Name : Advik Hegde **Div : D15C** **Roll No. : 15**

AIDS 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)

Answer :

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

1. Mean (10 pts)

To find the mean, sum all numbers and divide by the total count.

Mean = (82 + 66 + 70 + 59 + 90 + 78 + 76 + 95 + 99 + 84 + 88 + 76 + 82 + 81 + 91 + 64 + 79 + 76 + 85 + 90) ÷ 20
Mean = 1621 ÷ 20
Mean = 81.05

2. Median (10 pts)

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: Median is the average of the 10th and 11th values:
Median = (81 + 82) ÷ 2
Median = 81.5

3. Mode (10 pts)

The number that appears most frequently is:

- 76 (appears 3 times) and Others appear only once or twice.

Mode = 76

4. Interquartile Range (IQR) (20 pts)

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

Step 2:

- Q1 (25th percentile) = average of 5th and 6th values
  Q1 = (76 + 76) ÷ 2 = 76

- Q3 (75th percentile) = average of 15th and 16th values
  Q3 = (88 + 90) ÷ 2 = 89

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

Answer :

1) Machine Learning for Kids

- **Target Audience:**
  Primarily designed for school-age students (ages 8–16) and beginners in machine learning and AI.

- **Use by Target Audience:**
  Students use this tool to create simple machine learning models by training them with labeled examples. They often apply it in projects using platforms like Scratch or Python to build games, quizzes, or basic chatbots.

- **Benefits:**

    - Easy-to-understand interface tailored for young learners.
    - Encourages learning by doing hands-on experiments.
    - Helps demystify AI concepts in a classroom setting.

- **Drawbacks:**
    - Limited to basic models and functionality; not suitable for advanced projects.
    - Depends heavily on pre-built templates and may not provide much flexibility for deeper exploration.

2) Teachable Machine (by Google)

- **Target Audience:**
  Designed for general users, educators, students, and creatives who want to experiment with machine learning without coding.

- **Use by Target Audience:**
  Users train models using webcam images, sound, or pose data directly in the browser. It's often used in classrooms, art projects, or personal explorations to classify images, gestures, or sounds.

- **Benefits:**
    - No coding skills required.
    - Extremely user-friendly and quick to get results.
    - Allows export of models for real-world applications (e.g., TensorFlow.js or Arduino).

- **Drawbacks:**
    - Models are simple and may not generalize well to complex data.
    - Limited customization or control over algorithm details.

2. Analytic Type

- **Machine Learning for Kids:**
  Predictive Analytic
  Because it is used to build models that predict outcomes based on training data (e.g.,

predicting the correct answer based on input text or images).

- **Teachable Machine:**
  Predictive Analytic
  It also creates models that make predictions based on user-provided data (e.g., recognizing if the image is of a cat or a dog).

3. Learning Type

- **Machine Learning for Kids:**
  Supervised Learning
  The tool works by training models using labeled data (e.g., “this is happy,” “this is sad”), which is the core concept of supervised learning.

- **Teachable Machine:**
  Supervised Learning
  Users label their own training data (e.g., “Class 1,” “Class 2”), and the model learns to distinguish between them based on these labels.

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.

Anwer :

1. Key Takeaways from the Articles

a. "What’s in a Chart? A Step-by-Step Guide to Identifying Misinformation in Data Visualization" by Arthur Kakande

Article Link

Arthur Kakande emphasizes that data visualizations can be misleading due to several common practices:

- **Truncated Y-Axis:** Starting the Y-axis at a value other than zero can exaggerate differences between data points, leading to misinterpretation.

- **Correlation vs. Causation:** Visuals may imply a causal relationship where only a correlation exists, misleading viewers about the true nature of the data.
- **Color Misuse:** Using colors contrary to common associations (e.g., red for positive outcomes) can confuse the audience.
- **Trend Manipulation:** Altering scales or omitting data points can distort the perceived trend.

Kakande advocates for critical evaluation of charts, encouraging viewers to scrutinize axes, scales, and color usage to detect potential misinformation.

b. "How Bad COVID-19 Data Visualizations Mislead the Public" by Katherine Ellen Foley

Article Link

Katherine Ellen Foley discusses how, during the COVID-19 pandemic, some state health departments in the U.S. presented data in ways that hindered public understanding:

- **Snapshot Data Without Trends:** Presenting daily case numbers without historical context made it difficult to discern whether situations were improving or worsening.
- **Cluttered Visuals:** Overloading charts with information without clear organization led to confusion.
- **Ineffective Use of Pie Charts:** Pie charts were used inappropriately, making it hard for viewers to accurately interpret proportions.

