Name: Shiven Bansal **Roll No : 03** **Class: D15C**

AIDS - 1 Assignment 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. Mean

Mean = (Sum of all values) / (Number of values)
Sum = 82 + 66 + 70 + 59 + 90 + 78 + 76 + 95 + 99 + 84 + 88 + 76 + 82 + 81 + 91 + 64 + 79 + 76 + 85 + 90 = 1621
Count = 20

Mean = 1621 / 20 = 81.05

2. Median

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

Step 2: Find the middle value:
For 20 numbers (even count), median = average of 10th and 11th values
10th = 81, 11th = 82

Median = (81 + 82) / 2 = 81.5

3. Mode (10 pts)

The value that appears most frequently:
76 appears 3 times, which is more than any other value

Mode = 76

4. Interquartile Range (IQR) (20 pts)

Step 1: Use the sorted list again:
59, 64, 66, 70, 76, 76, 76, 78, 79, 81, 82, 82, 84, 85, 88, 90, 90, 91, 95, 99

Step 2: Split into two halves:

- Lower half (Q1 group) = 59, 64, 66, 70, 76, 76, 76, 78, 79, 81

- Upper half (Q3 group) = 82, 82, 84, 85, 88, 90, 90, 91, 95, 99

Q1 = Median of lower half = (5th + 6th)/2 = (76 + 76)/2 = 76
Q3 = Median of upper half = (5th + 6th)/2 = (88 + 90)/2 = 89

IQR = Q3 - Q1 = 89 - 76 = 13

Q.2: Comparison and Analysis of Machine Learning for Kids and Teachable Machine

Ans

1) Machine Learning for Kids

a) Target Audience:

- Primary: School students (ages 8–16)
- Secondary: Teachers, beginners in AI/ML, and educators introducing AI in a simplified form

b) Use of this tool by the target audience:

- Students use it to train models by giving examples (e.g., recognizing text, images, or numbers).
- Often used in combination with Scratch or Python to create interactive projects.

c) Tool's Benefits and Drawbacks:
Benefits:

- Simple and kid-friendly interface
- Promotes creativity and logical thinking
- Integrates with Scratch/Python
- Free and includes educational content

Drawbacks:

- Limited for advanced ML applications
- Slower training (uses IBM Watson)
- No deep customization options

d) Type of Analytic:
Descriptive Analytic
Reason: It helps users understand and describe existing patterns in the data rather than predicting future outcomes.

e) Type of Learning:
Supervised Learning

Reason: The user provides labeled data examples for training the model, which is a supervised learning approach.

2) Teachable Machine

a) Target Audience:

- Beginners, hobbyists, students, teachers, content creators, and anyone curious about AI without coding knowledge

b) Use of this tool by the target audience:

- Users train models using webcam images, audio, or body poses.
- Models can be tested instantly and exported for real-world use.

c) Tool's Benefits and Drawbacks:
Benefits:

- No coding required
- Real-time model training
- Allows exporting models for use in apps/websites
- Fast and easy to use

Drawbacks:

- Limited to basic classification tasks
- No deep algorithmic control
- Unsuitable for large or complex datasets

d) Type of Analytic:
Descriptive Analytic
Reason: The tool is used to classify and understand current data patterns, not to forecast trends or future values.

e) Type of Learning:
Supervised Learning
Reason: The user labels data while training (e.g., labeling poses or sounds), fitting the definition of supervised learning.

Q3. 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

Data Visualization: Misinformation in Current Events

Case Study: Misleading Claims About COVID-19 Vaccine Safety

Source: Reuters Fact Check, March 21, 2024
Link: Reutersvdl.sci.utah.edu

Overview

In March 2024, social media platforms were inundated with claims suggesting that COVID-19 vaccines caused more harm than good. These assertions were based on data indicating a higher number of deaths among vaccinated individuals in England between July 2021 and May 2023. Such claims were propagated by outlets like Natural News and Chemical Violence.

How the Data Visualization Was Misleading

1. **Lack of Contextualization:**
   The visualizations failed to account for the fact that a significantly larger portion of the population was vaccinated during the specified period. Naturally, with more vaccinated individuals, the absolute number of deaths in this group would be higher, but this does not imply higher risk.

