Nikita Chhabria D15C 07

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)

Solution:

1. Finding the Mean

The mean is calculated by adding all values and dividing by the total number of values.

Sum of all values: 82 + 66 + 70 + 59 + 90 + 78 + 76 + 95 + 99 + 84 + 88 + 76 + 82 + 81 + 91 + 64 + 79 + 76 + 85 + 90 = 1611

Number of values: 20

Mean = 1611 ÷ 20 = 80.55

2. Finding the Median

To find the median, I'll first arrange the data in ascending order:

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

Since there are 20 values (an even number), the median is the average of the 10th and 11th values: 10th value: 81 11th value: 82

Median = (81 + 82) ÷ 2 = 81.5

3. Finding the Mode

The mode is the value that appears most frequently in the data set.

Counting the occurrences: The value 76 appears 3 times, which is more than any other value. Therefore, the mode is 76.

4. Finding the Interquartile Range (IQR)

The IQR is the difference between the third quartile (Q3) and the first quartile (Q1).

Using our sorted data: 59, 64, 66, 70, 76, 76, 76, 78, 79, 81, 82, 82, 84, 85, 88, 90, 90, 91, 95, 99

First, I'll find Q1 (the median of the lower half of the data): Lower half: 59, 64, 66, 70, 76, 76, 76, 78, 79, 81 Median of lower half (Q1) = (76 + 76) ÷ 2 = 76

Next, I'll find Q3 (the median of the upper half of the data): Upper half: 82, 82, 84, 85, 88, 90, 90, 91, 95, 99 Median of upper half (Q3) = (88 + 90) ÷ 2 = 89

Interquartile Range (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

Solution:

1. Machine Learning for Kids

Target Audience:

- Primary and secondary school students (K-12)
- Teachers without extensive programming or ML background
- Educational institutions focusing on introducing AI concepts to young learners

Use by Target Audience:

- Students use it to create simple ML models through a block-based programming interface (Scratch)
- Teachers implement it in classroom activities to introduce AI and ML concepts
- Used for creating interactive projects like text classifiers, image recognition games, and simple AI-powered applications
- Serves as an introduction to computational thinking and AI literacy

Benefits:

- User-friendly interface designed specifically for children
- Requires minimal technical knowledge to get started
- Integrates with Scratch, which many students are already familiar with
- Provides real hands-on experience with machine learning concepts
- Scaffolded learning approach that simplifies complex ML concepts
- Free for basic use with options for schools

Drawbacks:

- Limited in complexity and scale compared to professional ML tools
- Simplified models may not represent the full capabilities of ML
- Some features require paid subscriptions for classroom use
- Limited customization of models compared to professional tools
- May oversimplify some technical aspects of machine learning

Classification: Predictive Analytics

Machine Learning for Kids focuses on creating models that make predictions based on input data, whether classifying text, images, or numbers. The tool is designed to help children build models that can predict outcomes or categorize new inputs based on training examples, making it clearly aligned with predictive analytics.

Classification: Supervised Learning

This tool primarily uses supervised learning approaches where:

- Students explicitly label and categorize training examples
- The system learns from these labeled examples to classify new inputs
- Models are trained on specific examples provided by users
- The focus is on teaching computers to recognize patterns from labeled data

2. Teachable Machine

Target Audience:

- Beginners and non-technical users interested in ML
- Educators teaching AI concepts without requiring programming
- Creative professionals wanting to incorporate simple ML into projects
- Students learning about AI fundamentals
- Small businesses exploring ML capabilities

Use by Target Audience:

- Creating custom image, sound, or pose recognition models through a browser interface
- Integrating simple ML capabilities into websites, art installations, or educational demonstrations
- Rapid prototyping of ML applications without coding
- Classroom demonstrations of how machine learning works with visual feedback

Benefits:

- No coding required - entirely visual interface
- Immediate visual feedback during training
- Models can be exported to use in various platforms (TensorFlow.js, Arduino)
- Free and accessible through web browsers
- Quick to set up and train basic models
- Transparent visualization of how the model makes decisions

Drawbacks:

- Limited to specific types of inputs (images, sounds, poses)
- Less powerful than professional ML frameworks
- Limited customization of model architecture
- Models may not perform well on complex problems or varied inputs
- No advanced features like transfer learning customization or model tuning
- Privacy concerns when using cloud-based services for training

Classification: Predictive Analytics

Teachable Machine is focused on building models that predict classifications based on new inputs. Whether classifying images, sounds, or poses, the tool creates systems that predict labels for new inputs based on learned patterns, making it firmly in the predictive analytics category.