Foley suggests that clear, context-rich, and appropriately designed visualizations are crucial for effective public communication.

2. Recent Event Example: Misleading COVID-19 Visualizations by Anti-Mask Groups

A study by MIT's Visualization Group highlighted how anti-mask communities created misleading data visualizations to challenge public health guidelines. These groups produced charts that mimicked scientific rigor but were based on flawed interpretations

Article Link

- **Selective Data Representation:** Focusing on data that supported their views while ignoring contradictory information.

- **Misinterpretation of Data:** Drawing conclusions that were not supported by the data, such as underestimating the severity of the pandemic.

- **Aesthetic Mimicry:** Designing visuals that appeared scientifically credible to lend unwarranted legitimacy to their claims.

These practices contributed to public confusion and resistance to health measures.

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 : Advik Hegde Div : D15C Roll No. : 15

Step 1: Set up and import the necessary libraries and upload the dataset into the runtime environment.

```
# Install necessary packages
!pip install -q imbalanced-learn scikit-learn

# Import required libraries
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.naive_bayes import GaussianNB
from sklearn.svm import SVC
from imblearn.over_sampling import SMOTE
import matplotlib.pyplot as plt
import seaborn as sns
```

```
[2] # Upload the file manually if needed
from google.colab import files
uploaded = files.upload()

# Read the CSV
df = pd.read_csv('diabetes.csv')  # Change this to the uploaded filename
df.head()
```

Step 2 : Data Preprocessing

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

# Replace 0s in some columns with NaN where 0 is not a valid value
columns_with_zeros = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']
df[columns_with_zeros] = df[columns_with_zeros].replace(0, np.nan)

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

# Separate features and target
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
```

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

Step 3 : Handling Class Imbalances using the SMOTE technique

```
# Apply SMOTE to balance the classes
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

# Check class distribution
print("Class distribution after SMOTE:")
print(pd.Series(y_resampled).value_counts())
```

```
Class distribution after SMOTE:
Outcome
1    500
0    500
Name: count, dtype: int64
```

Step 4: Train-Validation-Test Split (70/20/10)

```
[5] # First split into 90% train_val and 10% test
    X_train_val, X_test, y_train_val, y_test = train_test_split(
        X_resampled, y_resampled, test_size=0.10, random_state=42, stratify=y_resampled)

    # Now split 90% into 70% train and 20% validation (from the 90%)
    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)
```

Step 5: Feature Scaling

```
[6] scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_val = scaler.transform(X_val)
    X_test = scaler.transform(X_test)
```

Step 7: Model Training with Hyperparameter Tuning

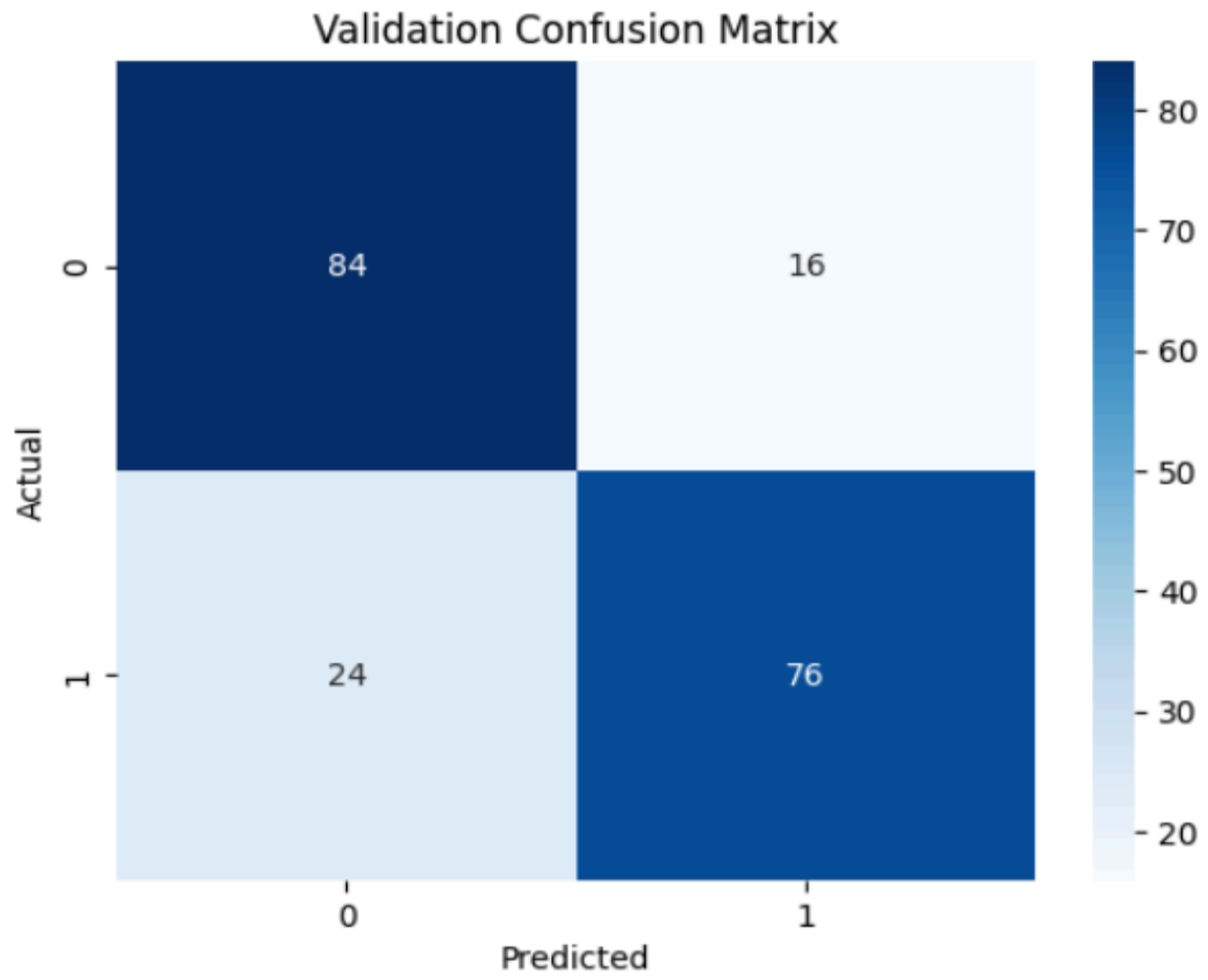
```
[7]  param_grid = {
         'C': [0.1, 1, 10],
         'kernel': ['linear', 'rbf'],
         'gamma': ['scale', 'auto']
     }

