Kaushal Sahebrao Galav Roll no - 11 D15C

AIDS-I Assignment No: 2

Q.1: Statistical Analysis of the Dataset

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

ANS - Mean (10 pts): The mean is calculated by summing all values and dividing by the total number of values. Mean = (82 + 66 + 70 + 59 + 90 + 78 + 76 + 95 + 99 + 84 + 88 + 76 + 82 + 81 + 91 + 64 + 79 + 76 + 85 + 90) / 20 = 1601 / 20 = **80.05**

Median (10 pts): 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 Median = (10th value + 11th value) / 2 = (81 + 82)/2 = **81.5**

Mode (10 pts): The most frequent number is 76, which appears 3 times. Mode = **76**

Interquartile Range (20 pts): Q1 (25th percentile) = median of first half = (5th + 6th)/2 = (76 + 76)/2 = 76 Q3 (75th percentile) = median of second half = (15th + 16th)/2 = (88 + 90)/2 = 89 IQR = Q3 - Q1 = 89 - 76 = **13**

Q.2: Comparison of Machine Learning Tools for Kids

ANS - 1. Machine Learning for Kids

- **Target Audience**: School-aged children (ages 8–16), educators
- **Use by Audience**: Allows students to create ML projects using image, text, or number-based data. Integrated with Scratch and Python.
- **Benefits**: Intuitive, easy to use, free, integrates with existing educational platforms.
- **Drawbacks**: Limited model complexity; less suitable for advanced users.
- **Analytic Type**: **Predictive Analytic -** as it uses trained models to make predictions.
- **Learning Type**: **Supervised Learning -** users provide labeled examples for training.

2. **Teachable Machine**

- **Target Audience**: Beginners, educators, artists, and developers.
- **Use by Audience**: Users can train models using images, audio, or poses through a webcam/microphone.
- **Benefits**: No coding required, real-time interaction, fast model training.
- **Drawbacks**: Models may overfit, not suitable for large-scale applications.
- **Analytic Type**: **Predictive Analytic -** it predicts based on real-time input.
- **Learning Type**: **Supervised Learning -** trained using labeled input data.

Q.3: Misinformation in Data Visualization

ANS - Data visualization is a powerful tool to convey complex information quickly and intuitively. However, when misused—intentionally or unintentionally—it can lead to serious misinformation. This often happens during crises such as the COVID-19 pandemic, where public perception directly influences behavior and policy.

Key Articles and Insights:

1. Arthur Kakande – "What's in a chart?" (Medium)
   - This article provides a step-by-step guide to identifying misleading charts. It highlights common red flags such as:
     - Truncated or manipulated axes
     - Misleading use of color
     - Unlabeled data points or scales
     - Charts without context
   - The goal is to teach readers how to critically evaluate visual data, emphasizing that beautiful graphics can still be deceptive.

2. Katherine Ellen Foley – "How bad Covid-19 data visualizations mislead the public" (Quartz)
   - The article discusses real examples where poor design choices contributed to public misunderstanding.
   - It stresses the responsibility of media and data scientists in avoiding:
     - Overuse of cumulative data (which hides trends)
     - Inappropriate chart types (e.g., pie charts for time series)
     - Inconsistent scales and color coding

Example of Real-World Misinformation:

Case Study: Misleading COVID-19 Charts in U.S. States During the early months of the pandemic, some state governments (notably Georgia and Florida) published bar graphs of COVID-19 cases where:

- The x-axis (dates) was out of order (e.g., jumping from May 7 to May 4 and back to May 6).
- The colors representing counties were reused inconsistently.
- Y-axes were truncated, making trends appear flat even as case numbers surged.

Why This Was Misleading:

- The non-chronological order gave the illusion of declining cases.
- Truncated y-axes minimized visual differences, suggesting improvement
- Inconsistent color mapping confused which regions were most affected.

Impact:

- These visualizations gave citizens and policymakers a false sense of security.
- Some interpreted the misleading visuals as evidence that restrictions could be lifted safely.
- Public health experts criticized these charts as dangerous.

Better Practices:

- Always keep axes clearly labeled and in proper order.
- Avoid truncated y-axes unless justified—and always indicate truncation.

- Use consistent and meaningful color schemes.
- Present both raw and per-capita numbers for clearer comparison.

Cited Source:

- The New York Times, "The Worst Charts of the Pandemic—and How We Can Do Better" https://www.nytimes.com/2020/07/02/upshot/coronavirus-data-charts.html

