Date__________
Page__________

Q1 Use the ~~at~~ following dataset for question
82, 66, 70, 59, 90, 78, 76, 95, 99, 84, 88, 76, 82, 81, 91, 64, 79, 76, 85, 90
① Find the Mean
② Find the Median
③ Find the Mode
④ Find the interquartile range

→

① Mean
Sum of all the numbers = 1611

$$\text{Mean} = \frac{1611}{20} = 80.55$$

② Find the Median
Sort the data → 59, 64, 66, 70, 76, 76, 76, 78, 79, 81, 82, 82, 84, 85, 88, 90, 90, 91, 95, 99

Median Calculations → With 20 values, the avg is 10^{th} and 11^{th} value.
10^{th} value = 81
11^{th} value = 82
$$\text{Median} = \frac{81+82}{2} = 81.5$$

③ Find the Mode
The number of 76 appears 3 times, which is more frequent than any other number.
∴ Mode = 76

Teacher's Sign.: __________

Teacher's Sign.: __________

Teacher's Sign.: __________

Date
Page

④ Find the Interquartile Range (IQR)

a - Lower Half (First 10 values)

59, 64, 66, 70, 76, 76, 76, 78, 79, 81

$Q1 = $ Avg of the 5th and 6th values $= \frac{76+76}{2} = 76$

b - Upper Half (Last 10 values)

82, 82, 84, 85, 88, 90, 90, 91, 95, 99

$Q3 = $ Avg of the 5th and 6th values $= \frac{88+90}{2} = 89$

c - IQR Calculations

$IQR = Q3 - Q1 = 89 - 76 = 13$

Q2 ① Machine learning for Kids ② Teachable Machine

ⓐ For each tool listed above

i - Identify the target audience

ii - Discuss the use of this tool by the target audience

iii - Identify the tools benefits and drawbacks

ⓐ Machine learning for Kids

i - Target audience

Primarily designed for K-12 students, educators and beginner coders

ii - Use by target audience

It allows young learners and teachers to create simple machine learning projects (eg. classifying text or images) using an intuitive, block-based interface.

iii - Drawbacks

a - Limited complexity

b - Oversimplification

c - Scalability.

iv - Benefits

a - Simplifies machine learning

b - Encourages creativity

c - Free and browser based

Teacher's Sign.:

Teachable Machine

i - Target audience

Aimed at educators, hobbyists, creative professionals, and non-technical users interested in quickly prototyping ML Models.

ii - Use by Target Audience

It enables users to train simple machine learning models using images, sounds, or poses by simply uploading examples or using a webcam.

iii - Benefits

a - Ease of use

b - Rapid prototyping

c - Visual and interactive

iv - Drawbacks

a - Limited customisations

b - Simplicity - Not ideal for complex

c - Dependancy on internet.

(b) From the two choices listed below, how would you describe each tool listed above? Why did you choose the answer?

i - Predictive analytic

ii - Descriptive analytic

Both Tools are best described as predictive analytic tools

① - They enable users to train models that can predict outcomes (such as classifying images or sounds) based on provided input data.

② - The focus is on learning patterns from labeled eg (i.e supervised learning) and then using these patterns to make predictions on new, unseen data.

Date______ Page______

Teacher's Sign.: ______

Date__________
Page__________

(c) From the three choices listed below, how would you describe each tool listed above? Why did you choose the answer?

i - Supervised learning
ii - Unsupervised learning
iii - Reinforcement learning

-> Both tools are based on supervised learnings.

① Supervised learning involves training a model on a dataset that includes both the inputs and the desired outputs (labels)

② In Machine learning for kids, users provide labeled examples (eg- "this is a cat" vs "this is not a cat" to train the model.

③ Similarly, Teachable Machine requires users to label examples so the model learns to differentiate between them.

④ Neither tool is set up for unsupervised (finding hidden patterns without labels) or reinforcement learning.

Q3 Data visualisation: Read the two short articles

① - What's in a chart? Step to step guide to identify misinfo in data visualisation

② How bad Covid-19 data visualisations mislead the public

Research a current event which highlights the results of misinformations based on data visualisation

Explain how the data visualisation method failed in presenting accurate information.

Teacher's Sign.: __________

→ Misleading Inflation Charts

ⓐ Context and current events

In early 2023, several prominent news outlets faced critisism for the way they visualised ~~it~~ inflation data.

ⓑ How the visualization method failed.

i - Truncated Y-Axis

By not beginning the Y-axis at zero, the charts exaggerated small fluctuations while understanding the real severnity of rising inflation.

