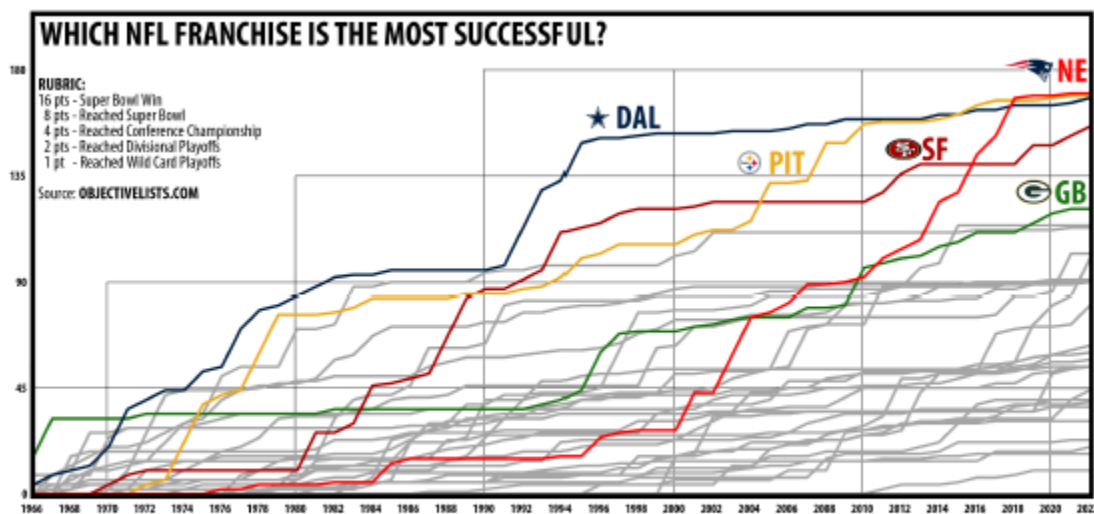


Fig. 8. Contrived metric

Based on the research paper "Misinformative Data Visualizations in the Sports Media Domain" by Drew Scott, I can identify several examples of how data visualizations can mislead readers, specifically in sports media.

One particularly interesting example from the paper is Figure 8, labeled as "Contrived Metric." The author identifies this as a case where a visualization creates a metric without an objective basis, which leads to questionable narratives. This type of misinformation is particularly problematic because it gives the appearance of scientific rigor while actually presenting subjective or arbitrary measurements. The paper categorizes this under "Lie with Statistics" in the Input stage of visualization, and notes that this was one of the most common issues found in their corpus (14 instances).

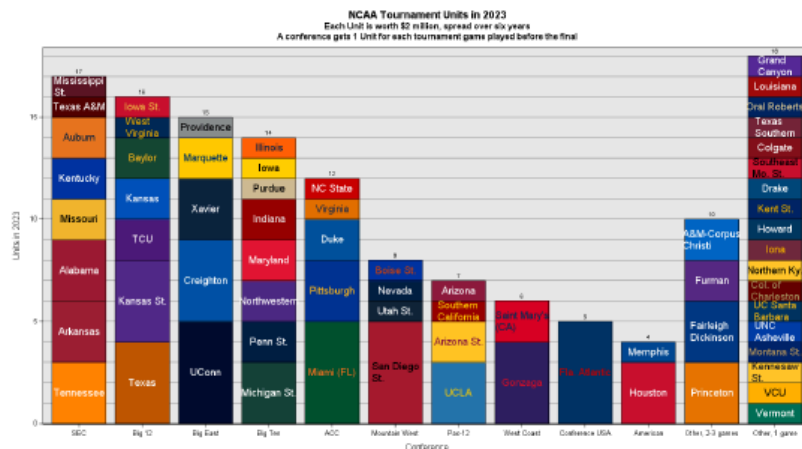


Fig. 3. Stacked bar chart for college basketball conference data

This relates to a current event from March 2023, when ESPN released a "Championship Leverage Index" during the NCAA basketball tournament that purported to show which teams had the most favorable paths to winning. The visualization used a complex formula combining multiple metrics that hadn't been validated, and presented the results as objective analysis. Several sports analysts, including those at The Athletic, criticized the visualization for creating a seemingly scientific metric that actually incorporated significant subjective weighting, leading fans to misunderstand team strengths. The contrived nature of the metric wasn't adequately explained, yet the presentation with precise decimal values and professional graphics gave it an air of authority.