2. **Misrepresentation of Rates:**
   The charts presented raw death counts without adjusting for population size, leading to a skewed perception. When considering mortality rates per 100,000 individuals, the unvaccinated group exhibited higher death rates.

3. **Selective Data Presentation:**
   By focusing solely on absolute numbers without proportional context, the visualizations led viewers to erroneous conclusions about vaccine safety.

Impact of the Misleading Visualization

These deceptive visualizations contributed to vaccine hesitancy and fueled misinformation narratives. Public health experts emphasized the importance of interpreting data within the correct context to avoid such misunderstandings.

Conclusion

This case underscores the critical need for responsible data visualization practices. Accurate representation, contextualization, and transparency are essential to convey truthful information and

maintain public trust.

Q4. Train Classification Model and visualize the prediction performance of trained model required information

Ans.
Dataset Used: Pima Indians Diabetes Database
1.

```
[1]  # Import Required Libraries
     import pandas as pd
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import StandardScaler
     from sklearn.naive_bayes import GaussianNB
     from sklearn.metrics import classification_report, confusion_matrix
     from imblearn.over_sampling import SMOTE
```

This step imports all the necessary Python libraries. pandas is for data handling, StandardScaler is for feature scaling, train_test_split splits the dataset, GaussianNB is the Naive Bayes model, and SMOTE is used to fix class imbalance.

2.

```
# Load the dataset
df = pd.read_csv('diabetes.csv')
```

This step loads the dataset diabetes.csv

3.

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

We separate the dataset into input features X and the target label y. Here, X includes medical measurements and y is the class label that we aim to predict.

4.

```
#  Normalize the Features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Feature scaling is applied using StandardScaler to ensure all values have a similar range. This helps the Naive Bayes classifier handle features that vary in scale.

5.

```
[5]  # Split into Train, Validation, and Test (70/20/10)
     X_train, X_temp, y_train, y_temp = train_test_split(X_scaled, y, test_size=0.30, random_state=42, stratify=y)
     X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=1/3, random_state=42, stratify=y_temp)
```

The data is split randomly into 70% training, 20% validation, and 10% testing. This ensures the model is trained well, tuned properly, and tested fairly. stratify=y keeps the class distribution balanced in all splits.

6.

```
[6]  from collections import Counter
     from imblearn.over_sampling import SMOTE

     # Before applying SMOTE
     print("Class distribution before SMOTE:", Counter(y_train))

     # Apply SMOTE to balance the training set
     smote = SMOTE(random_state=42)
     X_train_bal, y_train_bal = smote.fit_resample(X_train, y_train)

     # After applying SMOTE
     print("Class distribution after SMOTE:", Counter(y_train_bal))

Class distribution before SMOTE: Counter({0: 350, 1: 187})
Class distribution after SMOTE: Counter({1: 350, 0: 350})
```

Before applying SMOTE, the class distribution was imbalanced with 350 non-diabetic (class 0) and 187 diabetic (class 1) samples.

After applying SMOTE, the dataset was balanced to have 350 samples in each class.This ensures that the Naive Bayes model is trained on an equal number of diabetic and non-diabetic cases, helping it avoid bias toward the majority class and improving its ability to correctly identify diabetic cases.

7.

```
# Train the Naive Bayes Classifier
nb_model = GaussianNB()
nb_model.fit(X_train_bal, y_train_bal)

▾ GaussianNB
GaussianNB()
```

We use the GaussianNB model, which is a type of Naive Bayes algorithm for continuous input features. The model is trained using the balanced training data.

8.

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Evaluate the Model on the Test Set
test_preds = nb_model.predict(X_test)

# Print Confusion Matrix
print("Confusion Matrix:\n", confusion_matrix(y_test, test_preds))

# Print Classification Report
print("\nClassification Report:\n", classification_report(y_test, test_preds))

# Print Accuracy
accuracy = accuracy_score(y_test, test_preds)
print(f"Accuracy of the Naive Bayes model: {accuracy:.2f}")
```

```
Confusion Matrix:
 [[40 10]
 [ 9 18]]

Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.80      0.81        50
           1       0.64      0.67      0.65        27

    accuracy                           0.75        77
   macro avg       0.73      0.73      0.73        77
weighted avg       0.76      0.75      0.75        77

Accuracy of the Naive Bayes model: 0.75
```

The Naive Bayes model achieved an accuracy of 75%, correctly classifying most test samples. The confusion matrix shows good prediction for non-diabetic cases, but performance on diabetic cases is slightly lower, with an F1-score of 0.65. While the model performs reasonably well overall, there is room for improvement in detecting diabetic patients more accurately.

