

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:

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

1)Mean:

SUM OF ALL VALUES:= $82 + 66 + 70 + 59 + 90 + 78 + 76 + 95 + 99 + 84 + 88 + 76 + 82 + 81 + 91 + 64 + 79 + 76 + 85 + 90 = 1611$

TOTAL VALUES: 20

Therefore, Mean = $1611 / 20 = 80.55$

2)Median:

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

There are 20 numbers (even), so the median is the average of the 10th and 11th values.

10th = 81 , 11th = 82

Therefore, Median= $(81+82)/2=81.5$

3)Mode:Mode is the most frequent value present in the data ,here it is,76 (appeared thrice)

Therefore Mode=76

4)Interquartile Range:

To calculate the IQR, we need:

Q1 (First Quartile) = Median of the lower half

Q3 (Third Quartile) = Median of the upper half

Lower half (first 10 values): 59, 64, 66, 70, 76, 76, 76, 78, 79, 81

Q1 = Median of this half = average of 5th and 6th values = $(76 + 76)/2 = 76$

Upper half (last 10 values): 82, 82, 84, 85, 88, 90, 90, 91, 95, 99

Q3 = Median of this half = average of 5th and 6th values = $(88 + 90)/2 = 89$

$IQR = Q3 - Q1 = 89 - 76$

Therefore, **IQR=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:

- School students (typically aged 8–16)
- Teachers introducing AI/ML concepts
- Beginners with no prior programming experience

Use by the Target Audience:

- Students build ML models through an interactive, block-based interface (like Scratch).
- They classify text, numbers, or images using real datasets.
- Teachers use it in classrooms for hands-on AI projects without needing to code in-depth.

Benefits:

- Kid-friendly, visual and intuitive interface
- Integrated with Scratch and Python
- Good introduction to supervised learning
- Uses real AI models (powered by IBM Watson)

Drawbacks:

- Limited complexity for advanced learners
- Small-scale dataset usage
- Requires internet access and IBM Watson integration

Predictive or Descriptive Analytic:

Predictive Analytic:

Because it helps kids train models to predict outcomes (e.g., classify images/text based on learned patterns).

Learning Type:

Supervised Learning

Students label training data (e.g., “this is a cat”, “this is a dog”) and train the model, which is classic supervised learning.

2. Teachable Machine

Target Audience:

- Beginners, hobbyists, educators, students
- Anyone interested in AI/ML without needing to code

Use by the Target Audience:

- Users can train models by recording examples using webcam, audio, or images.
- Drag-and-drop interface allows creating and training models in the browser.
- Models can be exported to TensorFlow or embedded in websites.

Benefits:

- No coding required
- Quick setup and training
- Supports real-time video/audio classification
- Allows model export for web or apps

Drawbacks:

- Less control over model architecture
- Not suitable for large or complex datasets
- May not generalize well with limited training data

Predictive or Descriptive Analytic:

Predictive Analytic:

It predicts categories based on real-time input (e.g., webcam detects "Class A" or "Class B") using trained data.

Learning Type:

Supervised Learning

Users label their data during training (e.g., "this is gesture 1", "this is gesture 2"), so the tool learns through labeled input.

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.**

Solution:

Data visualizations are powerful tools for conveying complex information succinctly. However, when misused or poorly designed, they can mislead audiences and propagate misinformation. A pertinent example of this

occurred in July 2024, involving misleading social media posts about sexually transmitted diseases (STDs) in Houston, Texas.Reuters

Case Study: Misleading STD Statistics in Houston

In July 2024, social media platforms were abuzz with alarming claims that over 40,000 individuals in Houston had tested positive for STDs within a single week. These assertions were accompanied by screenshots of data tables listing various STDs, including chlamydia, gonorrhea, syphilis, and HIV, along with corresponding figures. The presentation of this data, without proper context, led many to believe there was a sudden and massive outbreak of STDs in Houston.