The visualization failed by creating what Scott would categorize as a "Contrived Metric" - a transformation applied to otherwise good data without an apparent or well-explained objective basis. This is particularly misleading because casual viewers typically trust data presented in visual form, especially when it comes from established media outlets. Without proper explanation of methodology or limitations, such visualizations can significantly shape public perception based on questionable analytical foundations.

The paper effectively demonstrates that the sports media domain, despite being considered more "lighthearted" than domains like public health or politics, still exhibits numerous examples of misinformative visualizations that can significantly impact public understanding of the subject matter.

Cite as: Drew Scott. Misinformative Data Visualizations in the Sports Media Domain. *TechRxiv*. April 03, 2024.

DOI: 10.36227/techrxiv.171216651.10279711/v1

Q. 4 Train Classification Model and visualize the prediction performance of trained model required information[Pima Indians Diabetes Database](https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database)

Dataset link: <https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database>

Dataset Description: Pima Indians Diabetes

This dataset contains medical diagnostic information of women of Pima Indian heritage, aged 21 and above. The goal is to predict whether a patient has diabetes based on several health-related measurements.

Feature Descriptions

- **Pregnancies:** Number of times the patient has been pregnant
- **Glucose:** Plasma glucose concentration (mg/dL) after 2 hours in an oral glucose tolerance test
- **BloodPressure:** Diastolic blood pressure (mm Hg)
- **SkinThickness:** Skin fold thickness (mm)
- **Insulin:** 2-Hour serum insulin (mu U/ml)
- **BMI:** Body Mass Index (weight in kg / height in m²)
- **DiabetesPedigreeFunction:** A score showing the likelihood of diabetes based on family history
- **Age:** Age of the patient (in years)
- **Outcome:** Target variable — 0 = No diabetes, 1 = Has diabetes

Step 1: Data loading

```
import pandas as pd
import numpy as np
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.naive_bayes import GaussianNB
from sklearn.svm import SVC
from imblearn.over_sampling import SMOTE
import matplotlib.pyplot as plt
import seaborn as sns
```

```
[2] df = pd.read_csv('/content/diabetes.csv')
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Step 2: Data preprocessing:

```
# Features and label
X = df.drop('Outcome', axis=1)
y = df['Outcome']

# Handle missing values in features (replace zeros with NaN where invalid)
cols_with_zero = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']
X[cols_with_zero] = X[cols_with_zero].replace(0, np.nan)

# Fill NaNs with mean values
X = X.fillna(X.mean())

# Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Step 3: Handle Class Imbalance

```
from imblearn.over_sampling import SMOTE

# Apply SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_scaled, y)
```


We used SMOTE (Synthetic Minority Over-sampling Technique) to balance the dataset. It creates synthetic examples of the minority class by interpolating between existing ones. This helps prevent the model from being biased toward the majority class during training.

Step 4: Train, Validation and Test Split should be 70/20/10, Train and Test split must be randomly done:

```
# First split (10% test)
X_train_val, X_test, y_train_val, y_test = train_test_split(
    X_resampled, y_resampled, test_size=0.10, random_state=42, stratify=y_resampled)

# Second split (20% of remaining for validation)
X_train, X_val, y_train, y_val = train_test_split(
    X_train_val, y_train_val, test_size=2/9, random_state=42, stratify=y_train_val)

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

```
➡ Train: (700, 8), Validation: (200, 8), Test: (100, 8)
```

Step 5: Train SVM with Hyperparameter Tuning

```
param_grid = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf'],
    'gamma': ['scale', 'auto']
}

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

print("Best Parameters:", grid.best_params_)
print("Validation Accuracy:", grid.score(X_val, y_val))
```

```
➡ Best Parameters: {'C': 10, 'gamma': 'auto', 'kernel': 'rbf'}
Validation Accuracy: 0.8
```

In this step, we are training an SVM (Support Vector Machine) model to classify the data. We use GridSearchCV to test different combinations of parameters like C, kernel, and gamma. It finds the best combination that gives the highest accuracy through cross-validation. This helps us choose the most effective settings for the SVM model automatically.

Step 6: Evaluate on Test Data

```
# Best model from GridSearchCV
best_model = grid.best_estimator_
y_pred = best_model.predict(X_test)

