Name: Dev Gaonkar **Div: D15C** **Roll no: 12**

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)

Ans.

1. Find the mean:
To find the mean we use the following formula:

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

=1611/20
=**80.55**

2. Find the median:
To find median, first we will sort the data in ascending order
Sorted data: 59, 64, 66, 70, 76, 76, 76, 78, 79, 81, 82, 82, 84, 85, 88, 90, 90, 91, 95, 99
Since n here is even we will find the average of the 2 numbers at middle position (10th and 11th position)

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

3. Find the mode:
mode is the value that appears most frequently in a dataset.
In the given dataset 76 appears 3 times,which is the most frequent and so mode is **76**.

4. Find the interquartile range:
Q1 = median of first 10 = (76 + 76) / 2 = 76
Q3 = median of last 10 = (88 + 90) / 2 = 89
IQR = Q3 - Q1 = 89 - 76 = **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

Ans.

1. Machine Learning for Kids

Target Audience:

Machine Learning for Kids is primarily designed for school students (aged 8 and above) and educators who want to introduce machine learning concepts in a simple and interactive way.

Use by Target Audience:

- Students use it to build machine learning models using intuitive interfaces like text, image, and number classification.
- Educators use it to design AI-based projects in classrooms and integrate machine learning with block-based coding environments like Scratch and App Inventor.

Benefits:

- User-Friendly: Simplifies complex machine learning concepts.
- Educational Integration: Easily connects with curriculum-based tools.
- Hands-On Learning: Promotes experiential learning through project-based tasks.
- Cloud-Based: No installation required, works on web browsers.

Drawbacks:

- Limited in Complexity: Not suitable for advanced learners or complex ML tasks.
- Internet Dependency: Requires an active internet connection.
- Limited Dataset Size: Works with small-scale datasets only.

Analytic Type:

-Predictive Analytic

Reason: It trains models to predict outcomes based on new inputs (e.g., classifying images or texts), which is the essence of predictive analytics.

Learning Type:

-Supervised Learning

Reason: The tool uses labeled data (input-output pairs) to train models, which clearly aligns with supervised learning principles.

2. Teachable Machine

Target Audience:

Teachable Machine by Google is targeted at a broad audience, including:

- Students
- Educators
- Artists
- Developers
- Anyone with no coding experience who wants to explore machine learning through visual tools.

Use by Target Audience:

- Students and educators use it to experiment with ML through simple image, sound, or pose recognition.
- Artists and creators use it to prototype interactive experiences.
- Developers export trained models to integrate into real applications using TensorFlow.js or TensorFlow Lite.

Benefits:

- No Coding Required: Extremely beginner-friendly.
- Versatile Inputs: Supports image, sound, and pose models.
- Fast Training: Uses browser-based training for instant results.
- Exportable Models: Easily downloadable for real-world applications.

Drawbacks:

- Limited Control: Advanced ML customization is not possible.
- Data Privacy: User-generated data may be temporarily uploaded to servers.
- Performance Limitation: Less effective for large datasets or complex model training.

Analytic Type:

-Predictive Analytic

Reason: Like ML for Kids, Teachable Machine is used to build models that make predictions based on trained data (e.g., classifying whether an image is a cat or a dog).

Learning Type:

-Supervised Learning

Reason: The tool requires labeled datasets (e.g., user provides examples of images labeled as "Class A", "Class B", etc.), indicating the use of 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.

Ans.

Case Study: Misleading Electricity Price Chart in Spain

In a government presentation, a chart showing Spain's electricity prices from 2004 to 2013 was widely criticized for its misleading structure. The chart initially used yearly data from 2004 to 2012, but suddenly switched to quarterly data in 2012–2013—without clearly indicating the change. This design choice made it seem like electricity prices were stabilizing, even though they were not.

How the Data Visualization Method Failed:

- Inconsistent Time Intervals: The sudden shift from yearly to quarterly data created an illusion of more frequent and smaller changes, deceiving the viewer into thinking prices were improving.