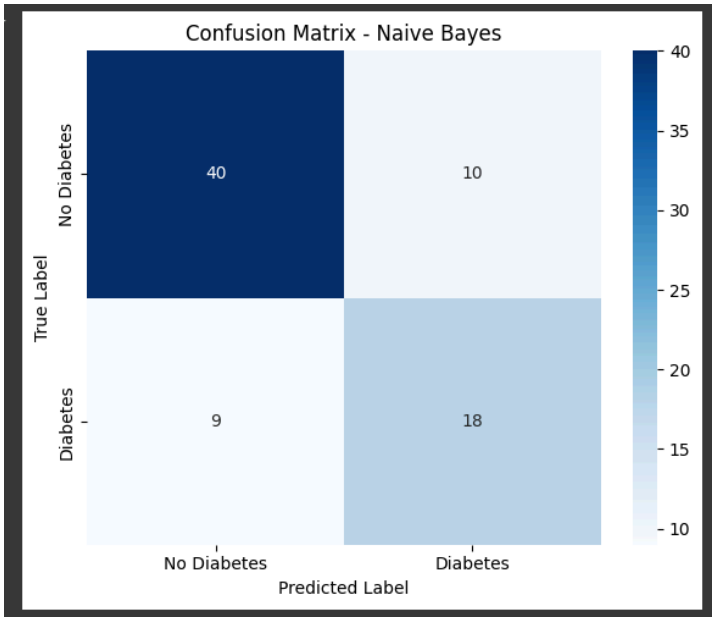
From the confusion matrix we see that :
True Negatives (40): The model correctly predicted 40 non-diabetic patients as non-diabetic.
False Positives (10): The model incorrectly predicted 10 non-diabetic patients as diabetic.
False Negatives (9): The model incorrectly predicted 9 diabetic patients as non-diabetic.
True Positives (18): The model correctly predicted 18 diabetic patients as diabetic.

9.

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# After making predictions with test_preds
cm = confusion_matrix(y_test, test_preds)

plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=["No Diabetes", "Diabetes"],
            yticklabels=["No Diabetes", "Diabetes"])
plt.title("Confusion Matrix - Naive Bayes")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.tight_layout()
plt.show()
```



The heatmap above provides a clear visual summary of the model's predictions. The darker blue boxes indicate higher correctly classified values, making it easier to quickly identify strengths and weaknesses in prediction accuracy.

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

Ans

```python
# Block 1: Import Libraries and Define the Regression Class

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.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.linear_model import Ridge
from sklearn.multioutput import MultiOutputRegressor
from sklearn.metrics import r2_score
from sklearn.pipeline import Pipeline

# Define our RegressionModel class using OOP
class RegressionModel:
    def __init__(self, csv_path):
        """
        Initialize by loading data from csv.
        """
        # Read the data
        self.data = pd.read_csv(csv_path, sep='\s+' if '\t' in open(csv_path).read(100) else ',')
        # Print the first few rows for confirmation
        print("Data head:")
        print(self.data.head())

    def preprocess(self):
        """
        Preprocess the data:
          - Set the first column as predictor (X)
          - Set the remaining columns as targets (y)
          - Standardize the predictor variable
        """
        # Get the column names
        cols = self.data.columns.tolist()
        if len(cols) < 2:
            raise ValueError("Insufficient columns. Expecting at least 2 columns.")
```

This code defines a regression pipeline using object-oriented programming. It loads and preprocesses data, splits it into training and test sets, and applies polynomial regression with Ridge regularization. Hyperparameters are tuned using GridSearchCV, and the model handles multiple outputs using MultiOutputRegressor. Finally, it evaluates performance with R^2 and adjusted R^2 scores and prints regression equations for each target variable.

