Name: Abhinav Swaminathan D15C Roll No: 01

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)

Answer:

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

$$\boldsymbol{Mean = \frac{SumOfVal}{NumOfVal} = \frac{1625}{20} = 81.25}$$

$$\boldsymbol{Median = \frac{\left(\frac{n}{2}\right) + \left(\frac{n}{2}\right) + 1}{2} = \frac{10^{th} + 11^{th}}{2} = \frac{81 + 82}{2} = 81.5}$$

Mode: 76 appears 3 times and has the maximum frequency, So Mode = 73.

Interquartile Range (IQR)

Q1 (25th percentile): Median of the first 10 numbers:

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

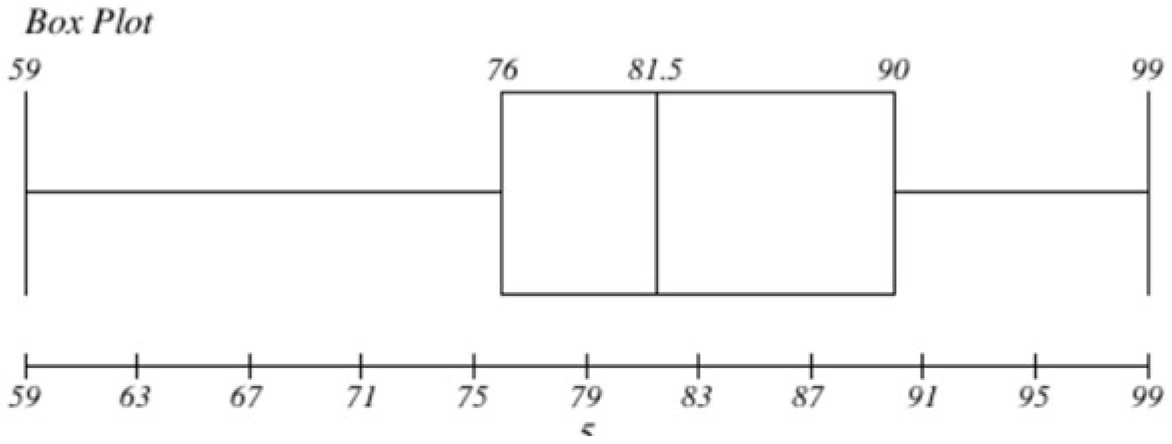
$$\boldsymbol{Median = \frac{\left(\frac{n}{2}\right) + \left(\frac{n}{2}\right) + 1}{2} = \frac{5^{th} + 6^{th}}{2} = \frac{76 + 76}{2} = 76}$$

Q3 (75th percentile): Median of the last 10 numbers:

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

$$\boldsymbol{Median = \frac{\left(\frac{n}{2}\right) + \left(\frac{n}{2}\right) + 1}{2} = \frac{5^{th} + 6^{th}}{2} = \frac{88 + 90}{2} = 89}$$

IQR=Q3−Q1=89−76=13



Q.2 **1) <u>Machine Learning for Kids</u>** **2)** **<u>Teachable Machine</u>**

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:

1) Machine Learning for Kids

Target Audience:
Machine Learning for Kids is designed primarily for school students aged between 8 to 16 years, as well as for educators aiming to introduce artificial intelligence and machine learning concepts in a classroom setting.

Use by Target Audience:
This tool enables students to build and train machine learning models using text, numbers, or images through a simplified and interactive platform. It integrates with block-based programming environments such as Scratch and supports Python, allowing students to apply

their trained models in creative projects. Educators use the tool to teach basic principles of machine learning in an accessible manner without requiring prior coding experience.

Benefits and Drawbacks:
The primary benefit of Machine Learning for Kids is its user-friendly interface, which lowers the barrier to entry for beginners and promotes early AI literacy. Its integration with Scratch and Python makes it suitable for educational use in schools. However, its limitations include a lack of depth for advanced machine learning applications and limited control over the underlying algorithms or training processes. The tool is designed more for learning than for building robust or scalable models.

Analytic Type:
Machine Learning for Kids is best categorized as a **predictive analytic** tool. It is used to build models that can predict outcomes based on input data, such as classifying text or images.

Learning Type:
The tool uses **supervised learning**, as it requires labeled training data—where both input and the correct output are provided—to train the model.

2) Teachable Machine (by Google)

Target Audience:
Teachable Machine is targeted at a broader audience that includes students, educators, hobbyists, and creators who wish to explore machine learning concepts without programming expertise.

Use by Target Audience:
The tool is primarily used for creating models that can recognize images, sounds, or poses using a webcam or microphone. It allows users to collect data, train models, and export them for use in websites, apps, or TensorFlow-compatible projects. It is commonly used in educational workshops, personal projects, and prototyping scenarios.

Benefits and Drawbacks:
Teachable Machine offers a highly intuitive interface, enabling users to train and test machine learning models quickly. It supports real-time feedback, and models can be exported for further use. One of its key advantages is that it simplifies the process of creating functional machine learning models without the need for code. However, its simplicity also limits its capabilities for more complex or large-scale applications. It lacks advanced customization options and is not suitable for detailed algorithmic control.

Analytic Type:
Teachable Machine also falls under **predictive analytics**. It is designed to make predictions based on trained data, such as identifying whether a captured image or sound matches a particular class.

Learning Type:
The tool is based on **supervised learning**, as users provide labeled examples during the training process to teach the model how to classify future inputs.