~~ii~~ This "compression" of vertical scale can ~~b~~ make sharp increases appear less dramatic, leading viewers to underestimate the economic impact.

ii - Misleading visual impact

Such distortions misrepresent the true magnitude of change in inflation, thereby influencing public opinion and potentially policy debates

Viewers may be misled into thinking that the economic situation is more stable than it really is.

ⓒ Source citation

For this example, see the Reuters article (Reuters, May 2023) discussing how misleading design choices in inflation charts contributed to public confusion about the actual inflation rates.

Date________
Page________

Teacher's Sign.: ____________

AIDS-I Assignment No: 2

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]

Ans:

- Split the temporary set into validation (20% total) and test (10% total)
- Since X_temp is 30% of the data: validation = (20/30) and test = (10/30) of X_temp

```
[6] # Assume the class label is the last column
    X = df.iloc[:, :-1]
    y = df.iloc[:, -1]

    # Split data: First into training (70%) and a temporary set (30%) for validation and test
    X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.30, random_state=42, stratify=y)

    # Split the temporary set into validation (20% total) and test (10% total)
    # Since X_temp is 30% of the data: validation = (20/30) and test = (10/30) of X_temp
    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)

    print("Training set shape:", X_train.shape)
    print("Validation set shape:", X_val.shape)
    print("Test set shape:", X_test.shape)
```

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

- Data Pre-processing: Feature Scaling
- Handle class imbalance using SMOTE on the training data

```
[7] # Data Pre-processing: Feature Scaling
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_val_scaled = scaler.transform(X_val)
    X_test_scaled = scaler.transform(X_test)

    print("Mean of scaled training data (approx.):", X_train_scaled.mean(axis=0))
```

```
Mean of scaled training data (approx.): [ 2.31554895e-17  3.30792708e-17 -2.18323187e-16  1.42240864e-16
  8.93140310e-17  3.63871978e-16  1.19085375e-16 -2.11707333e-16]
```

```
[8] # Handle class imbalance using SMOTE on the training data
    sm = SMOTE(random_state=42)
    X_train_res, y_train_res = sm.fit_resample(X_train_scaled, y_train)

    print("Resampled training set class distribution:")
    print(pd.Series(y_train_res).value_counts())
```

```
Resampled training set class distribution:
Outcome
1    350
0    350
Name: count, dtype: int64
```

- Hyperparameter tuning using GridSearchCV with an SVM classifier

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

svc = SVC(random_state=42)

grid_search = GridSearchCV(estimator=svc,
                           param_grid=param_grid,
                           cv=5,
                           scoring='accuracy',
                           n_jobs=-1,
                           verbose=1)

grid_search.fit(X_train_res, y_train_res)

print("Best Hyperparameters:", grid_search.best_params_)
print("Best cross-validation accuracy:", grid_search.best_score_)
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
Best Hyperparameters: {'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}
Best cross-validation accuracy: 0.8071428571428572
```

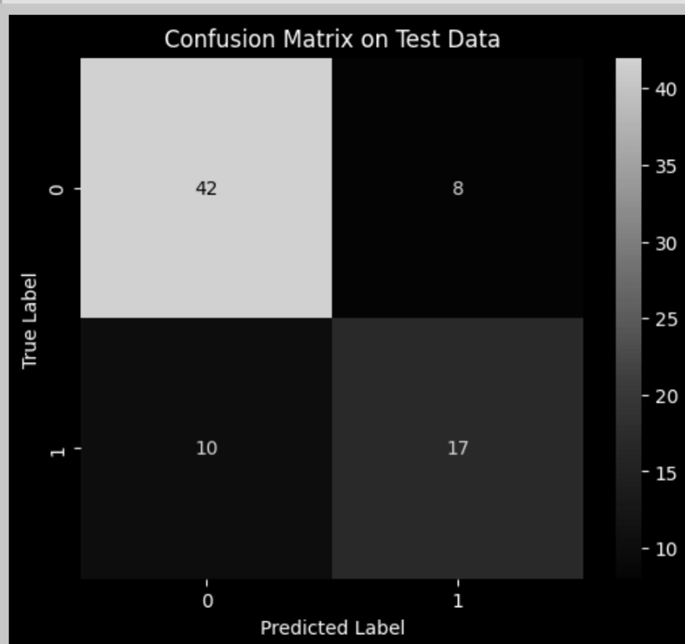
- Evaluate the tuned model on the validation and test sets

```
# Evaluate the tuned model on the validation and test sets
best_svc = grid_search.best_estimator_

# Predictions on the validation set
y_val_pred = best_svc.predict(X_val_scaled)

print("Validation Classification Report:")
print(classification_report(y_val, y_val_pred))