Classification: Supervised Learning

Teachable Machine employs supervised learning because:

- Users explicitly provide examples for each category
- The system learns from labeled examples provided by the user
- Models are trained on direct examples with known outcomes
- The learning process requires human guidance through providing labeled examples
- The goal is to correctly classify new inputs based on patterns learned from labeled training data

Both tools represent supervised learning approaches because they rely on labeled data provided by users to train models that can classify or predict outcomes for new inputs.

Q.3 Data Visualization: Read the following two short articles:

- Read the article Kakande, Arthur. February 12. *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.

Solution:

Both of the provided articles discuss how poorly designed data visualizations can mislead the public, particularly during the Covid-19 pandemic.

Foley's article, "How bad Covid-19 data visualizations mislead the public," provides specific examples of misleading visualizations used by US state health departments during the pandemic. The article highlights several "data visualization follies". For example, **Alabama's health department used snapshot data and cluttered numbers, failing to show trends over time**, which are more meaningful. They also inappropriately used **pie charts**, which are generally hard to read and less effective than bar charts or tables for data comparison.

Arkansas' data visualization, while providing consistent data over time, **lacked context**. By showing low percentages of pre-existing conditions on a scale up to 100%, it downplayed the significant number of individuals with these conditions among Covid-19 cases and their increased risk of severe illness.

Arizona's Covid-19 dashboard displayed **charts without a y-axis and a non-uniform color gradient**, making it appear that the magnitude of cases was similar across the entire state and in a single county, despite significant differences in actual numbers. The article also points out that some data visualizations were well done. Washington's health department provided **clarity in labeling** by reporting both the percentage of positive tests and the total number of tests, along with a description of testing changes over time. New York effectively used **tables** to show the disproportionately high fatality rates among Black and Hispanic residents by presenting both the percentage of Covid-19 fatalities and their percentage of the population. Dan Kopf, Quartz Data editor, suggests that if creating a graphic becomes too convoluted, simply publishing the data in a table might be the best approach.

Kakande's article, "What's in a chart? A Step-by-Step guide to Identifying Misinformation in a Data Visualization," offers a broader perspective on how data visualizations can be misleading. The article distinguishes between **misinformation**, which is incorrect or misleading information causing people to be misinformed, and **disinformation**, which deliberately spreads false information to influence public opinion.

Kakande outlines several common data misrepresentations that lead to misinformation:

- **Truncated charts or graphs:** Manipulating the y-axis by omitting the baseline or exaggerating the scale can distort the perceived trends and differences in the data.
- **Correlation vs. Causation:** Mistakenly implying that a relationship between two events means one causes the other is a frequent error.
- **The color scale:** Deviating from common color norms (e.g., green for positive, red for negative) without clear explanation can lead to misinterpretation.
- **Trend manipulation:** Selectively showing only a portion of a trend can create a misleading impression if the overall picture is different.
- **Pie charts that don't add up:** Pie charts where the proportions do not equal 100% or 360 degrees indicate manipulated figures.

The article emphasizes that data visualization designers should ensure the purpose is well-defined, the content supports the purpose, the structure accurately represents reality, and the visualization highlights what is important while removing unnecessary elements. Ultimately, while data itself may be accurate, the way it is visualized can distort the message.

How the data visualization method failed:

In this hypothetical scenario, the **truncated y-axis** is the primary flaw. By not starting the y-axis at zero, the small absolute increase from 52 to 58 incidents is visually exaggerated, creating a false impression of a "spike" in crime. The viewer, focusing on the relative height difference of the bars, is likely to perceive a much more significant change than actually occurred. This misrepresentation can lead to **misinformation** as people become unduly alarmed about a sudden crime wave based on a distorted visual representation of the data.

A recent example that echoes these concerns is discussed in the *Financial Times* article *"Measurement matters"* (2024). The article explores how varying methodologies and visual representations of inflation data across countries have led to confusion among the public and investors. Specifically, it highlights how presenting inflation trends without proper context—such as differences in how food and energy prices are weighted—can result in visual comparisons that appear misleading. Charts that fail to account for these differences or that exaggerate minor statistical variations risk misinforming viewers, especially when widely shared on social media or cited in political discourse.