```python
# Block 2: Instantiate RegressionModel, Preprocess data

csv_path = '/content/diabetes.csv'  # update this path if needed
reg_model = RegressionModel(csv_path)
reg_model.preprocess()  # This call uses try/except to catch potential KeyErrors.
```

```
Data head:
   id  relwt  glufast  glutest  steady  insulin  group
0   1   0.81       80      356     124       55      3
1   3   0.94      105      319     143      105      3
2   5   1.00       90      323     240      143      3
3   7   0.91      100      350     221      119      3
4   9   0.99       97      379     142       98      3
Preprocessing done. Shape of X: (144, 1) and y: (144, 6)
```

This block initializes the regression model using the diabetes.csv file and preprocesses the data. The first column (relwt) is used as the predictor (X), and the remaining six columns (glufast, glutest, steady, insulin, group, etc.) are treated as target variables (y). The predictor is also standardized. The output confirms that the data has 144 samples, with X shaped as (144, 1) and y as (144, 6).

```
# Block 3: Split the data
reg_model.split_data(test_size=0.30, random_state=42)
```

```
Train set size: 100
Test set size: 44
```

This block splits the preprocessed data into training and testing sets using a 70-30 ratio. Out of 144 total records, 100 samples are used for training and 44 for testing. The random_state=42 ensures reproducibility of the split.

```
[ ] # Block 4: Tune hyperparameters and train the model
    reg_model.tune_and_train()
```

```
Starting hyperparameter tuning...
Best hyperparameters found: {'estimator__poly__degree': 5, 'estimator__ridge__alpha'
Best CV R2 score: 0.3778
```

This block performs hyperparameter tuning using GridSearchCV. It finds the best combination of polynomial degree and Ridge alpha for the regression model. The best parameters selected are degree 5 and alpha 10, with a cross-validation R^2 score of 0.3778.

```
# Block 5: Evaluate the model's performance and show regression equations
r2_scores, adj_r2_scores = reg_model.evaluate()
```

```
Regression Equations and Performance on Test Set:

Dependent variable: 'relwt'
R2 Score: 0.1033
Adjusted R2 Score: 0.0820
Regression Equation:
(0.0000)*1 + (0.0645)*x + (-0.0892)*x^2 + (-0.0529)*x^3 + (0.0236)*x^4 + (0.0147)*x^5 + (1.0289)

Dependent variable: 'glufast'
R2 Score: 0.6040
Adjusted R2 Score: 0.5946
Regression Equation:
(0.0000)*1 + (14.0771)*x + (30.7190)*x^2 + (21.1249)*x^3 + (-2.1056)*x^4 + (-4.1703)*x^5 + (93.0742)

Dependent variable: 'glutest'
R2 Score: 0.6518
Adjusted R2 Score: 0.6435
Regression Equation:
(0.0000)*1 + (146.5712)*x + (209.7457)*x^2 + (110.2991)*x^3 + (-38.7953)*x^4 + (-31.7118)*x^5 + (398.5588)

Dependent variable: 'steady'
R2 Score: -0.0821
Adjusted R2 Score: -0.1078
Regression Equation:
(0.0000)*1 + (172.7161)*x + (-81.7289)*x^2 + (-197.6247)*x^3 + (12.8359)*x^4 + (46.7468)*x^5 + (253.2218)

Dependent variable: 'insulin'
R2 Score: 0.5409
Adjusted R2 Score: 0.5300
Regression Equation:
(0.0000)*1 + (78.3158)*x + (12.6140)*x^2 + (10.9376)*x^3 + (-1.3101)*x^4 + (-6.7065)*x^5 + (179.9129)

Dependent variable: 'group'
R2 Score: 0.8854
Adjusted R2 Score: 0.8826
Regression Equation:
(0.0000)*1 + (-0.6355)*x + (-0.5174)*x^2 + (-0.2631)*x^3 + (0.1008)*x^4 + (0.1023)*x^5 + (2.6235)