# Predictions on the test set
y_test_pred = best_svc.predict(X_test_scaled)

print("Test Classification Report:")
print(classification_report(y_test, y_test_pred))

# Overall accuracy scores
print("Validation Accuracy:", accuracy_score(y_val, y_val_pred))
print("Test Accuracy:", accuracy_score(y_test, y_test_pred))
```

```
Validation Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.76      0.78       100
           1       0.60      0.67      0.63        54

    accuracy                           0.73       154
   macro avg       0.70      0.71      0.71       154
weighted avg       0.74      0.73      0.73       154

Test Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.84      0.82        50
           1       0.68      0.63      0.65        27

    accuracy                           0.77        77
   macro avg       0.74      0.73      0.74        77
weighted avg       0.76      0.77      0.76        77

Validation Accuracy: 0.7272727272727273
Test Accuracy: 0.7662337662337663
```

- Visualization: Plot confusion matrix for the test set

```
# Visualization: Plot confusion matrix for the test set
cm = confusion_matrix(y_test, y_test_pred)

plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix on Test Data')
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

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

Ans

- Builds a pipeline with polynomial features and Ridge regression, and tunes hyperparameters using GridSearchCV.

```
def build_and_tune_model(self):
    """Builds a pipeline with polynomial features and Ridge regression, and tunes hyperparameters using GridSearchCV."""
    # Create a pipeline
    pipeline = Pipeline([
        ('scaler', StandardScaler()),
        ('poly', PolynomialFeatures()),
        ('ridge', Ridge())
    ])

    # Hyperparameter grid: tuning polynomial degree and Ridge alpha
    param_grid = {
        'poly__degree': [2, 3, 4, 5],
        'ridge__alpha': [0.1, 1, 10, 100]
    }

    # Grid search with 5-fold CV (note: scoring based on R2 score)
    grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='r2', n_jobs=-1, verbose=1)

    grid_search.fit(self.X_train, self.y_train)

    self.model = grid_search.best_estimator_
    self.best_params_ = grid_search.best_params_

    print("Best hyperparameters:", self.best_params_)
    print("Best cross-validation R2 score:", grid_search.best_score_)
```

- Evaluates the model using the test set and computes R2 and adjusted R2 scores.

```
def evaluate_model(self):
    """Evaluates the model using the test set and computes R2 and adjusted R2 scores."""
    # Predict on test data
    y_pred = self.model.predict(self.X_test)

    # Compute R2 score for each target
    r2_scores = []
    adjusted_r2_scores = []
    n = self.X_test.shape[0]

    # For one independent variable, the number of predictors in the final model is determined by the polynomial degree
    degree = self.best_params_['poly__degree']
    # The number of features created by PolynomialFeatures with one input is: degree + 1
    p = degree

    print("Test Set Evaluation:")
    for i in range(self.y_test.shape[1]):
        r2 = r2_score(self.y_test[:, i], y_pred[:, i])
        # Compute adjusted R2: 1 - (1-R2)*(n-1)/(n-p-1)
        adj_r2 = 1 - (1 - r2) * (n - 1) / (n - p - 1)
        r2_scores.append(r2)
        adjusted_r2_scores.append(adj_r2)
        print(f"Dependent Variable {i+1} -- R2: {r2:.4f}, Adjusted R2: {adj_r2:.4f}")

    # Check if all adjusted R2 scores meet the threshold
    if all(adj >= 0.99 for adj in adjusted_r2_scores):
        print("All adjusted R2 scores are above 0.99")
    else:
        print("Warning: Not all adjusted R2 scores are above 0.99")

    return y_pred, r2_scores, adjusted_r2_scores
```