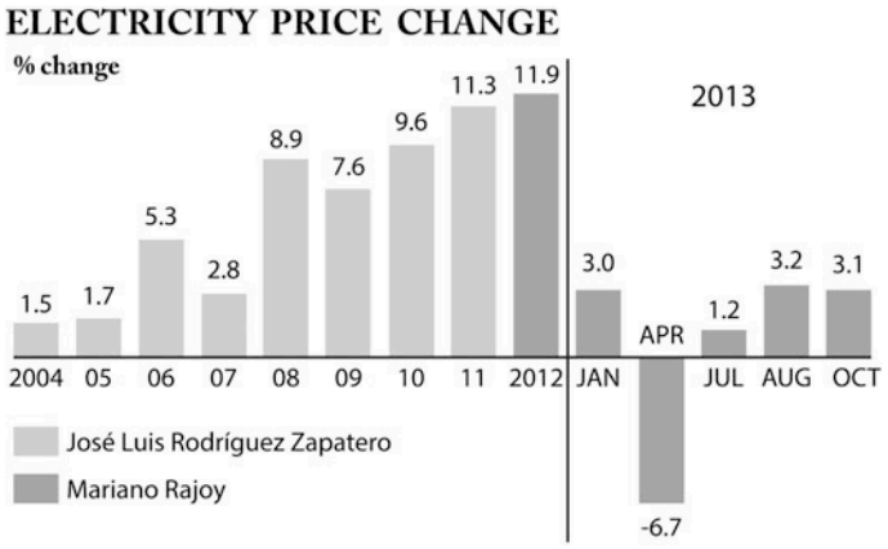
- Misleading Bar Heights: Bars representing just 3 months were nearly the same height as bars representing full years. This visually distorted the magnitude of the data.

- Lack of Transparency: No labels or annotations clarified the change in data intervals, causing further confusion and misleading interpretation.

Why This Is Problematic:

Data visualizations are meant to simplify and clarify information, but when time scales or measurements are altered without explanation, it can manipulate perception. Viewers may draw inaccurate conclusions, especially in politically or economically sensitive topics.

Visual Aid:



Misleading Electricity Price Chart link

Conclusion:

This example demonstrates the importance of consistent scales, accurate labeling, and clear communication in data visualizations. Even when data is accurate, poor visual representation can result in unintentional or intentional misinformation, affecting public opinion and policy decisions.

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

Ans.

Step 1. Importing Required Libraries

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_auc_score, roc_curve
from imblearn.over_sampling import SMOTE
import seaborn as sns
import matplotlib.pyplot as plt
```

Step 2. Loading the Dataset

- Loaded using pandas.read_csv().
- The last column is used as the target label.
- Features (X) and target (y) are separated.

```
df = pd.read_csv("diabetes.csv")
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
```

Step 3. Data Preprocessing

- No null values → no imputation required.
- No categorical variables → no encoding needed.
- Feature Scaling is essential for SVM → used 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 4. Splitting the Dataset

- Used train_test_split() in two steps:
    1. First split into 70% Train and 30% Temp (Validation + Test)
    2. Then split Temp into 20% Validation and 10% Test
- Stratification used to maintain class distribution.

```
# Split Data: 70% train, 20% validation, 10% test
X_temp, X_test, y_temp, y_test = train_test_split(X, y, test_size=0.10, random_state=42, stratify=y)
X_train, X_val, y_train, y_val = train_test_split(X_temp, y_temp, test_size=2/9, random_state=42, stratify=y_temp)
```

Step 5. Handling Class Imbalance

- Applied SMOTE on the training set.
- Balances the classes by generating synthetic samples for the minority class.

```
smote = SMOTE(random_state=42)
X_train_bal, y_train_bal = smote.fit_resample(X_train_scaled, y_train)
```

Step 6. Feature Scaling

- Applied StandardScaler:
    - Fit on training data.
    - Transformed train, validation, and test data using the same scaler to avoid data leakage.

Step 7. Model Training with Hyperparameter Tuning

- Used Support Vector Classifier (SVC) with GridSearchCV for tuning:
    - Parameters like 'C', 'kernel', and 'gamma' were tuned.
    - Cross-validation used to select the best combination.