# Accuracy
print("Test Accuracy:", accuracy_score(y_test, y_pred))

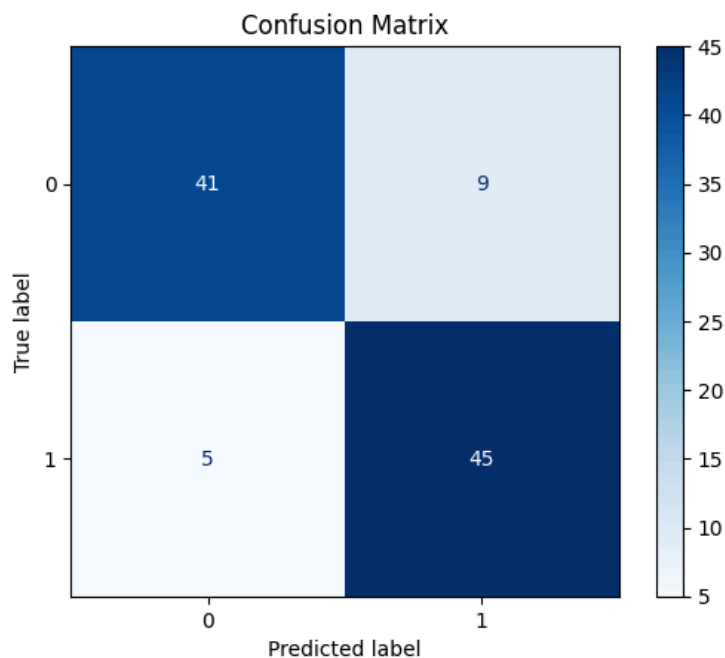
# Classification report
print("\nClassification Report:\n", classification_report(y_test, y_pred))

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

Test Accuracy: 0.86

Classification Report:

	precision	recall	f1-score	support
0	0.89	0.82	0.85	50
1	0.83	0.90	0.87	50
accuracy			0.86	100
macro avg	0.86	0.86	0.86	100
weighted avg	0.86	0.86	0.86	100



Q.5 Train Regression Model and visualize the prediction performance of trained model**Dataset Description:**

This dataset is focused on heart disease prediction, where several features (patient demographics, clinical measures, and medical history) are used to predict various health indicators related to the heart. Let's break down the columns:

1. Unnamed: 0:
 - A non-essential index column (likely used for row identification).
2. Age:
 - The age of the patient (numeric).
 - Age plays a significant role in the risk of heart disease.
3. Sex:
 - Gender of the patient (binary: 0 for female, 1 for male).
4. ChestPain:
 - Type of chest pain experienced by the patient (categorical).
 - Values like "typical", "asymptomatic", "nonanginal", and "nontypical" are used to classify the type of chest pain, which can be a symptom of heart disease.
5. RestBP (Resting Blood Pressure):
 - Resting blood pressure (mm Hg).
 - Blood pressure is a key indicator of heart disease risk.
6. Chol (Cholesterol):
 - Serum cholesterol level in mg/dl.
 - High cholesterol is often associated with heart disease.
7. Fbs (Fasting Blood Sugar):
 - Fasting blood sugar (binary: 0 if < 120 mg/dl, 1 if ≥ 120 mg/dl).
 - High fasting blood sugar may indicate a higher risk of heart disease.
8. RestECG (Resting Electrocardiographic Results):
 - Results of the resting electrocardiogram (categorical).
 - Possible values could indicate normal, ST-T wave abnormality, or left ventricular hypertrophy.
9. MaxHR (Maximum Heart Rate):
 - Maximum heart rate achieved during exercise (numeric).
 - A lower maximum heart rate can indicate poor cardiovascular fitness.
10. ExAng (Exercise Induced Angina):
 - Whether or not exercise induced angina (chest pain) occurred (binary: 0 or 1).
 - Induces a measure of the patient's ability to exercise without chest pain.
11. Oldpeak:
 - Depression of the ST segment in the electrocardiogram during exercise (numeric).
 - It's used to evaluate heart ischemia, which indicates potential coronary artery disease.

12. Slope:

- The slope of the peak exercise ST segment (categorical).
- This is related to the degree of heart disease, showing if the heart rate response is abnormal during exercise.