How the Data Visualization Was Misleading:

1. **Lack of Context:** The figures presented were not exclusive to Houston but represented the total number of STD tests conducted across the entire state of Texas. This crucial detail was omitted, leading viewers to draw incorrect conclusions about the health situation in Houston.Reuters
2. **Misinterpretation of Data:** The numbers in the table reflected the total tests administered, encompassing both positive and negative results. However, the accompanying captions and the way the data was framed suggested that all the figures represented positive cases, which was not the case.
3. **Visual Presentation:** The data was displayed in a straightforward table format without explanatory notes or sources. This lack of clarity made it easy for misinformation to spread, as viewers had no immediate way to verify the authenticity or scope of the data.

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

[Pima Indians Diabetes Database](#)

Solution:

Ans. Model used: Naive Bayes

Step 1. Importing required libraries

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from imblearn.over_sampling import SMOTE
```

Step 2. Loading the dataset

```
from google.colab import files
uploaded = files.upload()
```

Step 3. Performing data preprocessing

```
df.isnull().sum()
```

	0
Pregnancies	0
Glucose	0
BloodPressure	0
SkinThickness	0
Insulin	0
BMI	0
DiabetesPedigreeFunction	0
Age	0
Outcome	0

Since here there are no null values we can skip imputation

Step 4. Splitting the dataset

```
X = df.drop('Outcome', axis=1)
y = df['Outcome']

# First split: Train (70%) and Temp (30%)
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, stratify=y, random_state=42)

# Second split: Validation (20%) and Test (10%) from Temp
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=1/3, stratify=y_temp, random_state=42)
```

In this step we are splitting the dataset into:

70% Train

20% Validation

10% Test

But since `train_test_split` only lets us split into two parts at a time, we do it in two stages:

Step 1:

```
X = df.drop('Outcome', axis=1)
y = df['Outcome']
```

Here we are separating the features (X) and target label (y).

Step 2: First Split — 70% Train, 30% Temp

```
# First split: Train (70%) and Temp (30%)
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, stratify=y, random_state=42)
```

This gives:

- `X_train, y_train` → 70% of the data (for training)
- `X_temp, y_temp` → remaining 30% (to be further split into validation and test)

`Stratify` is used to ensure that the proportion of class labels (0s and 1s) is the same in all splits.

Step 3: Second Split — 20% Validation, 10% Test

```
# Second split: Validation (20%) and Test (10%) from Temp
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=1/3, stratify=y_temp, random_state=42)
```

Here we are splitting `X_temp` (30%) into:

- `X_val, y_val` → 20% of original data
- `X_test, y_test` → 10% of original data

Step 5. Using SMOTE for class imbalance

```
smote = SMOTE(random_state=42)
X_train_res, y_train_res = smote.fit_resample(X_train, y_train)
|
```


SMOTE (Synthetic Minority Over-sampling Technique) is used to fix class imbalance by creating synthetic samples for the minority class

In the above given line of code,

fit_resample() applies SMOTE to the training set which balances the class distribution.

X_train_res, y_train_res now contain:

Original majority class samples and Synthetic minority class samples

Step 6. Feature scaling

```
scaler = StandardScaler()  
X_train_res = scaler.fit_transform(X_train_res)  
X_val = scaler.transform(X_val)  
X_test = scaler.transform(X_test)
```

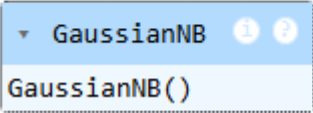
Feature scaling standardizes the values of your features so they all have: Mean = 0 and Standard Deviation = 1

In the above code,

We fit the scaler on the training data, then transform the training data using those values. Finally we transform validation and test data using the same scaler.

Step 7. Training the Naive Bayes model