```
svm = SVC(probability=True)
param_grid = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf'],
    'gamma': ['scale', 'auto']
}
grid = GridSearchCV(svm, param_grid, cv=5)
grid.fit(X_train_bal, y_train_bal)
best_model = grid.best_estimator_
```

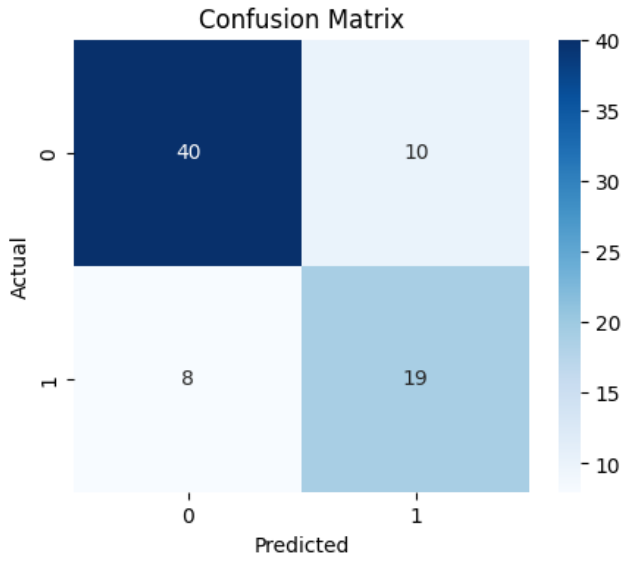
Step 8. Evaluation and Metrics

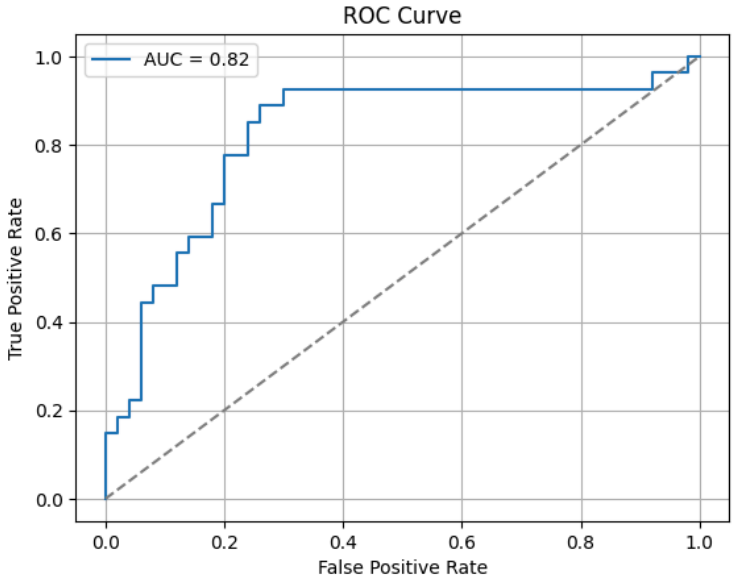
```
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.80      0.82        50
           1       0.66      0.70      0.68        27

    accuracy                           0.77        77
   macro avg       0.74      0.75      0.75        77
weighted avg       0.77      0.77      0.77        77

Accuracy: 0.7662337662337663
```





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

Model Evaluation Summary (SVM Classifier)

- Test Accuracy: 76.6%
    - The model performs consistently on unseen data, indicating decent generalization ability.

Class-wise Performance (from Classification Report)

Class 0 (Non-Diabetic)

- Precision: 83%
  → When the model predicts "Non-Diabetic", it's correct 83% of the time.

- Recall: 80%
  → The model correctly identifies 80% of actual "Non-Diabetic" cases.

- F1-Score: 82%
  → Balance between precision and recall is strong.

Class 1 (Diabetic)

- Precision: 66%
  → Around two-thirds of the predictions for "Diabetic" are correct.

- Recall: 70%
  → The model is able to detect 70% of actual diabetic patients.

- F1-Score: 68%
  → Performance is moderately good, but there's room for improvement in catching diabetic cases more accurately.

Conclusion

- The SVM model achieves good accuracy and performs fairly well in distinguishing between diabetic and non-diabetic individuals.
- It detects the majority of diabetic cases (**70% recall**) while keeping the false positives at a reasonable level.
- Slight improvements in recall and precision for the minority class (diabetic) could further enhance the model's clinical usefulness.

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

Ans.

1. Required Libraries

```
import pandas as pd, numpy as np

from sklearn.linear_model import Ridge

from sklearn.preprocessing import PolynomialFeatures, StandardScaler

from sklearn.pipeline import Pipeline

from sklearn.model_selection import train_test_split, GridSearchCV

from sklearn.metrics import r2_score, mean_squared_error

import matplotlib.pyplot as plt

import seaborn as sns
```

We make use of:

- Pipeline to organize the sequence of preprocessing steps: polynomial feature generation, feature scaling, and applying ridge regression.

- GridSearchCV to find the best hyperparameters via cross-validation.

- r2_score and mean_squared_error to evaluate model accuracy.

2. Defining a Class for Regression

```
class RegressionModel:

    def __init__(self, model_pipeline, param_grid):

        ...
```

This class neatly encapsulates model fitting, evaluation, and hyperparameter optimization.
It is modular and flexible, making it applicable to other regression use-cases with minimal changes.

3. Importing the Dataset

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

df = pd.read_excel(url)
```