13. Ca (Number of Major Vessels Colored by Fluoroscopy):

- The number of major blood vessels (0 to 3) that have been colored by fluoroscopy during an angiogram (numeric).
- Used as a measure of coronary artery disease severity.

14. Thal:

- A blood disorder associated with the heart (categorical).
- Values like "fixed", "normal", or "reversible" could indicate various stages of the heart disease or the condition of the coronary arteries.

15. AHD (Heart Disease):

- Whether the patient has heart disease (binary: "Yes" or "No").
- The target variable for classification, indicating if the patient has a heart condition.

Step 1: 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, LinearRegression
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
```

Step 2: Load the dataset

```
[17] import pandas as pd

# Load dataset
df = pd.read_csv("/content/Heart.csv")

# Drop index column if present
df.drop(columns=['Unnamed: 0'], errors='ignore', inplace=True)

# Convert categorical to numerical for correlation analysis
df_encoded = pd.get_dummies(df, drop_first=True)

# Display head
df_encoded.head()
```

	Age	Sex	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	ChestPain_nonanginal	ChestPain_nontypical	ChestPain_typical	Thal_normal	Thal_reversable	AHD_Yes
0	63	1	145	233	1	2	150	0	2.3	3	0.0	False	False	True	False	False	False
1	67	1	160	286	0	2	108	1	1.5	2	3.0	False	False	False	True	False	True
2	67	1	120	229	0	2	129	1	2.6	2	2.0	False	False	False	False	True	True
3	37	1	130	250	0	0	187	0	3.5	3	0.0	True	False	False	True	False	False
4	41	0	130	204	0	2	172	0	1.4	1	0.0	False	True	False	True	False	False

Step 3: Define the Linear Regression Model using OOP

```

# Model definition from previous step
class LinearModelForTarget:
    def __init__(self, target, top_features):
        self.target = target
        self.features = top_features
        self.model = LinearRegression()

    def train_and_evaluate(self, data):
        X = data[self.features]
        y = data[self.target]

        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

        self.model.fit(X_train, y_train)
        y_pred = self.model.predict(X_test)

        r2 = r2_score(y_test, y_pred)
        n, p = X_test.shape
        adj_r2 = 1 - (1 - r2) * (n - 1) / (n - p - 1)
        mae = mean_absolute_error(y_test, y_pred)
        rmse = np.sqrt(mean_squared_error(y_test, y_pred))

        print(f"\nMetrics for {self.target}:")
        print(f"R2 Score: {r2:.4f}")
        print(f"Adjusted R2: {adj_r2:.4f}")
        print(f"MAE: {mae:.4f}")
        print(f"RMSE: {rmse:.4f}")

```

Step 4: Train the model and evaluate the model then plot the predictions:

```

feature_sets = {
    'Oldpeak': ['Slope', 'AHD_Yes', 'MaxHR', 'Thal_normal']
}

import matplotlib.pyplot as plt
import numpy as np

# Step 8: Train and evaluate each target using its best feature set and plot the predictions
for target, features in feature_sets.items():
    print(f"\n🚀 Training model for {target} with features: {features}")
    model = LinearModelForTarget(target, features)
    model.train_and_evaluate(df_encoded)

    # Predictions for plotting
    X = df_encoded[features]
    y = df_encoded[target]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
    y_pred = model.model.predict(X_test)