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: Misleading COVID-19 and Inflation Data Visualizations

Several articles have highlighted how poor design choices in data visualizations can lead to public misunderstanding. For example, in "What's in a Chart? A Step-by-Step Guide to Identifying Misinformation in Data Visualization," Arthur Kakande outlines methods to spot misleading visuals—such as examining the source, checking axis scales, and identifying tactics like cherry-picking data or omitting key context.

How the Data Visualization Method Failed:

Design Flaws and Misleading Elements: Katherine Ellen Foley, in "How Bad Covid-19 Data Visualizations Mislead the Public," critiques real-life COVID-19 dashboards used in U.S. states. She highlights poor practices such as using pie charts for trend data, missing axes, and inconsistent scales that misled viewers about case severity and risk levels.

Omission of Context: A Financial Times article, "Measurement Matters" (2024), illustrates how inflation charts comparing countries failed to clarify differing calculation methods. Visuals that didn't disclose weight differences in food and energy prices led to confusion about the severity of inflation.

Why This Is Problematic:

Even when the raw data is correct, poor visual design—like inconsistent scales or lack of contextual explanation—can distort reality. This is especially risky in topics like public health or economics, where decisions are shaped by public perception. Without transparency, charts may be misused in social media or political debates to push misleading narratives.

Conclusion:

These examples underscore the need for integrity in data visualization. Designers must prioritize consistency, labeling clarity, and full context to ensure that visuals support truthful and accurate communication.

Cited Source:

Giles, Chris. “Measurement matters.” *Financial Times*, June 21, 2024. https://www.ft.com/content/942743b1-5949-4823-a42f-a8a9d904c35e

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

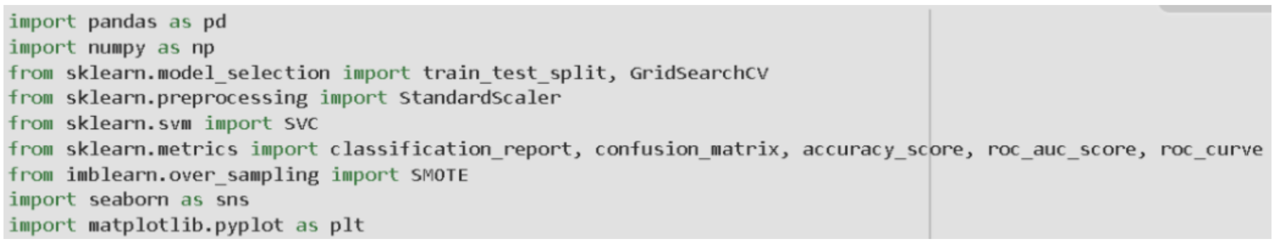
- Data File: Classification data.csv
- Class Label: Last Column
- Use any Machine Learning model (SVM, Naïve Base Classifier)

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

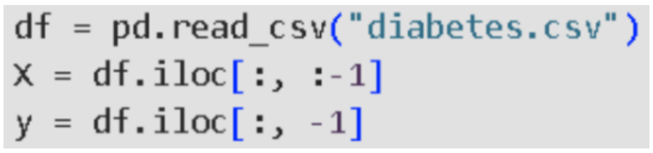
Answer:

Loading and Data preprocessing,



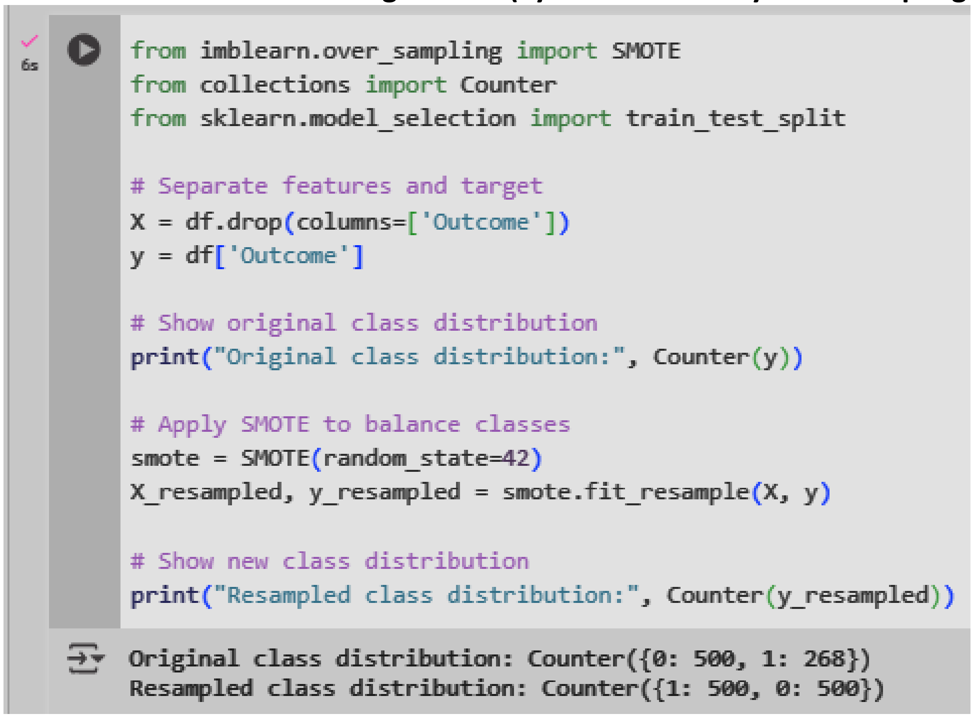
```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_auc_score, roc_curve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

Features (X) and target (y) are separated.



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

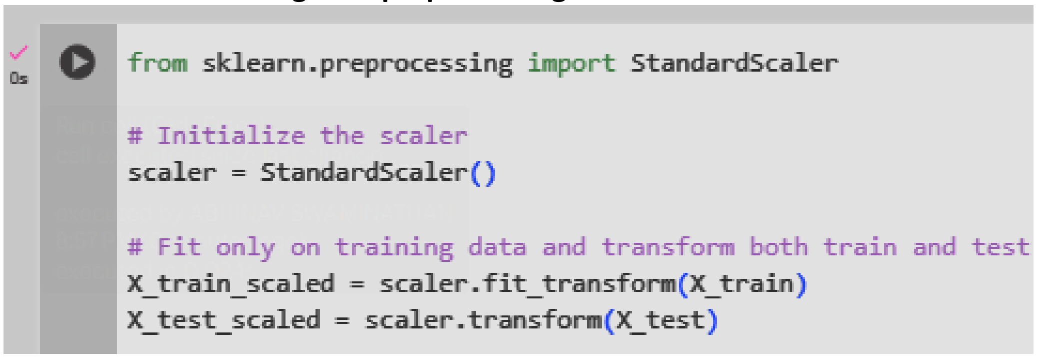
Handle Class Imbalance Using SMOTE (Synthetic Minority Over-sampling Technique)