The dataset consists of several real estate-related features, such as:

- Distance to the nearest MRT station
- Count of nearby convenience stores
- Age of the building
- Latitude and longitude coordinates

4. Cleaning and Splitting the Data

```
df = df.drop(columns=['No'])  # Removing the serial number column

X = df.drop(columns=['Y house price of unit area'])  # Feature matrix

y = df['Y house price of unit area']  # Target vector
```

- X: All input features (excluding the house price)
- y: The target variable representing house price per unit area

5. Splitting the Dataset for Training and Testing

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

- Data is divided using a 70/30 ratio
- random_state=42 ensures consistent splits during re-runs

6. Constructing the Pipeline and Setting Hyperparameters

```
pipeline = Pipeline([

    ('poly', PolynomialFeatures()),
```

```
    ('scaler', StandardScaler()),
    ('ridge', Ridge())
])
```

- PolynomialFeatures: Adds non-linear transformations to the input
- StandardScaler: Ensures features have zero mean and unit variance (crucial for ridge regression)
- Ridge: Applies L2 regularization to control model complexity

Hyperparameter grid for tuning:

```
param_grid = {
    'poly__degree': [2, 3, 4],
    'ridge__alpha': [0.1, 1, 10, 100]
}
```

- We test polynomial degrees from 2 to 4
- Ridge penalty (alpha) values from 0.1 to 100 are evaluated

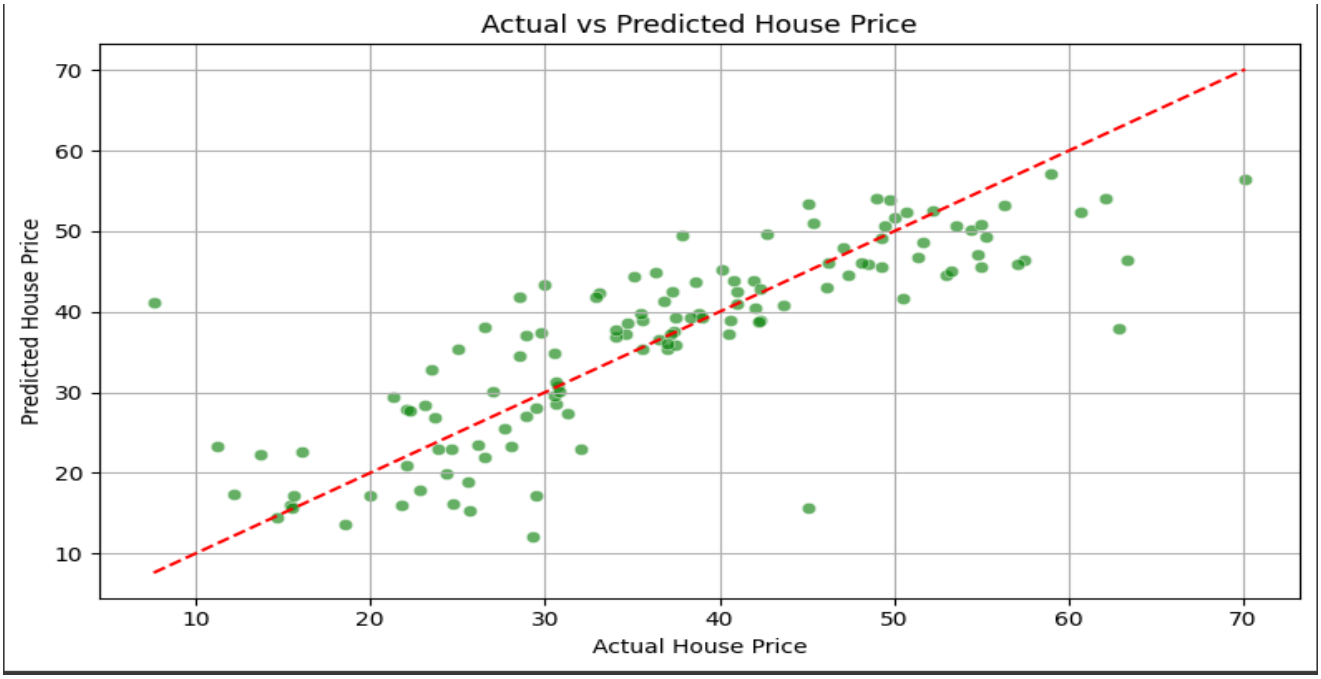
7. Model Training and Testing

```
best_model = reg_model.train(X_train, y_train)
y_test_actual, y_pred = reg_model.evaluate(best_model, X_test, y_test)
```

