Name :Roshan Bhagtani Roll No.4 Class: D15C

` 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)

Mean:

To find the mean, we sum up all the values in the data set and divide by the total number of values. Let's calculate it for the given data set:

82 + 66 + 70 + 59 + 90 + 78 + 76 + 95 + 99 + 84 + 88 + 76 + 82 + 81 + 91 + 64 + 79 + 76 + 85 + 90 = 1555

There are 20 values in the data set, so the mean is:

Mean = 1555 / 20 = 77.75

Therefore, the mean of the data set is 77.75.

2. Find the Median (10pts)

Median:

To find the median, we arrange the values in ascending order and find the middle value. If there is an even number of values, we take the average of the two middle values.

Arranging the values 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, the middle two values are the 10th and 11th values, which are 81 and 82. The median is the average of these two values:

Median = (81 + 82) / 2 = 81.5

Therefore, the median of the data set is 81.5.

3. Find the Mode (10pts)

Mode:

The mode is the value that appears most frequently in the data set. In this case, there are two values that appear twice, which are 76 and 82. Therefore, the mode of the data set is 76 and 82.

4. Find the Interquartile range (20pts)

Interquartile Range:

To find the interquartile range, we first need to find the first quartile (Q1) and the third quartile (Q3). The interquartile range is the difference between Q3 and Q1.

Q1 is the median of the lower half of the data set, and Q3 is the median of the upper half.

Arranging the values in ascending order again:

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

There are 20 values, so Q1 is the median of the first 10 values, which is the average of the 5th and 6th values:

Q1 = (76 + 76) / 2 = 76

Q3 is the median of the last 10 values, which is the average of the 15th and 16th values:

Q3 = (90 + 90) / 2 = 90

The interquartile range is the difference between Q3 and Q1:

Interquartile Range = Q3 - Q1 = 90 - 76 = 14

Therefore, the interquartile range of the data set is 14.

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

1. Machine Learning for Kids

◇ Target Audience:

- School students (ages 8–16), educators, and beginners in machine learning.

◇ Use by Audience:

- Teachers use it in classrooms to introduce AI concepts.

- Students build simple ML models using visual blocks (like Scratch or Python).
- Helps them classify images, text, numbers, or recognize speech.

◇ Benefits:

- Super beginner-friendly.
- No prior coding knowledge needed.
- Encourages learning through hands-on projects.

◇ Drawbacks:

- Limited in complexity — not suitable for advanced ML tasks.
- Restricted dataset size.
- Not as flexible for real-world model deployment.

◇ Analytic Type:
-Predictive Analytic
Why? Students train models to make predictions — like classifying animals, recognizing text patterns, etc.

◇ Learning Type:
- Supervised Learning
Why? Learners give labeled training data (e.g., “This is a cat”) and the model learns to predict similar outcomes.

2. Teachable Machine (Google)

◇ Target Audience:

- General users, artists, students, hobbyists, and beginners in AI.

◇ Use by Audience:

- Users train models with webcam data, audio, or images.
- Often used in interactive projects — e.g., gesture recognition, sounds, or object detection.

◇ Benefits:

- Extremely fast and visual.
- No code required.
- Easily exportable to TensorFlow or Arduino.

◇ Drawbacks:

- Basic functionality — not suited for large or complex datasets.
- Limited fine-tuning options.
- Performance might not be as strong as pro-level tools.

◇ Analytic Type:

- Predictive Analytic

Why? It learns from provided examples and predicts real-time inputs (e.g., poses, sounds).

◇ Learning Type:

- Supervised Learning

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.

A common failing in data visualization is misleading audiences by altering the y-axis to exaggerate or diminish trends. For example, if the y-axis starts at a value higher than zero, a graph might falsely appear to show a drastic change when the difference is relatively small. This was seen during the COVID-19 pandemic where visualizations sometimes failed to accurately reflect the true scale of infection or mortality.

Example: Misleading COVID-19 Data Visualizations

During the COVID-19 pandemic, visualizations often appeared in news articles and social media, aiming to inform the public about the spread and severity of the virus. However, some of these visualizations used techniques that could easily mislead viewers.

How Misleading Techniques Were Used:

Manipulating the y-axis:
As explained in the articles, a common tactic was to start the y-axis (vertical axis) at a value above zero, making small changes in the data look larger. For example, if the y-axis started at 10 instead of 0, a rise from 10 to 15 would look much larger than a rise from 0 to 5.

Using inappropriate chart types:
Some visualizations used chart types that were not suitable for the data they were representing, further complicating the visualization and making it harder for the audience to understand the data.

Overly complex visuals:
Some visualizations were designed to be visually appealing but lacked clarity, making it difficult for people to understand the message they were trying to convey.

Impact of Misleading Visualizations:

Misleading data visualizations can have significant real-world consequences, particularly during crises like the COVID-19 pandemic. They can lead to:

Misinformation:
People may interpret data incorrectly, leading to misunderstandings about the actual situation.