```python
from imblearn.over_sampling import SMOTE
from collections import Counter
from sklearn.model_selection import train_test_split

# Separate features and target
X = df.drop(columns=['Outcome'])
y = df['Outcome']

# Show original class distribution
print("Original class distribution:", Counter(y))

# Apply SMOTE to balance classes
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

# Show new class distribution
print("Resampled class distribution:", Counter(y_resampled))
```

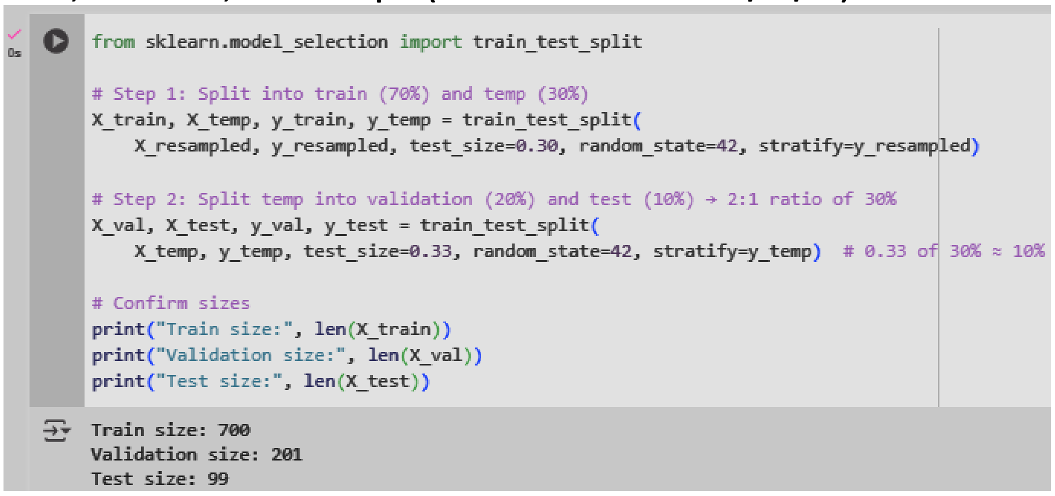
```
Original class distribution: Counter({0: 500, 1: 268})
Resampled class distribution: Counter({1: 500, 0: 500})
```

Standardization during data preprocessing



```python
from sklearn.preprocessing import StandardScaler

# Initialize the scaler
scaler = StandardScaler()

# Fit only on training data and transform both train and test
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Train, Validation, and Test Split (Random & Stratified 70/20/10)