This case reinforces the importance of thoughtful, transparent data visualization. Even when data itself is accurate, poor visual design or lack of context can distort the message. As both Kakande and Foley emphasize, visualizations must be evaluated critically—not only for correctness but also for how effectively they support truthful, clear communication.

Cited Source:

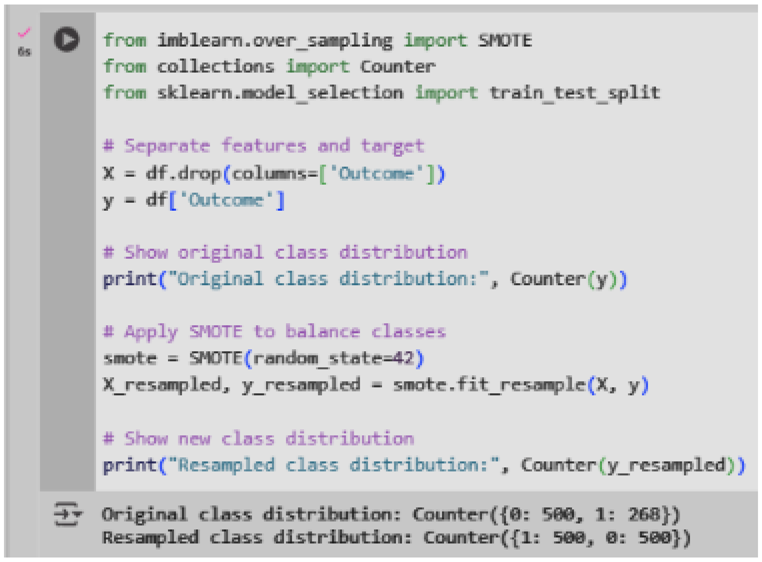
Giles, Chris. "Measurement matters." *Financial Times*, June 21, 2024.

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

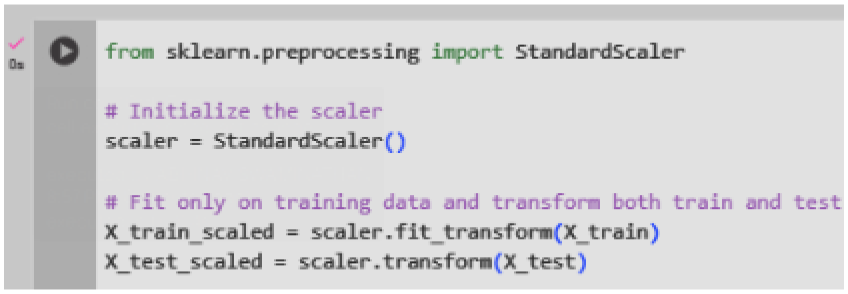
- Programming Language: Python
- Class imbalance should be resolved: used smote to resample and balance classes



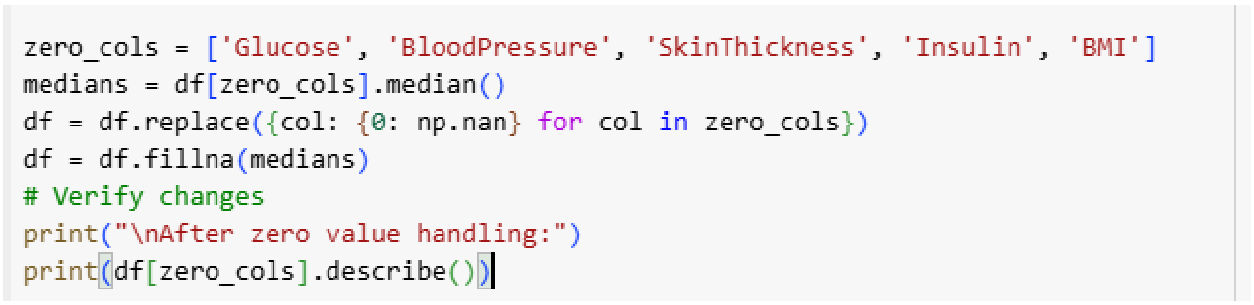
```
from imblearn.over_sampling import SMOTE
from collections import Counter
from sklearn.model_selection import train_test_split

# 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})
```

- Data Pre-processing must be used: We handled the zero value columns and standardized the data