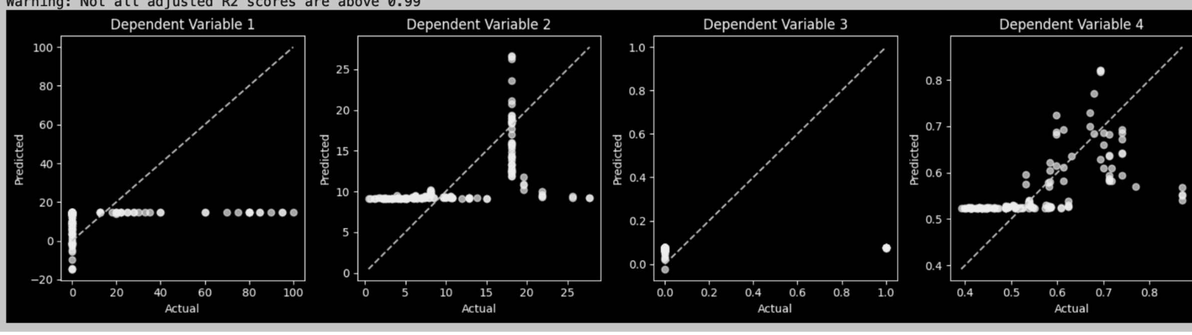
- Visualizes the predictions vs. actual values for each dependent variable.

```
def visualize_results(self, y_pred):
    """Visualizes the predictions vs. actual values for each dependent variable."""
    num_targets = self.y_test.shape[1]

    plt.figure(figsize=(15, 4))
    for i in range(num_targets):
        plt.subplot(1, num_targets, i+1)
        plt.scatter(self.y_test[:, i], y_pred[:, i], alpha=0.7, color='blue')
        plt.plot([self.y_test[:, i].min(), self.y_test[:, i].max()],
                 [self.y_test[:, i].min(), self.y_test[:, i].max()], 'r--')
        plt.xlabel('Actual')
        plt.ylabel('Predicted')
        plt.title(f'Dependent Variable {i+1}')
    plt.tight_layout()
    plt.show()
```

- Final Evaluation Result

```
[3] # Instantiate the BostonHousingRegression class with the dataset path
    bh_reg = BostonHousingRegression('BostonHousing.csv')

    # Run the complete regression analysis
    bh_reg.run()
```

```
Data loaded successfully.
X shape: (506, 1)  y shape: (506, 4)
Training set shape: (354, 1) (354, 4)
Test set shape: (152, 1) (152, 4)
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Best hyperparameters: {'poly__degree': 2, 'ridge__alpha': 10}
Best cross-validation R2 score: 0.1448114369471162
Test Set Evaluation:
Dependent Variable 1 -- R2: 0.0658, Adjusted R2: 0.0533
Dependent Variable 2 -- R2: 0.3148, Adjusted R2: 0.3056
Dependent Variable 3 -- R2: 0.0023, Adjusted R2: -0.0111
Dependent Variable 4 -- R2: 0.3667, Adjusted R2: 0.3582
Warning: Not 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).

Ans:

Key Features and Their Importance

The Wine Quality dataset (available on Kaggle) consists of various physicochemical properties of wine samples. The target is to predict wine quality (rated 0–10) based on these features. Below are the main features and their significance:

- **Fixed Acidity**: Refers to non-volatile acids that contribute to the wine's taste and structure. Plays a role in freshness and sharpness.

- **Volatile Acidity**: High values lead to an undesirable vinegar-like taste. It's a key indicator of wine spoilage.

- **Citric Acid**: Adds flavor and freshness. Higher amounts usually enhance the wine's sensory appeal.

- **Residual Sugar**: Represents the amount of sugar left after fermentation. It affects sweetness and body.

- **Chlorides**: Reflects the salt content. Excessive chlorides can negatively affect taste.

- **Free Sulfur Dioxide**: Used to prevent microbial growth. Its balance is crucial for preservation without affecting flavor.

- **Total Sulfur Dioxide**: Sum of all SO_2 forms. High levels can produce off-odors and suppress aroma.

- **Density**: Correlates with sugar and alcohol content. Affects texture and perception of richness.

- **pH**: Indicates acidity or alkalinity. Impacts wine stability and freshness.

- **Sulphates**: Act as preservatives. Moderate levels enhance flavor and longevity.

- **Alcohol**: A key determinant of wine quality. Higher alcohol levels often correlate with better ratings due to improved mouthfeel and aroma.

Among these, alcohol, volatile acidity, and sulphates are the most influential in predicting wine quality. A good balance of acidity, sugar, and alcohol is essential.

Handling Missing Data in Feature Engineering

Although the wine quality dataset is typically clean, handling missing data is a crucial step in any data preprocessing pipeline.

Common Imputation Techniques:

1. **Mean/Median Imputation**
    - Replace missing numerical values with the column's mean or median.
    - Pros: Simple and fast.
    - Cons: Can distort the data distribution, especially if data is skewed.
2. **Mode Imputation**
    - Best for categorical variables, replacing missing values with the most frequent category.
    - Pros: Maintains category consistency.
    - Cons: Can reduce variance and mask true data diversity.
3. **K-Nearest Neighbors (KNN) Imputation**
    - Estimates missing values based on similar records.
    - Pros: Maintains local data structure.
    - Cons: Computationally expensive and sensitive to irrelevant features.
4. **Multivariate Imputation (e.g., MICE)**
    - Uses regression models to estimate missing values based on other features.
    - Pros: Captures complex relationships between features.
    - Cons: More complex and resource-intensive.