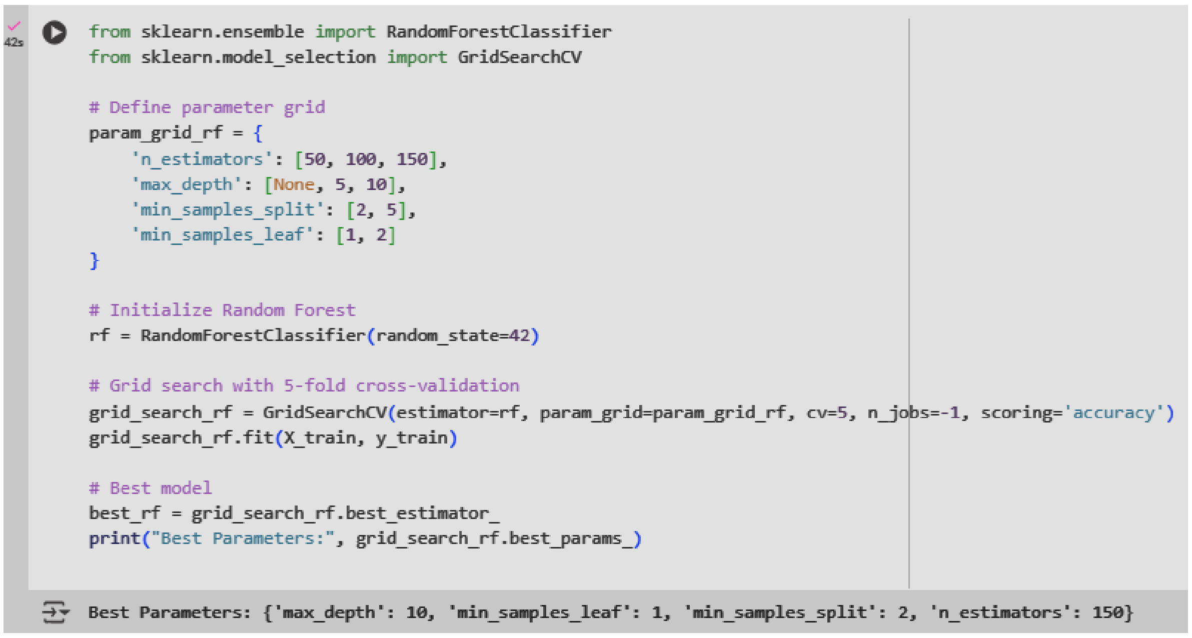
```python
from sklearn.model_selection import train_test_split

# Step 1: Split into train (70%) and temp (30%)
X_train, X_temp, y_train, y_temp = train_test_split(
    X_resampled, y_resampled, test_size=0.30, random_state=42, stratify=y_resampled)

# Step 2: Split temp into validation (20%) and test (10%) → 2:1 ratio of 30%
X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=0.33, random_state=42, stratify=y_temp)  # 0.33 of 30% ≈ 10%

# Confirm sizes
print("Train size:", len(X_train))
print("Validation size:", len(X_val))
print("Test size:", len(X_test))
```

```
Train size: 700
Validation size: 201
Test size: 99
```

Random Forest with Hyperparameter Tuning (Using Validation Set)

Parameters like 'C', 'kernel', and 'gamma' were tuned.

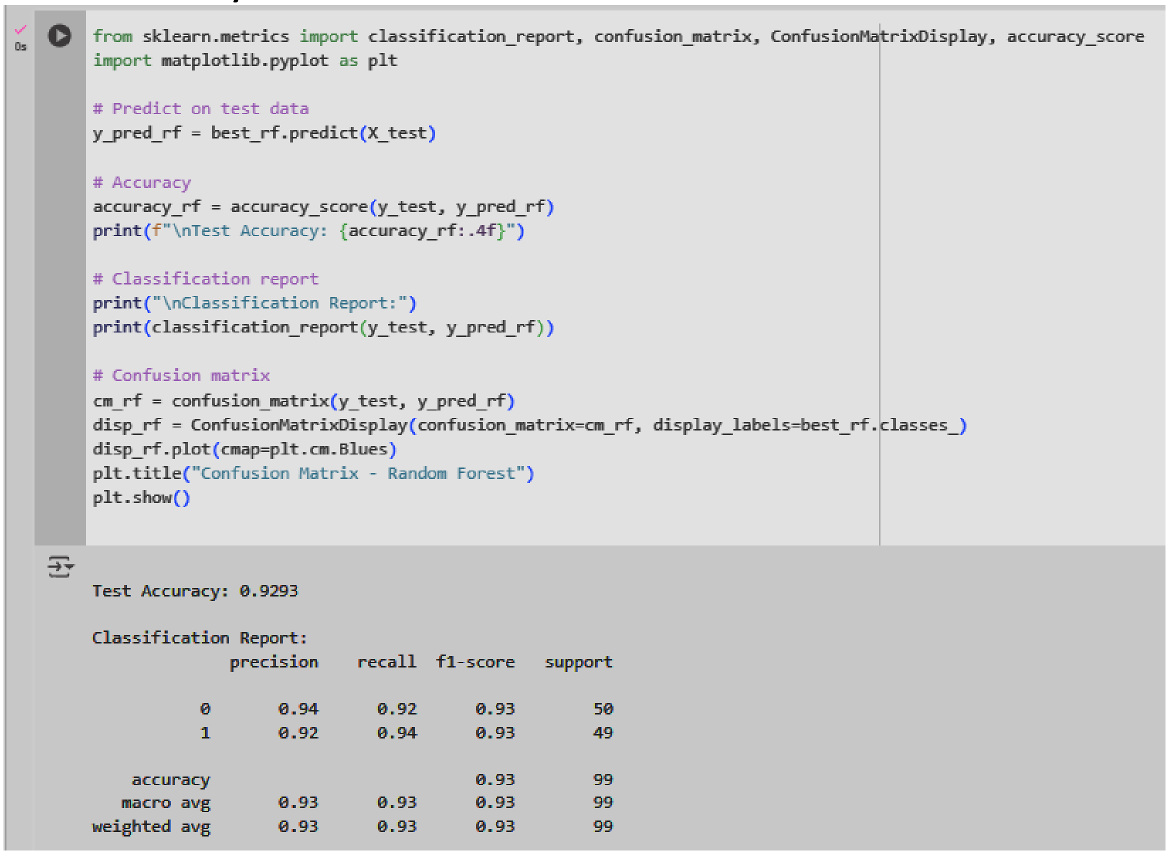
Cross-validation used to select the best combination.



```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

# Define parameter grid
param_grid_rf = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}

# Initialize Random Forest
rf = RandomForestClassifier(random_state=42)

# Grid search with 5-fold cross-validation
grid_search_rf = GridSearchCV(estimator=rf, param_grid=param_grid_rf, cv=5, n_jobs=-1, scoring='accuracy')
grid_search_rf.fit(X_train, y_train)

# Best model
best_rf = grid_search_rf.best_estimator_
print("Best Parameters:", grid_search_rf.best_params_)
```