     svm = SVC()
     grid = GridSearchCV(svm, param_grid, cv=5)
     grid.fit(X_train, y_train)

     print("Best SVM Parameters:", grid.best_params_)

     # Validate
     y_val_pred = grid.predict(X_val)
```

```
Best SVM Parameters: {'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}
```

Step 8: Evaluate the Model (Validation)

```
print("Validation Accuracy:", accuracy_score(y_val, y_val_pred))
print("\nClassification Report:\n", classification_report(y_val, y_val_pred))

# Confusion Matrix
conf_matrix = confusion_matrix(y_val, y_val_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title("Validation Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

```
Validation Accuracy: 0.8

Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.84      0.81       100
           1       0.83      0.76      0.79       100

    accuracy                           0.80       200
   macro avg       0.80      0.80      0.80       200
weighted avg       0.80      0.80      0.80       200
```



Step 9: Final Test Evaluation

```
y_test_pred = grid.predict(X_test) if 'grid' in locals() else nb.predict(X_test)

print("Test Accuracy:", accuracy_score(y_test, y_test_pred))
print("\nTest Classification Report:\n", classification_report(y_test, y_test_pred))
```

```
Test Accuracy: 0.85

Test Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.84      0.85        50
           1       0.84      0.86      0.85        50

    accuracy                           0.85       100
   macro avg       0.85      0.85      0.85       100
weighted avg       0.85      0.85      0.85       100
```

Name : Advik Hegde **Div : D15C** **Roll No. : 15**

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)

Answer :

Import Libraries:

- Essential libraries like pandas, numpy, train_test_split, GridSearchCV, StandardScaler, XGBRegressor, and r2_score are imported. These libraries handle data processing, model training, hyperparameter tuning, and performance evaluation.

Load Dataset:

- The dataset is read from the Excel file Real estate valuation data set.xlsx using pd.read_excel().
- The column No (index column) is dropped as it's not relevant for the model.

Separate Features and Target:

- The features (X) are separated by dropping the target column 'Y house price of unit area', which is stored in y.

- X contains the input variables, and y contains the output (target variable).

Standardize Features:

- The features X are standardized using StandardScaler(). This step ensures that all features have a mean of 0 and a standard deviation of 1, which is important for models like gradient boosting.

Train/Test Split:

- The dataset is split into training (70%) and test (30%) sets using train_test_split(). This step is crucial for evaluating the model's performance on unseen data.

Define Model and Hyperparameters:

- An XGBRegressor model is defined, and hyperparameters like learning_rate, max_depth, and n_estimators are set for tuning.
- A GridSearchCV is used to search through different combinations of these hyperparameters to find the best performing model, with cross-validation (5-fold) and r2 scoring.