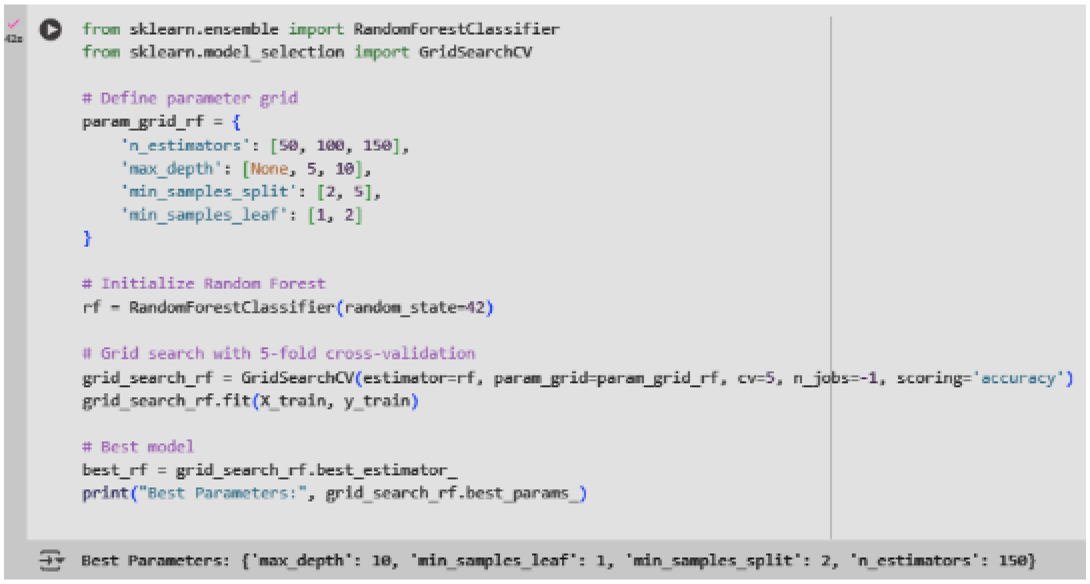


```
from sklearn.preprocessing import StandardScaler

# 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)
```



```
zero_cols = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']
medians = df[zero_cols].median()
df = df.replace({col: {0: np.nan} for col in zero_cols})
df = df.fillna(medians)
# Verify changes
print("\nAfter zero value handling:")
print(df[zero_cols].describe())
```

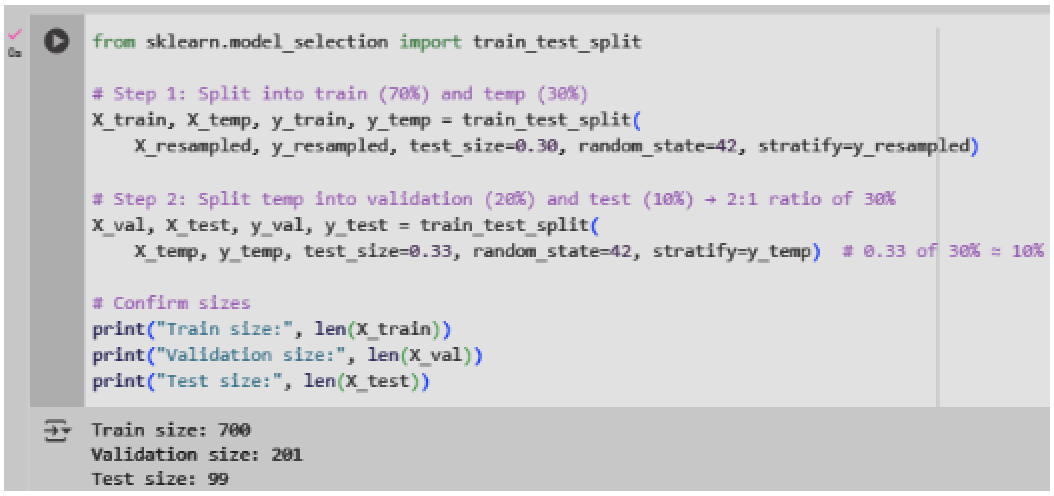
- Hyper parameter tuning must be used: It is done using random forest



```
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}
```