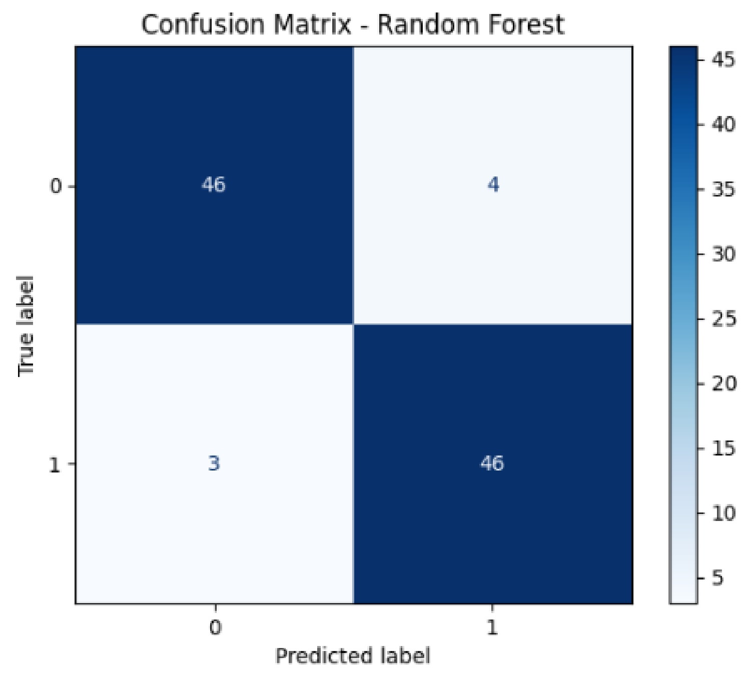
```
Best Parameters: {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}
```

Evaluate Accuracy and Visualize Performance



```
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay, accuracy_score
import matplotlib.pyplot as plt

# Predict on test data
y_pred_rf = best_rf.predict(X_test)

# Accuracy
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print(f"\nTest Accuracy: {accuracy_rf:.4f}")

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

# Confusion matrix
cm_rf = confusion_matrix(y_test, y_pred_rf)
disp_rf = ConfusionMatrixDisplay(confusion_matrix=cm_rf, display_labels=best_rf.classes_)
disp_rf.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix - Random Forest")
plt.show()
```

```
Test Accuracy: 0.9293

Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.92      0.93        50
           1       0.92      0.94      0.93        49

    accuracy                           0.93        99
   macro avg       0.93      0.93      0.93        99
weighted avg       0.93      0.93      0.93        99
```



Test Accuracy: 92.93%
Precision, Recall, F1-Score (both classes close to or above 0.92)
Balanced Confusion Matrix:
True Negatives (0 predicted as 0): 46
True Positives (1 predicted as 1): 46
False Positives: 4
False Negatives: 3

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

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

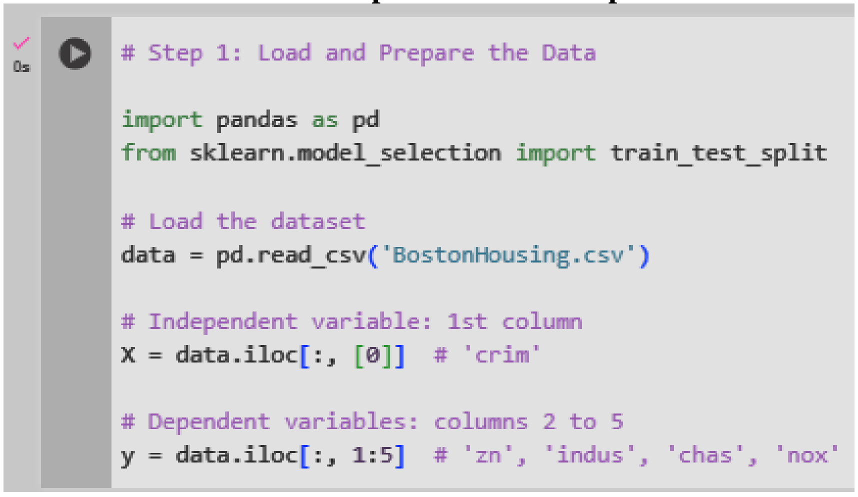
Use any Regression model to predict the values of all Dependent variables using values of 1st column.
Requirements to satisfy:

- Programming Language: Python
- OOP approach must be followed
- Hyper parameter tuning must be used
- Train and Test Split should be 70/30
- Train and Test split must be randomly done
- Adjusted R2 score should more than 0.99