    # Plot Actual vs Predicted
    plt.figure(figsize=(6, 4))
    plt.scatter(y_test, y_pred, color='blue', alpha=0.6, label='Predicted vs Actual')
    plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], 'r--', lw=2, label='Perfect Prediction')
    plt.xlabel("Actual Values")
    plt.ylabel("Predicted Values")
    plt.title(f"Actual vs Predicted for {target}")
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()

```



Training model for Oldpeak with features: ['Slope', 'AHD_Yes', 'MaxHR', 'Thal_normal']

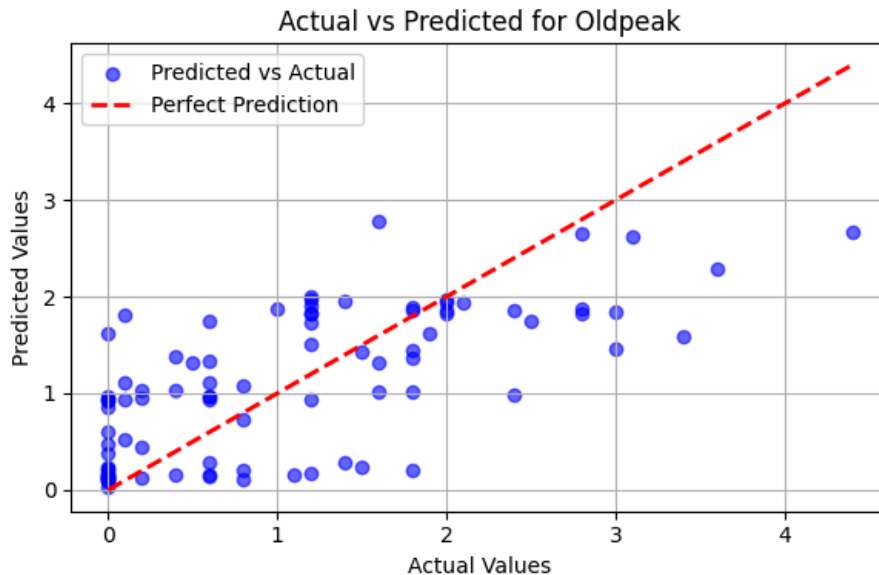
Metrics for Oldpeak:

R2 Score: 0.4746

Adjusted R2: 0.4501

MAE: 0.5949

RMSE: 0.7489



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.

Dataset: <http://kaggle.com/datasets/yasserh/wine-quality-dataset>

Key Features of the Wine Quality Dataset

1. Fixed Acidity
 - Represents non-volatile acids like tartaric acid.
 - Importance: Impacts wine's freshness and stability. Too much or too little affects taste.
2. Volatile Acidity
 - Mainly acetic acid (vinegar-like smell).
 - Importance: High levels lead to unpleasant aroma, reducing quality.
3. Citric Acid
 - Found naturally in wine, adds freshness and flavor.
 - Importance: Enhances flavor, contributes to acidity balance.
4. Residual Sugar
 - Sugar left after fermentation.

- Importance: Affects sweetness; balance is key for quality perception.
- 5. Chlorides
 - Amount of salt in wine.
 - Importance: High levels negatively affect taste; small amounts may enhance flavor.
- 6. Free Sulfur Dioxide
 - Prevents microbial growth and oxidation.
 - Importance: Crucial for wine preservation but must be balanced.
- 7. Total Sulfur Dioxide
 - Includes both free and bound SO_2 .
 - Importance: Excess leads to a pungent smell; impacts shelf life and safety.
- 8. Density
 - Related to sugar and alcohol content.
 - Importance: Indicates fermentation status and body of wine.
- 9. pH
 - Measures acidity or basicity.
 - Importance: Affects stability, color, and taste.
- 10. Sulphates
 - Antimicrobial and antioxidant.
 - Importance: Contributes to SO_2 levels; affects preservation and flavor.
- 11. Alcohol
 - Ethanol content in wine.
 - Importance: Strongly correlated with quality. Higher alcohol often perceived as higher quality.

Handling Missing Data During Feature Engineering

1. Detection:
 - Used `df.isnull().sum()` in Pandas to check missing values.
2. Imputation Techniques Used:
 - Numerical columns (e.g., alcohol, pH):
 - Used mean or median imputation depending on skewness.
 - For example:
`df['alcohol'].fillna(df['alcohol'].mean(), inplace=True)`

1. Mean Imputation

Advantages:

- Simple and fast to implement.
 - Maintains dataset size.
- Works well with symmetric, normally distributed data.

Disadvantages:

- Affected by outliers.
- Reduces data variability.
- Can introduce bias in skewed distributions.

2. Median Imputation

Advantages:

- Robust to outliers.
- Better suited for skewed data.
- Preserves central tendency better than mean for non-normal data.

Disadvantages:

- Still doesn't account for correlation between features.
- Less effective for normally distributed data.

3. Mode Imputation (for categorical or discrete data)

Advantages:

- Simple to apply.
- Maintains most frequent category.

Disadvantages:

- Not suitable for continuous data.
- Can cause overrepresentation of the mode.

4. Dropping Missing Rows

Advantages:

- Very simple to apply.
- Avoids introducing potential bias from imputation.

Disadvantages:

- Leads to data loss.
- Not suitable if many rows have missing values.
- May reduce model performance due to smaller training set.