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 -

```python
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Features where 0 is invalid and should be considered missing
cols_with_zero_invalid = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']

# Replace 0s with NaN
df[cols_with_zero_invalid] = df[cols_with_zero_invalid].replace(0, np.nan)

# Impute missing values with median
df.fillna(df.median(), inplace=True)

# Separate features and target
X = df.drop('Outcome', axis=1)
y = df['Outcome']

# Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train-Validation-Test Split
X_train_val, X_test, y_train_val, y_test = train_test_split(X_scaled, y, test_size=0.10, random_state=42, stratify=y)
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)  # (2

# Final shapes
print("Train set:", X_train.shape)
print("Validation set:", X_val.shape)
print("Test set:", X_test.shape)
```

```
Train set: (537, 8)
Validation set: (154, 8)
Test set: (77, 8)
```

```python
[7] from imblearn.over_sampling import SMOTE
from collections import Counter
```

```python
from imblearn.over_sampling import SMOTE
from collections import Counter

# Apply SMOTE only on the training data (NOT on validation or test)
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

# Check the class distribution before and after
print("Before SMOTE:", Counter(y_train))
print("After SMOTE:", Counter(y_train_resampled))
```

```
Before SMOTE: Counter({0: 350, 1: 187})
After SMOTE: Counter({0: 350, 1: 350})
```

```python
print("\nMissing values:\n", df.isnull().sum())
```

```
Shape of dataset: (768, 9)

Data types:
 Pregnancies                  int64
Glucose                      int64
BloodPressure                int64
SkinThickness                int64
Insulin                      int64
BMI                        float64
DiabetesPedigreeFunction   float64
Age                          int64
Outcome                      int64
dtype: object

Sample data:
    Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1

Missing values:
 Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

```
best_svm = grid_search.best_estimator_

# Evaluate on validation set
y_val_pred = best_svm.predict(X_val)
print("◆ SVM Validation Accuracy:", accuracy_score(y_val, y_val_pred))
print("◆ Classification Report:\n", classification_report(y_val, y_val_pred))
```

```
◆ SVM Validation Accuracy: 0.6948051948051948
◆ Classification Report:
               precision    recall  f1-score   support

           0       0.78      0.73      0.76       100
           1       0.56      0.63      0.59        54

    accuracy                           0.69       154
   macro avg       0.67      0.68      0.67       154
weighted avg       0.71      0.69      0.70       154
```

```
[10] from sklearn.naive_bayes import GaussianNB

nb = GaussianNB()
nb.fit(X_train_resampled, y_train_resampled)

# Evaluate on validation set
y_val_pred_nb = nb.predict(X_val)
print("◆ Naive Bayes Validation Accuracy:", accuracy_score(y_val, y_val_pred_nb))
print("◆ Classification Report:\n", classification_report(y_val, y_val_pred_nb))
```

```
◆ Naive Bayes Validation Accuracy: 0.7662337662337663
◆ Classification Report:
               precision    recall  f1-score   support

           0       0.83      0.80      0.82       100
           1       0.66      0.70      0.68        54

    accuracy                           0.77       154
   macro avg       0.74      0.75      0.75       154
weighted avg       0.77      0.77      0.77       154
```

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

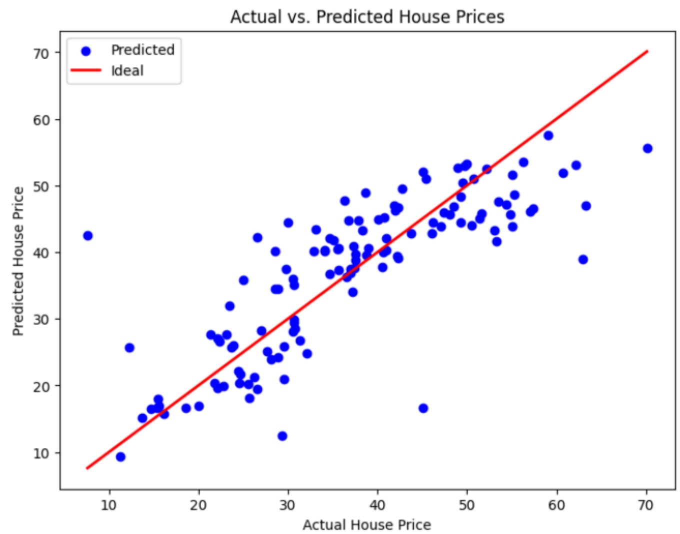
ANS -

```python
# Split the data into training and testing sets
regression_model.split_data()

# Tune hyperparameters and train the best model
regression_model.tune_and_train_model()

# Evaluate the model and print the Adjusted R² score
adj_r2 = regression_model.evaluate_model()
```

```
Best parameters from GridSearch: {'poly__degree': 2, 'ridge__alpha': 10}
R²: 0.6492937127071428
Adjusted R²: 0.6314611896244551
```



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:

- Fixed acidity, Volatile acidity, Citric acid, Residual sugar, Chlorides, Free sulfur dioxide, Total sulfur dioxide, Density, pH, Sulphates, Alcohol

- Importance:
    - Alcohol: Most correlated with quality (higher alcohol, better quality)
    - Volatile Acidity: Negative correlation (lower acidity, better quality)
    - Sulphates and Citric Acid: Add to flavor; positively correlated
- Handling Missing Data:
    - Technique Used: Mean/median imputation for numerical columns
    - Alternatives: KNN Imputation, model-based imputation
- Advantages and Disadvantages:
    - Mean/Median: Easy, fast; may reduce variability
    - KNN: Captures local structure; computationally expensive
    - Model-based: More accurate; risk of bias, overfitting
- Dataset Source: Kaggle Wine Quality Dataset (https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009)