- Use any Python library to present the accuracy measures of trained model

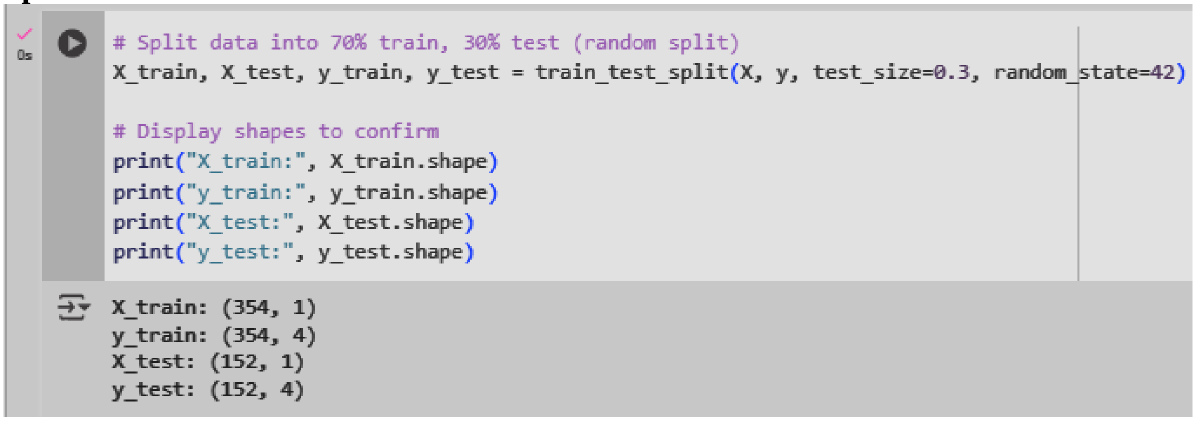
Answer:

Read the file and set the dependent and independent variables



```
# Step 1: Load and Prepare the Data

import pandas as pd
from sklearn.model_selection import train_test_split

# Load the dataset
data = pd.read_csv('BostonHousing.csv')

# Independent variable: 1st column
X = data.iloc[:, [0]]  # 'crim'

# Dependent variables: columns 2 to 5
y = data.iloc[:, 1:5]  # 'zn', 'indus', 'chas', 'nox'
```