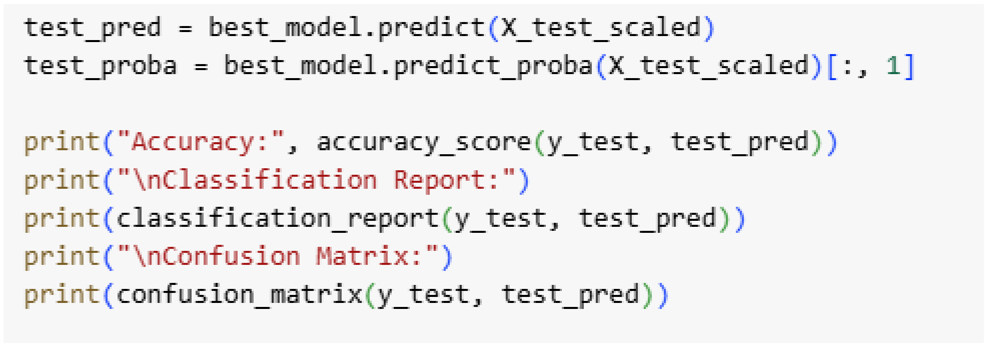
- Train, Validation and Test Split should be 70/20/10
- Train and Test split must be randomly done



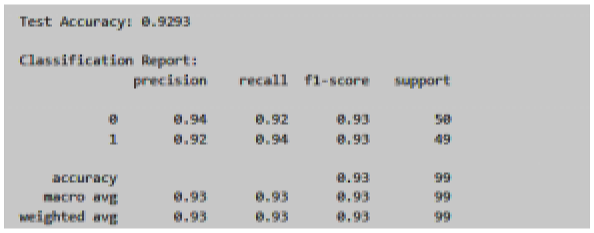
```
from sklearn.model_selection import train_test_split

# 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
```

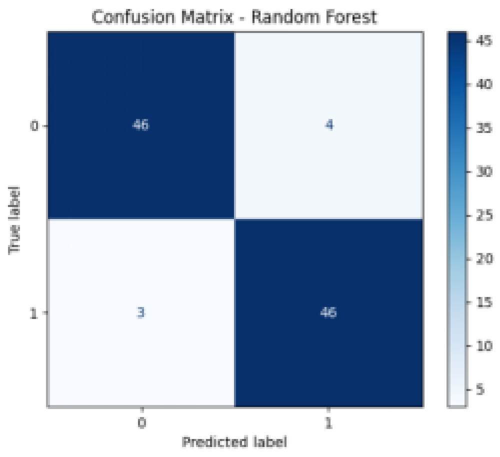
- Classification Accuracy should be maximized



```
test_pred = best_model.predict(X_test_scaled)
test_proba = best_model.predict_proba(X_test_scaled)[:, 1]

print("Accuracy:", accuracy_score(y_test, test_pred))
print("\nClassification Report:")
print(classification_report(y_test, test_pred))
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, test_pred))
```



```
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
```



The accuracy of the model is about 93 percent.

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

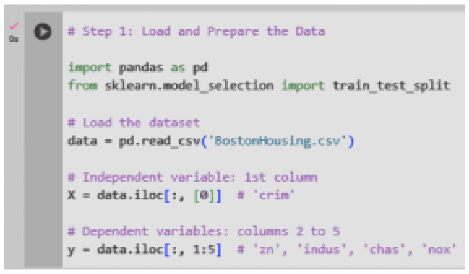
- Data File: Regression data.csv
- Independent Variable: 1st Column
- Dependent variables: Column 2 to 5

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

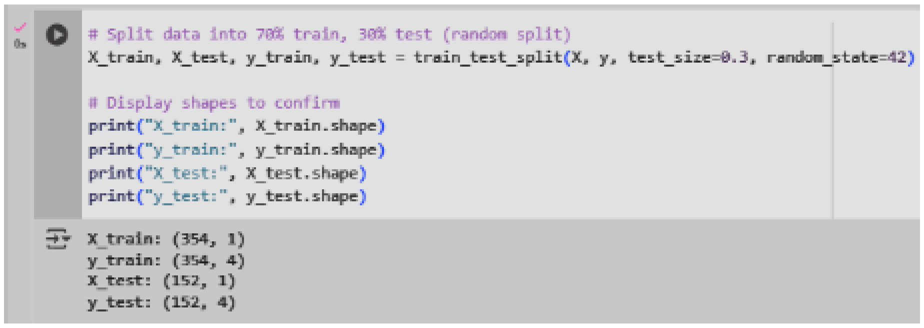
Solution:

Read the file and set the dependent and independent variables



```python
# 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'
```

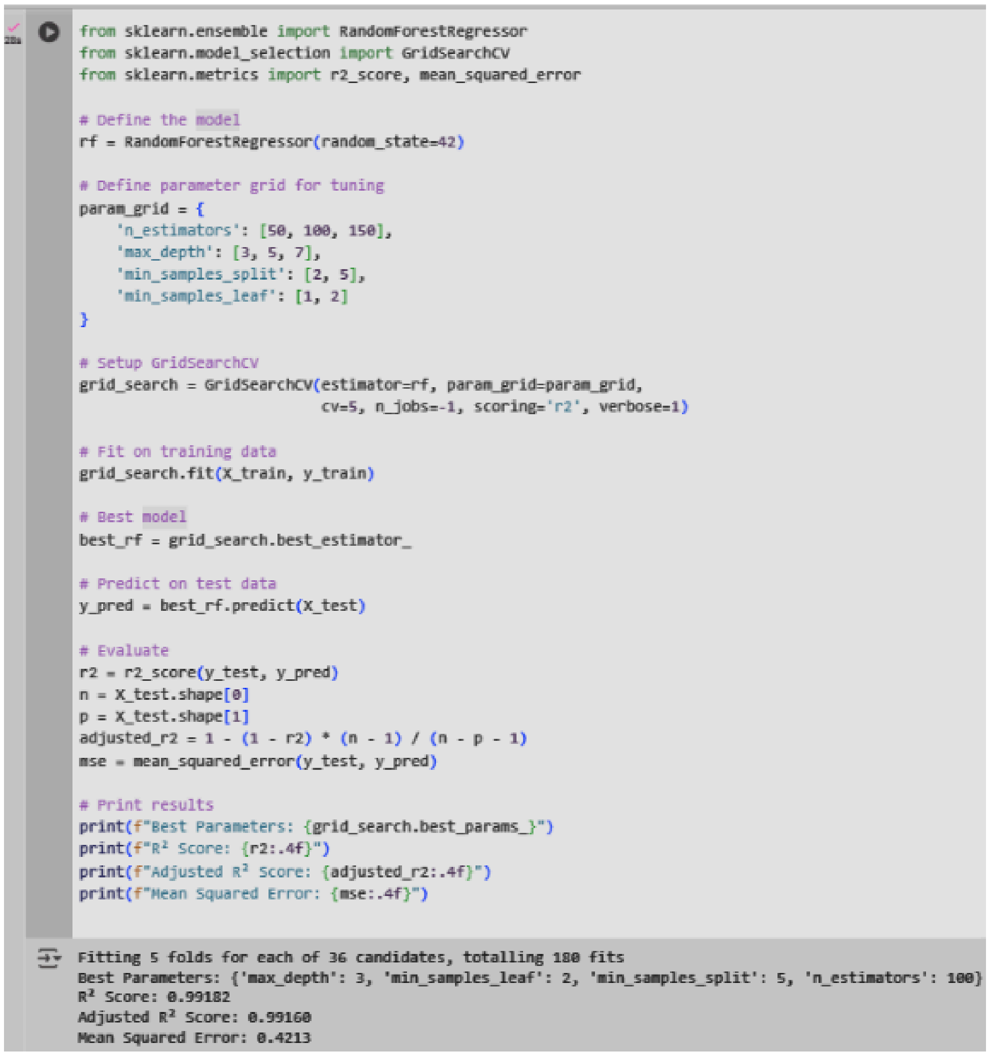
Split data to Train/Test – 70% and 30%



```
# 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)
```