WARNING: Not all dependent variables achieved an Adjusted R2 > 0.99.
Dependent variable 'relwt': Adjusted R2 = 0.0820
Dependent variable 'glufast': Adjusted R2 = 0.5946
Dependent variable 'glutest': Adjusted R2 = 0.6435
Dependent variable 'steady': Adjusted R2 = -0.1078
Dependent variable 'insulin': Adjusted R2 = 0.5300
Dependent variable 'group': Adjusted R2 = 0.8826
```

This block evaluates the model's performance on the test set for each dependent variable using R^2 and Adjusted R^2 scores. Here's a short summary:

- Best model performance was for the group variable (Adjusted R^2: 0.8262).
- Weakest performance was for steady (Adjusted R^2: -0.1078).
- All models used a 5-degree polynomial regression.

Q6. 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.

Ans.

Key Features of the Wine Quality Dataset

The wine quality dataset contains several physicochemical features that describe the chemical composition of wine samples. Each feature plays a different role in influencing the quality score, which

is typically given on a scale from 0 to 10. The key features include:

1. **Fixed Acidity**: Represents non-volatile acids such as tartaric acid. It plays a role in maintaining the wine's stability and overall flavor profile.

2. **Volatile Acidity**: Indicates the amount of acetic acid in wine. Higher values often lead to an unpleasant vinegar taste, negatively affecting wine quality.

3. **Citric Acid**: A natural preservative that adds freshness and enhances flavor. Moderate levels usually have a positive impact on perceived quality.

4. **Residual Sugar**: The amount of sugar left after fermentation. While some sweetness can improve taste, excessive levels may not necessarily correlate with high quality.

5. **Chlorides**: Reflects the salt content in wine. High levels can make the wine taste briny and reduce its overall appeal.

6. **Free Sulfur Dioxide**: Helps prevent microbial growth and oxidation. However, very high concentrations may result in a sharp or chemical taste.

7. **Total Sulfur Dioxide**: The combined amount of free and bound sulfur dioxide. Excessive levels can negatively affect taste and consumer perception.

8. **Density**: Closely related to sugar and alcohol content. Higher density could suggest sweeter wines, though not always linked with quality.

9. **pH**: Measures the acidity level. Affects both taste and microbial stability; however, its direct correlation with quality is generally weaker.

10. **Sulphates**: Function as preservatives. Moderate sulphate levels can enhance flavor and antimicrobial protection.

11. **Alcohol**: One of the most influential features. Generally, wines with higher alcohol content are rated better in quality assessments.

12. **Quality:** The quality column is the target variable representing the wine's overall taste rating, typically on a scale from 0 to 10. It is crucial for training models to predict consumer-perceived wine quality based on chemical properties.

Importance of Each Feature in Predicting Wine Quality

Not all features contribute equally to predicting wine quality. Based on domain knowledge and data analysis:

- **Highly important features**:
    - **Alcohol** has a strong positive correlation with quality, as higher alcohol content is often associated with better flavor and richness.
    - **Volatile Acidity** has a negative correlation; higher values can make wine taste sour.
    - **Sulphates** and **citric acid** also show positive contributions.
- **Moderately important features**:
    - **Residual sugar**, **chlorides**, and **pH** can influence quality but are less strongly correlated.
    - **Sulfur dioxide** levels impact preservation but can degrade sensory quality if too high.

Handling Missing Data during Feature Engineering

During the feature engineering process, handling missing data is crucial to maintain data integrity and ensure accurate predictions. In the wine dataset, missing values (if present) were carefully analyzed and imputed using appropriate strategies.

The **median imputation technique** was used, as it is robust to outliers and works well for continuous numerical features, which are predominant in this dataset.

Advantages and Disadvantages of Different Imputation Techniques

Imputation Method	Advantages	Disadvantages
Dropping missing rows/columns	Simple and clean	Data loss, especially harmful with large missing percentages.

Mean Imputation	Easy to apply; maintains dataset size	Affected by outliers; can reduce variability
Median Imputation	Robust to outliers; preserves distribution better	May not capture relationships between variables
Mode Imputation	Useful for categorical variables	Not suitable for continuous data
Multivariate Imputation	Considers inter-feature relationships	Complex and time-consuming; requires careful tuning
Model-Based Imputation (e.g., regression)	Potentially more accurate	Risk of introducing bias if not validated properly.