Train the Model:

- The model is trained using the training data (X_train, y_train) and the GridSearchCV method. The grid search will try different hyperparameter combinations and select the best one based on R^2 score.

Evaluate Performance:

- The best model from grid_search is used to make predictions on the test data (X_test).
- The R^2 score (r2_score()) is computed to measure how well the model explains the variance in the test set.
- The Adjusted R^2 score is calculated to adjust the R^2 score for the number of features used, providing a more reliable performance metric for models with multiple predictors.

Name : Advik Hegde **Div : D15C** **Roll No. : 15**

```python
# 1. Import libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import r2_score
from sklearn.preprocessing import StandardScaler
from xgboost import XGBRegressor

# 2. Load the dataset
data = pd.read_excel('Real estate valuation data set.xlsx')
data = data.drop(columns=['No'])  # Remove index column if present

# 3. Separate features and target
X = data.drop(columns=['Y house price of unit area'])
y = data['Y house price of unit area']

# 4. Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 5. Train/test split (70/30)
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.3, random_state=42
)
```

```python
# 6. Define model and hyperparameters
param_grid = {
    'learning_rate': [0.05, 0.1, 0.2],
    'max_depth': [3, 4, 5],
    'n_estimators': [100, 150, 200]
}

grid_search = GridSearchCV(
    XGBRegressor(objective='reg:squarederror', random_state=42),
    param_grid,
    scoring='r2',
    cv=5,
    verbose=1,
    n_jobs=-1
)

# 7. Train the model
grid_search.fit(X_train, y_train)

# 8. Evaluate performance
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

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))

print("R² Score:", round(r2, 4))
print("Adjusted R² Score:", round(adjusted_r2, 4))
print("Best Parameters:", grid_search.best_params_)
```

```
Fitting 5 folds for each of 27 candidates, totalling 135 fits
R² Score: 0.7258
Adjusted R² Score: 0.7119
Best Parameters: {'learning_rate': 0.05, 'max_depth': 3, 'n_estimators': 100}
```

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).

Answer :

Key Features of the Wine Quality Dataset

The **Wine Quality Dataset** includes several key features that influence the overall quality of the wine. These features are:

- **Fixed Acidity**: Affects the wine's freshness and preservation.
- **Volatile Acidity**: Contributes to a sharp, vinegar-like taste.
- **Citric Acid**: Enhances the freshness and balance of the wine.
- **Residual Sugar**: Determines the sweetness level and balance.
- **Chlorides**: Influences the salty taste and flavor balance.
- **Free Sulfur Dioxide**: Preserves the wine and affects stability.
- **Total Sulfur Dioxide**: Affects preservation and taste.
- **Density**: Indicates alcohol content and the mouthfeel of the wine.
- **pH**: Impacts the wine's stability and aging potential.
- **Sulphates**: Affects preservation and aroma.
- **Alcohol**: Contributes to the body and flavor complexity of the wine.
- **Quality (Target)**: A score from 0 to 10 representing the wine’s overall quality.

Importance of Features in Predicting Wine Quality

- **Acidity-related features** like **Fixed Acidity**, **Volatile Acidity**, and **pH** are crucial for determining the balance and freshness of wine, which are key indicators of quality.

- **Residual Sugar** and **Alcohol** significantly impact sweetness, body, and the perceived quality of the wine.

- **Sulfur Dioxide**, **Chlorides**, and **Sulphates** play an important role in wine preservation and stability.

- **Density** and **Citric Acid** provide information about the mouthfeel, aroma, and overall balance, which also contribute to quality.

Handling Missing Data

During the feature engineering process, **missing data** in the Wine Quality Dataset can be handled using the following techniques:

- **Mean/Median Imputation**:
    - **Advantages**: Simple and quick, good for numerical features with small amounts of missing data.
    - **Disadvantages**: Can introduce bias, especially if data isn't missing randomly, and reduces variance.
- **KNN Imputation**:
    - **Advantages**: Utilizes the similarity between data points, providing more accurate imputation.
    - **Disadvantages**: Computationally expensive, especially for large datasets.
- **Regression Imputation**:
    - **Advantages**: Predicts missing values based on relationships between features, more accurate than mean/median imputation.
    - **Disadvantages**: Requires building a predictive model, which can lead to overfitting.
- **Model-based Imputation** (e.g., using Random Forest):
    - **Advantages**: Very accurate as it leverages complex patterns in the data.

- **Disadvantages**: Computationally intensive and requires more resources.

In summary, each feature in the dataset is vital in determining the quality of wine, and the choice of imputation technique depends on the dataset's nature and the desired trade-off between computational efficiency and accuracy.