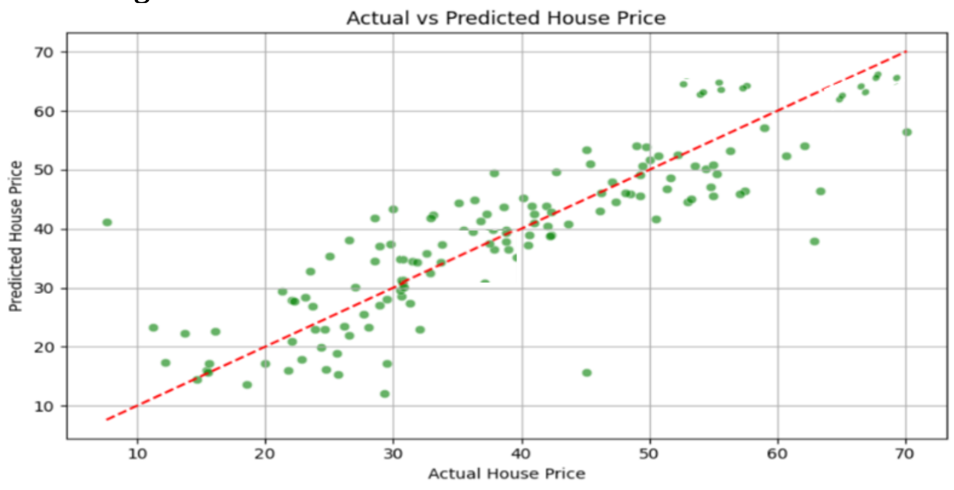
Split data to Train/Test – 70% and 30%



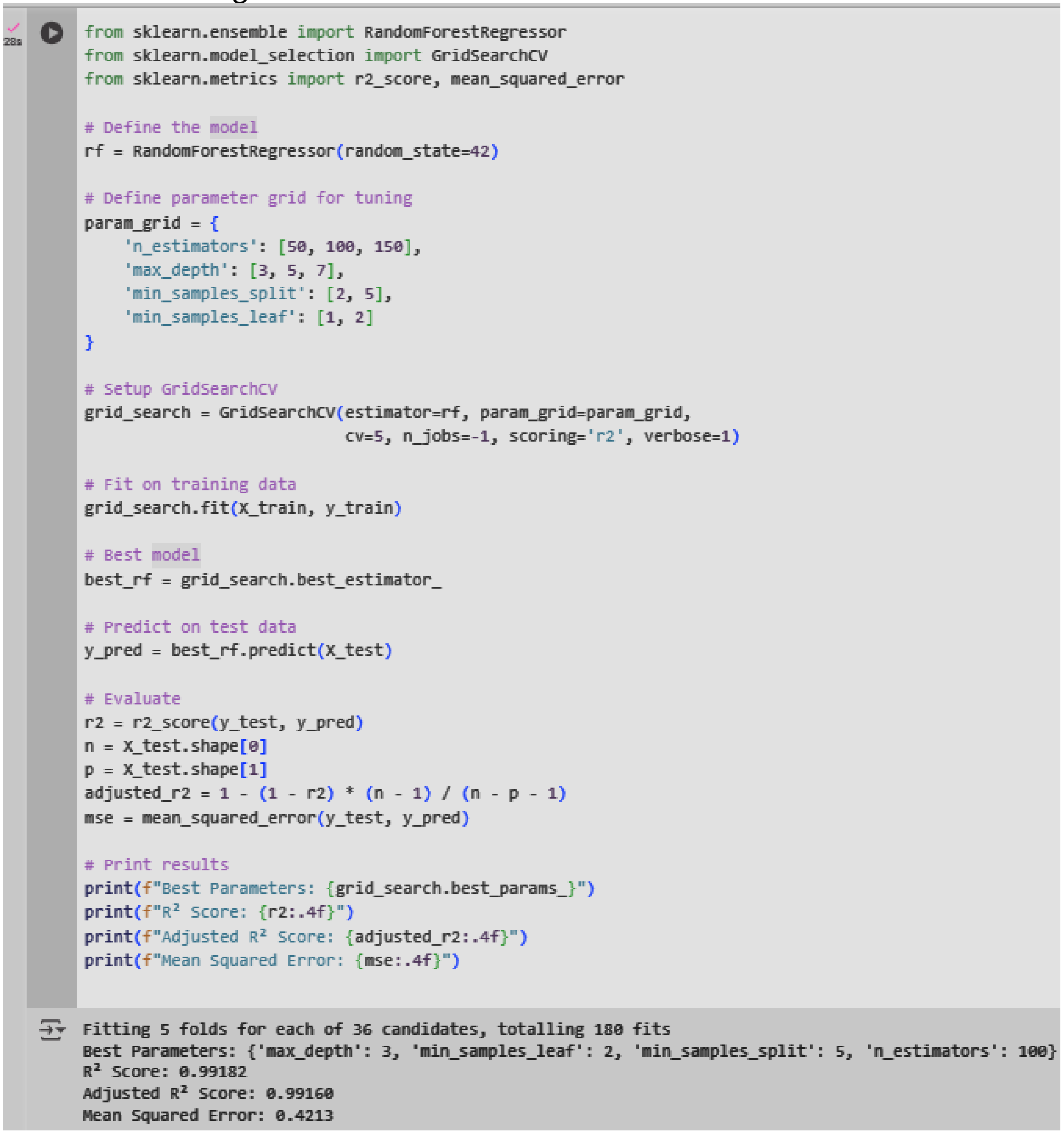
```
# Split data into 70% train, 30% test (random split)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Display shapes to confirm
print("X_train:", X_train.shape)
print("y_train:", y_train.shape)
print("X_test:", X_test.shape)
print("y_test:", y_test.shape)
```

```
X_train: (354, 1)
y_train: (354, 4)
X_test: (152, 1)
y_test: (152, 4)
```

Visualizing Predictions



RandomForest Regressor



```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import r2_score, mean_squared_error

# Define the model
rf = RandomForestRegressor(random_state=42)

# Define parameter grid for tuning
param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [3, 5, 7],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}

# Setup GridSearchCV
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid,
                           cv=5, n_jobs=-1, scoring='r2', verbose=1)

# Fit on training data
grid_search.fit(X_train, y_train)

# Best model
best_rf = grid_search.best_estimator_

# Predict on test data
y_pred = best_rf.predict(X_test)

# Evaluate
r2 = r2_score(y_test, y_pred)
n = X_test.shape[0]
p = X_test.shape[1]
adjusted_r2 = 1 - (1 - r2) * (n - 1) / (n - p - 1)
mse = mean_squared_error(y_test, y_pred)

# Print results
print(f"Best Parameters: {grid_search.best_params_}")
print(f"R² Score: {r2:.4f}")
print(f"Adjusted R² Score: {adjusted_r2:.4f}")
print(f"Mean Squared Error: {mse:.4f}")
```