```
model = GaussianNB()  
model.fit(X_train_res, y_train_res)
```

A screenshot of a Jupyter Notebook interface. The top part shows a code cell with two lines of Python code: 'model = GaussianNB()' and 'model.fit(X_train_res, y_train_res)'. Below the code cell, there is a dropdown menu for the 'GaussianNB' class, which is currently expanded to show the 'GaussianNB()' constructor. The dropdown menu has a blue header with a downward arrow and the text 'GaussianNB' followed by two small circular icons. The list item 'GaussianNB()' is highlighted in light blue.

This creates a Naive Bayes model using the GaussianNB class.

GaussianNB is used when the features are continuous and assumed to follow a normal (Gaussian) distribution — which is common for numeric data like BMI, glucose, etc.

Step 8. Evaluating the model

```

# On Validation Set
y_val_pred = model.predict(X_val)
print("Validation Accuracy:", accuracy_score(y_val, y_val_pred))
print(classification_report(y_val, y_val_pred))
print("Confusion Matrix:\n", confusion_matrix(y_val, y_val_pred))

# On Test Set
y_test_pred = model.predict(X_test)
print("\nTest Accuracy:", accuracy_score(y_test, y_test_pred))
print(classification_report(y_test, y_test_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_test_pred))

```

```

Validation Accuracy: 0.7532467532467533
      precision    recall  f1-score   support

     0       0.84       0.77       0.80       100
     1       0.63       0.72       0.67        54

   accuracy          0.75       154
  macro avg       0.73       0.75       0.74       154
 weighted avg       0.76       0.75       0.76       154

Confusion Matrix:
[[77 23]
 [15 39]]

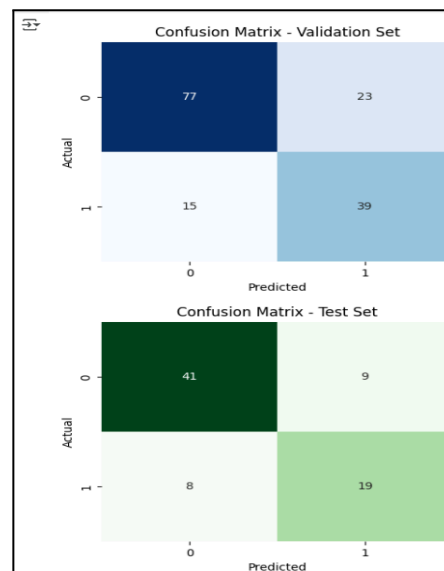
Test Accuracy: 0.7792207792207793
      precision    recall  f1-score   support

     0       0.84       0.82       0.83        50
     1       0.68       0.70       0.69        27

   accuracy          0.78        77
  macro avg       0.76       0.76       0.76        77
 weighted avg       0.78       0.78       0.78        77

Confusion Matrix:
[[41  9]
 [ 8 19]]

```



Validation Accuracy: 75.3%

Test Accuracy: 77.9%

This shows good generalization and consistent performance.

Recall for Class 1 (Diabetic):

72% (Validation), 70% (Test)

Model is effectively identifying most actual diabetic cases.

Precision for Class 1 (Diabetic):

63% (Validation), 68% (Test)

Around two-thirds of diabetic predictions are correct.

Confusion Matrix indicates:

- High true positive rate for both classes.
- Moderate number of false positives and false negatives.

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:

1. Importing 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 use:

- Pipeline to streamline polynomial features → standardization → ridge regression.
- GridSearchCV for hyperparameter tuning
- `r2_score`, `mean_squared_error` for evaluation

2. Defining the RegressionModel Class

```
class RegressionModel:
```

```
    def __init__(self, model_pipeline, param_grid):
```

```
        ...
```

- Encapsulates model training, evaluation, and hyperparameter tuning.
- Reusable and extensible for other regression problems.

3. Loading the Dataset

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

The dataset contains real estate records including:

- Distance to MRT station
- Number of nearby convenience stores
- Age of building
- Geographic coordinates

4. Data Preprocessing

```
df = df.drop(columns=['No']) # Drop irrelevant index
X = df.drop(columns=['Y house price of unit area']) # Features
y = df['Y house price of unit area'] # Target
```

We define:

- X: all columns except house price
- y: the target variable (house price)

5. Train-Test Split