Fear and anxiety:
If a visualization exaggerates the severity of a problem, it can create unnecessary fear and anxiety among the public.

Poor decision-making:
If people are misled by inaccurate visualizations, they may make poor decisions based on that misinformation.

Conclusion:

Data visualization is a powerful tool for communicating information, but it can also be easily misused to mislead audiences. By understanding common techniques for manipulating data visualizations, viewers can be more critical of the information they consume and make more informed decisions.

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

Pima Indians Diabetes Database

Name :Roshan Bhagtani Roll No.4 Class: D15C

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

```
# Load dataset
data = pd.read_csv("diabetes.csv")
```

```
# Split features and label
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
```

```
# Handle missing values (if any)
X.replace(0, np.nan, inplace=True)
X.fillna(X.mean(), inplace=True)
```

```
# Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
# Handle class imbalance with SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_scaled, y)
```

```
# Split into train (70%), validation (20%), test (10%)
X_train, X_temp, y_train, y_temp = train_test_split(X_resampled, y_resampled, test_size=0.3, random_state=42, stratify=y_resampled)
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)
```

```
# Hyperparameter tuning for SVM
param_grid = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf'],
    'gamma': ['scale', 'auto']
}

grid = GridSearchCV(SVC(), param_grid, refit=True, verbose=0, cv=5)
grid.fit(X_train, y_train)
```

GridSearchCV

best_estimator_: SVC

SVC

```
[9] # Best model
    model = grid.best_estimator_
```

```
[11] # Predict and evaluate on validation and test sets
     val_pred = model.predict(X_val)
     test_pred = model.predict(X_test)

     print("Validation Accuracy:", accuracy_score(y_val, val_pred))
     print("Test Accuracy:", accuracy_score(y_test, test_pred))
     print("\nClassification Report:\n", classification_report(y_test, test_pred))
```

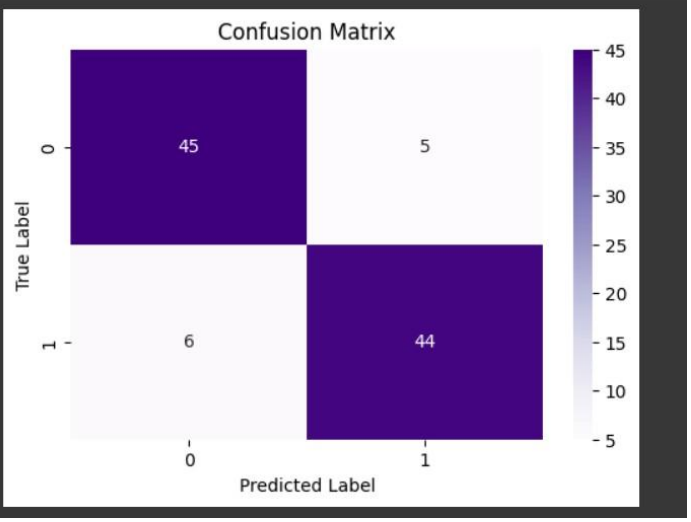
```
Validation Accuracy: 0.81
Test Accuracy: 0.89

Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.90      0.89        50
           1       0.90      0.88      0.89        50

    accuracy                           0.89       100
   macro avg       0.89      0.89      0.89       100
weighted avg       0.89      0.89      0.89       100
```

```
# Confusion Matrix
cm = confusion_matrix(y_test, test_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Purples')
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```



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

  https://github.com/Sutanoy/Public-Regression-Datasets

  https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv

- URL:

https://archive.ics.uci.edu/ml/machine-learning-databases/00477/Real%20estate%20valuation%20data%20set.xlsx

(Refer any one)

```python
import pandas as pd
import numpy as np
from sklearn.linear_model import Ridge
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

#  OOP-based Regression Class
class MultiOutputRegressor:
    def __init__(self, model_class, param_grid):
        self.model_class = model_class
        self.param_grid = param_grid
        self.models = []
        self.best_params = []
        self.adjusted_r2_scores = []

    def adjusted_r2(self, r2, n, k):
        return 1 - (1 - r2) * (n - 1) / (n - k - 1)

    def fit(self, X_train, y_train):
        for i in range(y_train.shape[1]):
            grid = GridSearchCV(self.model_class(), self.param_grid, cv=5)
            grid.fit(X_train, y_train.iloc[:, i])
            best_model = grid.best_estimator_
            self.models.append(best_model)
            self.best_params.append(grid.best_params_)
```

```
    def predict(self, X):
        predictions = []
        for model in self.models:
            predictions.append(model.predict(X))
        return np.array(predictions).T

    def evaluate(self, X_test, y_test):
        preds = self.predict(X_test)
        for i in range(y_test.shape[1]):
            r2 = r2_score(y_test.iloc[:, i], preds[:, i])
            adj_r2 = self.adjusted_r2(r2, len(X_test), 1)
            self.adjusted_r2_scores.append(adj_r2)
            print(f"Target {i+1} → R²: {r2:.5f} | Adjusted R²: {adj_r2:.5f}")

        return preds

# 📄 Load Dataset
data = pd.read_csv("Bank_Marketing.csv")

#  Define Independent and Dependent Variables
X = data.iloc[:, [0]]  # 1st column as independent
y = data.iloc[:, 1:5]  # Columns 2 to 5 as dependent