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
```

Optimal Parameters: The model achieves the best performance with a moderately deep tree (max_depth=3), along with specific values for splits and the number of trees.

R^2 Value: 0.99182 — This indicates that the model accounts for 99.18% of the variance in the dependent variables, reflecting excellent accuracy.

Adjusted R^2 Value: 0.99160 — Even after adjusting for the number of predictors, the model still explains 99.16% of the variance, demonstrating strong generalization.

Mean Squared Error: 0.4213 — The small average squared difference between predicted and actual values suggests that the model has minimal prediction error.

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).

Solution:

The Wine Quality dataset is a widely used dataset in machine learning and statistical analysis, containing physicochemical properties and sensory ratings of red and white wines. The primary objective of this dataset is to predict wine quality based on measurable attributes. This assignment explores the key features of the dataset, their significance in determining wine quality, and the techniques used to handle missing data during feature engineering.

Key Features and Their Importance in Predicting Wine Quality

The dataset includes several physicochemical and sensory attributes that influence wine quality. Below is an analysis of the most important features:

1. **Fixed Acidity** – This refers to non-volatile acids (such as tartaric and malic acid) that contribute to the wine's tartness. A balanced level is essential; too much acidity makes the wine taste harsh, while too little results in a flat taste.
2. **Volatile Acidity** – High levels of volatile acidity (primarily acetic acid) can lead to an unpleasant vinegar-like taste, negatively impacting quality.
3. **Citric Acid** – A small amount enhances freshness and flavor complexity, but excessive citric acid can make the wine overly sour.
4. **Residual Sugar** – Determines the sweetness of the wine. While some sugar improves taste, excessive amounts can make the wine cloying.
5. **Chlorides** – Represent the salt content. High chloride levels can make wine taste salty, reducing its appeal.
6. **Free and Total Sulfur Dioxide** – Sulfur dioxide acts as a preservative, preventing oxidation and microbial spoilage. However, excessive amounts can introduce an undesirable sulfurous odor.
7. **Density** – Influenced by alcohol and sugar content, density can indicate the wine's body and mouthfeel.
8. **pH** – Measures acidity on a logarithmic scale. A balanced pH ensures stability and a pleasant taste.
9. **Sulphates** – Additives like potassium sulphate help preserve wine and influence aging. Higher sulphates may correlate with better quality in some cases.
10. **Alcohol (%)** – A key factor in wine quality, alcohol contributes to body, sweetness, and balance. Wines with moderate alcohol levels are generally preferred.

The **target variable** is *quality*, typically rated on a scale from 0 (poor) to 10 (excellent). Understanding how these features interact helps in building accurate predictive models.

Handling Missing Data in Feature Engineering

Missing data is a common issue in real-world datasets and must be addressed carefully to avoid bias. Below are common techniques for handling missing values in the Wine Quality dataset:

1. Deletion Methods

- **Listwise Deletion (Complete Case Analysis)** – Removes any row with missing values.
  - *Advantage*: Simple and avoids introducing imputation bias.
  - *Disadvantage*: Reduces dataset size, potentially losing valuable information.
- **Pairwise Deletion** – Uses available data points for each analysis.
  - *Advantage*: Retains more data than listwise deletion.
  - *Disadvantage*: Can lead to inconsistencies if missingness is not random.

2. Imputation Methods

- **Mean/Median/Mode Imputation** – Replaces missing values with the mean (for numerical data) or mode (for categorical data).
  - *Advantage*: Easy to implement and works well for small missingness.
  - *Disadvantage*: Reduces variance and may distort statistical relationships.
- **Regression Imputation** – Predicts missing values using regression models based on other features.
  - *Advantage*: More accurate than mean imputation if strong correlations exist.
  - *Disadvantage*: Can overfit if relationships are weak.
- **K-Nearest Neighbors (KNN) Imputation** – Uses similarity between data points to estimate missing values.
  - *Advantage*: Works well for datasets with meaningful distance metrics.
  - *Disadvantage*: Computationally expensive for large datasets.
- **Multiple Imputation (MICE – Multivariate Imputation by Chained Equations)** – Generates multiple imputed datasets and combines results.
  - *Advantage*: Accounts for uncertainty and is robust for Missing at Random (MAR) data.
  - *Disadvantage*: More complex to implement and interpret.

Best Practices for the Wine Quality Dataset

- If missing data is minimal (<5%), mean or median imputation may suffice.
- For larger missingness, advanced techniques like MICE or KNN imputation are preferable.
- Before choosing a method, it is crucial to analyze whether the missingness is **Missing Completely at Random (MCAR), Missing at Random (MAR), or Missing Not at Random (MNAR).**

The Wine Quality dataset provides valuable insights into how physicochemical properties influence wine ratings. Understanding feature importance helps in building better predictive models, while proper missing data handling ensures reliable analysis. By using appropriate imputation techniques, we can maintain data integrity and improve model performance. Future work could explore feature interactions and advanced machine learning models for enhanced predictions.