- The model is trained using 5-fold cross-validation during hyperparameter search
- The best configuration found is used to generate predictions
- Performance metrics:
    - R^2: How well the model explains variation in the target
    - Adjusted R^2: Corrects R^2 for number of predictors used
    - MSE: The average squared error between predicted and actual values

8. Visualizing Predictions

```
sns.scatterplot(x=y_test_actual, y=y_pred)
plt.plot([y_test_actual.min(), y_test_actual.max()], [y_test_actual.min(), y_test_actual.max()], 'r--')
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs Predicted House Prices")
plt.show()
```



- Scatterplot compares actual and predicted prices
- A red dashed line shows where predictions would lie if they were perfect

Final Model Results

Best Parameters: {'poly__degree': 2, 'ridge__alpha': 1}

R^2 Score: 0.6552

Adjusted R^2 Score: 0.6376

Mean Squared Error: 57.6670

- The model successfully explains about 66% of the variation in house prices
- The adjusted R^2 accounts for model complexity and offers a more realistic performance measure

Why a Perfect R^2 (>0.99) May Be Unrealistic

- Real datasets often contain noisy or incomplete data
- Not all influential factors may be captured (e.g., neighborhood popularity, future infrastructure plans)
- While polynomial features can increase expressiveness, excessive complexity can cause overfitting
- Ridge regression helps reduce this overfitting but cannot recover information not present in the dataset

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.

Key Features of the Wine Quality Dataset:

The dataset includes several physicochemical attributes of wine that help predict wine quality (on a scale of 0–10). Common features include:

Feature	Description	Importance
fixed acidity	Tartaric and other non-volatile acids	Affects stability and taste
volatile acidity	Acetic acid content	High levels produce an unpleasant vinegar taste
citric acid	Citric acid content	Adds freshness and flavor
residual sugar	Sugar left after fermentation	Influences sweetness; excess can indicate incomplete fermentation
chlorides	Salt content	Affects salinity and taste
free sulfur dioxide	Free SO_2 preventing microbial growth	Impacts shelf life and preservation
total sulfur dioxide	Total SO_2 (free + bound)	Excess may lead to off-odors; affects health and taste
density	Mass-to-volume ratio	Related to sugar content and alcohol
pH	Acidity/alkalinity	Influences microbial stability and flavor
sulphates	Antimicrobial additive	Associated with wine preservation and bitterness
alcohol	Alcohol content	Major determinant of taste, body, and aroma
quality	(Target) Wine quality rating (0–10)	Label used for regression or classification

Importance of Each Feature in Predicting Wine Quality:

1. Alcohol: Strong positive correlation — higher alcohol often leads to better quality.

2. Volatile Acidity: Strong negative correlation — high levels reduce perceived quality.

3. Sulphates: Slight positive correlation — relates to preservation and flavor.
4. Citric Acid & pH: Contribute to the crispness and freshness of wine.
5. Residual Sugar: Important for sweetness, but high amounts in dry wines might be a flaw.
6. Density: Indirectly relates to alcohol and sugar — helpful but not directly impactful alone.

Insight: A combination of features — especially alcohol, acidity, and sulphates — better predicts quality than any single feature.

Handling Missing Data During Feature Engineering:

In many Kaggle versions of this dataset, no missing values are present.
However, if missing data existed, these are common strategies used:

1. Dropping Rows (Simple Removal):

- When to use: Small number of missing rows (<5%)
- Advantage: Easy, no distortion of data
- Disadvantage: Loss of potentially useful information

2. Mean/Median/Mode Imputation:

- When to use: Numerical data with random missingness
- Advantage: Simple and fast
- Disadvantage: Ignores feature relationships, can reduce variance

3. K-Nearest Neighbors (KNN) Imputation:

- Uses: Similar instances to fill missing values
- Advantage: Preserves multivariate relationships
- Disadvantage: Computationally expensive for large datasets

4. Regression Imputation:

- Idea: Predict missing values using regression from other features
- Advantage: More informed than mean/median
- Disadvantage: Risk of overfitting and propagation of error

5. Iterative Imputation (e.g., MICE):

- Advanced Technique: Iteratively models each feature with missing values as a function of other features
- Advantage: Most statistically robust
- Disadvantage: Slower, more complex

Conclusion:

- Key features like alcohol, volatile acidity, and sulphates are vital in predicting wine quality.
- Handling missing data carefully during feature engineering is critical to maintain model performance.
- Choose imputation techniques based on the nature of data, amount of missingness, and model complexity.