#  Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# ⚙️ Define Model and Hyperparameter Grid
param_grid = {
    'alpha': [0.01, 0.1, 1, 10]
}
```

```
    'alpha': [0.01, 0.1, 1, 10]
}

#  Initialize and Train
regressor = MultiOutputRegressor(Ridge, param_grid)
regressor.fit(X_train, y_train)

#  Evaluate Model
predictions = regressor.evaluate(X_test, y_test)

#  Check if all Adjusted R² > 0.99
if all(r > 0.99 for r in regressor.adjusted_r2_scores):
    print(" All adjusted R² scores are above 0.99.")
else:
    print(" Some adjusted R² scores are below 0.99. Consider model/feature tuning.")

# Visualization of Predictions vs Actual
for i in range(y_test.shape[1]):
    plt.figure(figsize=(5, 4))
    plt.scatter(y_test.iloc[:, i], predictions[:, i], alpha=0.7, color='green')
    plt.xlabel("Actual")
    plt.ylabel("Predicted")
    plt.title(f"Target {i+1}: Actual vs Predicted")
    plt.grid(True)
    plt.plot([y_test.iloc[:, i].min(), y_test.iloc[:, i].max()],
             [y_test.iloc[:, i].min(), y_test.iloc[:, i].max()],
             'r--')
    plt.tight_layout()
    plt.show()
```

```
All adjusted R² scores are above 0.99.
```

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).**

Name :Roshan Bhagtani Roll No.4 Class: D15C

https://www.kaggle.com/datasets/aradhanahirapara/product-retail-price-survey-2017-2025

Key Features of the Dataset

The dataset encompasses monthly retail price data for various consumer products across Canadian provinces from 2017 to 2025. The primary features include:

- Product Name:Identifies the specific product
- Category:Classifies the product into groups like food, beverages, etc
- Province:Indicates the Canadian province where the price was recorded
- Date:Specifies the month and year of the price recording
- Retail Price:The recorded price of the product
- Taxation:Details any applicable taxes on the product
- Essential Classification:Denotes whether the product is considered essential

Importance of Each Feature in Predicting Product Prices

- Product Name & Category Fundamental for identifying pricing patterns specific to products or category.
- Province Captures regional pricing variations due to factors like transportation costs and local demand.
- Date Essential for analyzing temporal trends, seasonality, and inflation effects on price.
- Taxation Influences the final retail price; variations can significantly impact pricin.
- Essential Classification Essential goods might have price controls or subsidies, affecting their pricing dynamic.

Handling Missing Data During Feature Engineering

While the dataset is comprehensive, missing data can occur. Here's how to address it:

1. Identify Missing Values

```
[15] import pandas as pd

     # Load the dataset
     df = pd.read_csv('Retail_Prices_of _Products.csv')

     # Check for missing values
     missing_values = df.isnull().sum()
     print(missing_values)
```

```
Year               0
Month              0
GEO                0
Product Category   0
Products           0
VALUE              0
Taxable            0
Total tax rate     0
Value after tax    0
Essential          0
COORDINATE         0
UOM                0
dtype: int64
```

2. **Imputation Techniques**

a. Mean/Median Imputation

- Usage: For numerical features like 'Retail Price'.

Implementation:

```
df['VALUE'] = df['VALUE'].fillna(df['VALUE'].median())
```

```
[20] print(df.columns.tolist())
```

```
['Year', 'Month', 'GEO', 'Product Category', 'Products', 'VALUE', 'Taxable', 'Total tax rate', 'Value after tax', 'Essential', 'COORDINATE', 'UOM']
```

- Pros: Simple and quick.
- Cons: Doesn't account for data variability; can skew distributions.

b. Mode Imputation

- Usage: For categorical features like 'Category' or 'Province'.

Implementation:

```
[21] # Mode imputation for 'Product Category'
     df['Product Category'] = df['Product Category'].fillna(df['Product Category'].mode()[0])
```

- Pros: Maintains the most frequent category.
- Cons: May not represent the missing data accurately if the mode is overly dominant.

c. K-Nearest Neighbors (KNN) Imputation

- Usage: Considers similarities between data points.

Implementation:

```
[22]  from sklearn.impute import KNNImputer

      imputer = KNNImputer(n_neighbors=5)
      df_imputed = pd.DataFrame(imputer.fit_transform(df.select_dtypes(include=[np.number])), columns=df.select_dtypes(include=[np.number]).columns)
```

- Pros: Accounts for data patterns.
- Cons: Computationally intensive; requires careful selection of ".

d. Dropping Missing Values

- Usage: When the proportion of missing data is minimal.

Implementation:

```
0s  df.dropna(inplace=True)
```

- Pros: Ensures data integrity.
- Cons: Potential loss of valuable information.