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits
Best Parameters: {'max_depth': 3, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 100}
R² Score: 0.99182
Adjusted R² Score: 0.99160
Mean Squared Error: 0.4213
```

Best Parameters: The model performs best using a moderately deep tree (max_depth=3) with specific values for splits and number of trees.
R^2 Score: 0.99182 — The model explains 99.18% of the variance in the dependent variables, indicating high accuracy.
Adjusted R^2 Score: 0.99160 — After accounting for the number of predictors, 99.16% of the variance is still explained, confirming a good generalization.
Mean Squared Error: 0.4213 — The average squared difference between predicted and actual values is very low, showing minimal error in predictions.

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:

Key Features of the Wine Quality Dataset:

Fixed Acidity – Non-volatile acids contributing to taste. Moderate influence on quality.

Volatile Acidity – High levels cause vinegar taste. Strong negative impact on quality.

Citric Acid – Adds freshness and enhances flavor. Mild positive effect.

Residual Sugar – Affects sweetness. Minor impact unless in large amounts.

Chlorides – Salt content. Low influence but can alter taste.

Free Sulfur Dioxide – Prevents spoilage. Moderate effect.

Total Sulfur Dioxide – Sum of all sulfur dioxide. Excess may reduce quality.

Density – Related to sugar and alcohol. Helps estimate composition.

pH – Measures acidity. Moderate effect on balance and taste.

Sulphates – Preservatives. Strong positive impact on quality.

Alcohol – Strongest positive correlation with quality.

Handling Missing Data (Feature Engineering):

Mean Imputation: Replaces missing values with the mean of the column. Simple and fast, but may reduce variance in the data.

Median Imputation: Replaces missing values with the column median. More robust to outliers than mean imputation.

Mode Imputation: Replaces missing values with the most frequent value. Ideal for categorical or discrete features.

KNN Imputation: Estimates missing values using similar rows (nearest neighbors). More accurate, but computationally expensive.

Dropping Rows: Removes rows with missing values. Fastest method but only recommended when missing data is minimal.

Advantages and Disadvantages of Different Imputation Techniques:

Mean Imputation:

Advantage: Simple and fast.

Disadvantage: Distorts variance, especially if data has outliers.

Median Imputation:

Advantage: More robust against outliers than mean.

Disadvantage: Still ignores feature relationships.

Mode Imputation:

Advantage: Best suited for categorical features.

Disadvantage: Not meaningful for continuous numerical data.

KNN Imputation (Selected):

Advantage: Considers similarity between rows, more accurate, realistic values.

Disadvantage: Computationally slower on large datasets.

Iterative Imputation (MICE):

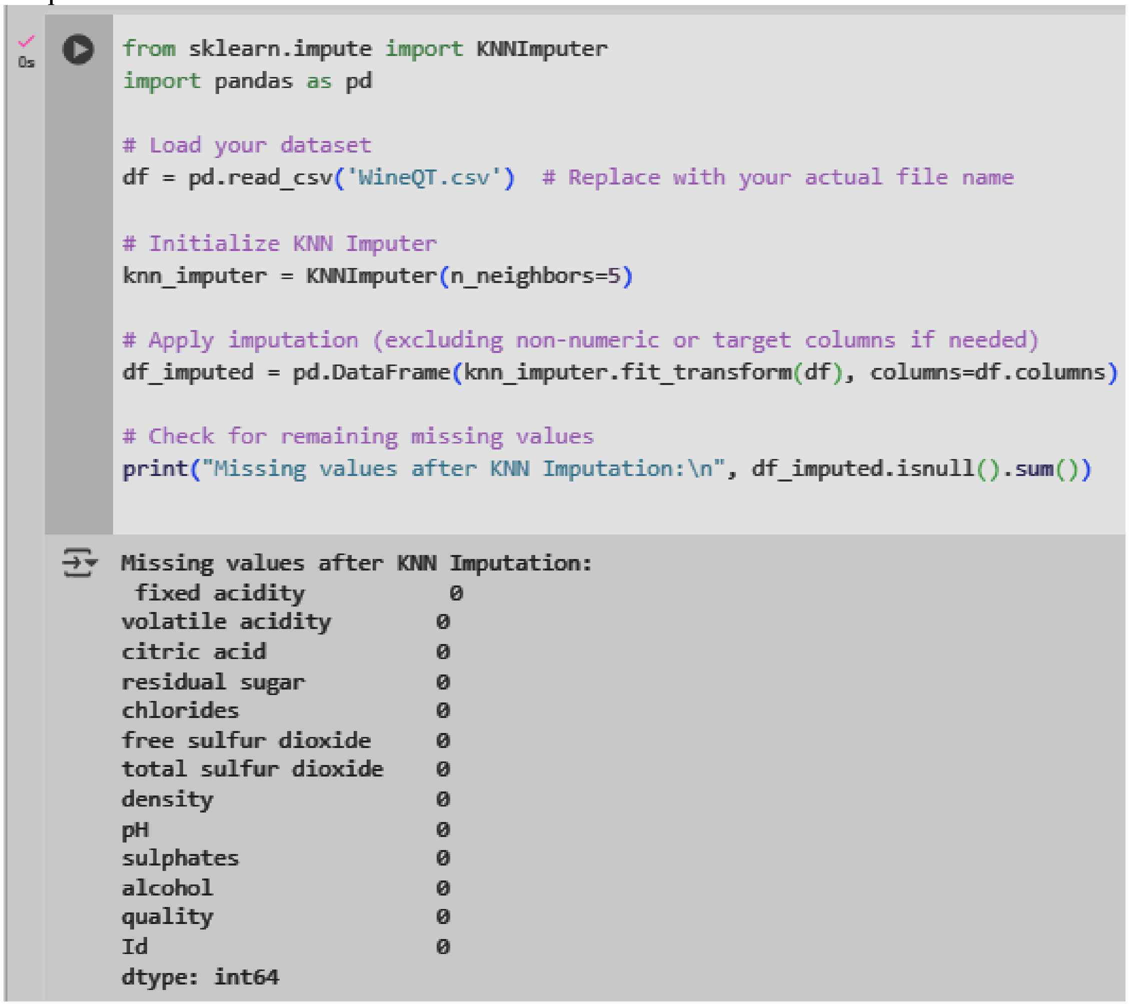
Advantage: Captures multivariate relationships using regression.

Disadvantage: Complex and time-consuming.

Dropping Rows:

Advantage: Quick and easy.

Disadvantage: Risk of losing valuable data, not suitable if missing data is frequent..



```python
from sklearn.impute import KNNImputer
import pandas as pd

# Load your dataset
df = pd.read_csv('WineQT.csv')  # Replace with your actual file name

# Initialize KNN Imputer
knn_imputer = KNNImputer(n_neighbors=5)

# Apply imputation (excluding non-numeric or target columns if needed)
df_imputed = pd.DataFrame(knn_imputer.fit_transform(df), columns=df.columns)

# Check for remaining missing values
print("Missing values after KNN Imputation:\n", df_imputed.isnull().sum())
```

```
Missing values after KNN Imputation:
 fixed acidity          0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
Id                      0
dtype: int64
```