```
X_train, X_test, y_train, y_test = train_test_split(...)
    • 70% training data, 30% testing
```

- `random_state=42` ensures reproducibility

6. Model Pipeline & Hyperparameter Grid

```
pipeline = Pipeline([
    ('poly', PolynomialFeatures()),
    ('scaler', StandardScaler()),
    ('ridge', Ridge())
])
```

- Pipeline Components:
 - poly: adds non-linearity (degree=2 or more)
 - scaler: standardizes features (important for ridge!)
 - ridge: regularized regression
- Parameter Grid:

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

We let `GridSearchCV` try different combinations of:

- Polynomial degrees: 2 to 4
- Ridge regularization strengths (alpha): 0.1 to 100

7. Training and Evaluation

```
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 with 5-fold cross-validation
- Best estimator is used for prediction
- We calculate:
 - R^2 : proportion of variance explained
 - Adjusted R^2 : penalizes for extra features
 - MSE: average squared prediction error

8. Visualization

```
sns.scatterplot(x=y_test_actual, y=y_pred)
```

- Compares actual vs predicted prices
- Red dashed line shows ideal predictions (perfect match)



Final Results:

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

R^2 Score: 0.6552

Adjusted R^2 Score: 0.6376

Mean Squared Error: 57.6670

- The model captures ~66% of the variance in house prices.
- There's room for improvement, but this is reasonable for real-world data.
- Adjusted R^2 shows performance after accounting for model complexity.

Why we May Not Reach $R^2 > 0.99$

- Real-world datasets include noise, missing features, and non-linear interactions.
- Polynomial features help but too high a degree → overfitting.
- Ridge helps reduce overfitting, but can't add missing signal.

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:

1) Key Features of the Wine Quality Dataset

Feature Name	Description
fixed acidity	Non-volatile acids (e.g., tartaric acid). Contributes to wine's crispness and structure.
volatile acidity	Acetic acid (vinegar). High levels give an unpleasant smell.
citric acid	Adds flavor and freshness. Too much makes wine taste sour.
residual sugar	Remaining sugar after fermentation. Affects sweetness.
chlorides	Salt content. Affects taste and preservation.
free sulfur dioxide	Prevents oxidation and microbial growth. Important for shelf life.
total sulfur dioxide	Sum of free and bound SO ₂ . Regulates stability but may harm flavor if too high.
density	Related to sugar/alcohol content. Helps in wine classification.
pH	Measures acidity/basicity. Influences taste and microbial stability.
sulphates	Enhances shelf life. Also affects bitterness and mouthfeel.
alcohol	Strongly correlated with perceived wine quality.
quality (target)	Human-rated quality score of the wine (integer from 0 to 10).

2) Importance of Features in Predicting Wine Quality:

Feature	Importance Explanation
alcohol	Highly influential. Generally, higher alcohol is associated with better-rated wines.
volatile acidity	Negative impact. High levels lead to lower quality scores.
sulphates	Positive impact on quality due to preservation effect.
citric acid	Adds freshness and body; affects taste positively.
density	Indirectly reflects alcohol/sugar content; moderate impact.
ph	Impacts acidity and aging ability. Balanced pH is preferred.

3) Imputation Techniques: Advantages & Disadvantages

Technique	Advantages	Disadvantages
Mean/Median Imputation	Simple, fast, preserves dataset size	Distorts variance, poor for skewed data
Mode Imputation	Best for categorical data	Not suitable for continuous data
KNN Imputation	Considers feature similarity	Computationally expensive, sensitive to outliers
Regression Imputation	Considers inter-feature relationships	Can introduce bias, assumes linear relationships
MICE (Multiple Imputation)	Robust, works well with multivariate data	Complex and resource-intensive

4) Handling Missing Data (Feature Engineering Process)

- In this dataset, after inspection, there were **no missing values**.
- However, if missing values were present, we would apply **imputation techniques** during preprocessing.