

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)

Ans:

Mean (10 pts):

$$\text{Mean} = \text{Sum} / \text{Count} = 1611 / 20 = 80.55$$

Median (10 pts):

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

$$\text{Median} = (81 + 82) / 2 = 81.5$$

Mode (10 pts):

Mode = Most frequent value = 76

Interquartile Range (20 pts):

$$Q1 = (76 + 76) / 2 = 76$$

$$Q3 = (88 + 90) / 2 = 89$$

$$\text{IQR} = Q3 - Q1 = 13$$

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

Ans:

1 Machine Learning for Kids

2. Target Audience:
 - Primarily designed for school students (ages 8–16), educators, and beginners who are new to machine learning and want a simplified introduction.
3. Use by Target Audience:
 - Students and teachers use the platform to build simple machine learning models (image, text, numbers) using labeled data.
 - It integrates with Scratch and Python, enabling users to apply machine learning models in games or applications.

- The drag-and-drop style helps students understand concepts like classification, training data, and testing.
4. Benefits:
- Beginner-friendly with a visual interface.
 - Good for educational settings to introduce AI and ML concepts.
 - Offers real-world examples like recognizing text, images, or numbers.
 - Integrates with Scratch for interactive projects.
5. Drawbacks:
- Limited to basic ML concepts – not suitable for advanced users.
 - Can be slow with larger datasets.
 - Lacks in-depth customization or control over model parameters.

2 Teachable Machine

- Target Audience:
 - Geared toward students, creators, hobbyists, artists, and educators who want to experiment with machine learning quickly and without code.
- Use by Target Audience:
 - Users create ML models by training with webcam input (images), microphone (sounds), or uploading files (poses, gestures, etc.).
 - Models can be used directly or exported for use in web apps or TensorFlow projects.
- Benefits:
 - No coding skills required.
 - Extremely easy to use – just collect samples and train.
 - Supports exporting to TensorFlow.js for developers to use in custom apps.
 - Great for prototyping and demos.
- Drawbacks:
 - Lacks flexibility for fine-tuning models.
 - Performance is limited by training data size and quality.
 - Focused mainly on classification tasks.

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

Ans:

Machine Learning for Kids – Descriptive Analytic

Reason: It helps users understand patterns in the data through examples like classifying types of text or recognizing objects. It's more focused on explaining and demonstrating concepts rather than forecasting future events.

Teachable Machine – Predictive Analytic

Reason: This tool is oriented towards making predictions, such as classifying new images or audio into one of the trained categories. It uses training data to predict the class of unseen input, fitting the definition of predictive analytics.

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

Ans:

Machine Learning for Kids – Supervised Learning

Reason: It uses labeled data provided by the user (e.g., labeling pictures of cats and dogs). The model is trained on these labeled inputs to learn how to classify new examples.

Teachable Machine – Supervised Learning

Reason: Similar to Machine Learning for Kids, Teachable Machine also uses labeled examples to train the model. The user teaches the machine by providing examples of each class, and the system learns to classify new input based on this.

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.

Ans:

A recent example of misinformation stemming from flawed data visualization involves the misrepresentation of Antarctic sea ice data to downplay climate change concerns.

Misleading Visualization of Antarctic Sea Ice Data

In April 2024, social media posts circulated a comparison of Antarctic sea ice levels on March 9, 1997, and March 9, 2024. These posts claimed that the sea ice extent had remained unchanged over the 27-year period, suggesting that climate change warnings were exaggerated. However, this comparison was misleading for several reasons:

1. **Cherry-Picked Data Points:** The comparison focused solely on two specific dates, ignoring the broader context and variability of sea ice levels over time.
2. **Lack of Long-Term Trend Analysis:** By selecting isolated data points, the visualization failed to represent the overall declining trend in Antarctic sea ice extent observed in recent decades.
3. **Misleading Visual Representation:** The visualization likely omitted essential elements such as error bars or trend lines, which are crucial for accurately conveying data variability and trends. Experts from the National Snow and Ice Data Center (NSIDC) emphasized that such comparisons are not scientifically valid and can mislead the public about the realities of climate change. They highlighted that while sea ice levels can fluctuate annually, the long-term trend shows a significant decline, particularly since 2015.

Lessons Learned

This case underscores the importance of:

- **Contextualizing Data:** Visualizations should present data within an appropriate temporal and spatial context to accurately reflect trends and patterns.
- **Avoiding Selective Representation:** Selecting specific data points without justification can lead to misinterpretation and misinformation.
- **Ensuring Transparency:** Including comprehensive labels, scales, and explanations in visualizations helps prevent misinterpretation. By adhering to these principles, data visualizations can more effectively inform the public and support evidence-based decision-making.

Source: [Reuters Fact Check – Antarctic ice data cherry-picked to make false global warming claim](#)

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

(<https://www.kaggle.com/competitions/data-science-assignments>)

Ans:

Dataset link: <https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database>

Dataset Description: Pima Indians Diabetes

This dataset contains medical diagnostic information of women of Pima Indian heritage, aged 21 and above. The goal is to predict whether a patient has diabetes based on several health-related measurements.


Feature Descriptions

- **Pregnancies:** Number of times the patient has been pregnant
- **Glucose:** Plasma glucose concentration (mg/dL) after 2 hours in an oral glucose tolerance test
- **BloodPressure:** Diastolic blood pressure (mm Hg)
- **SkinThickness:** Skin fold thickness (mm)
- **Insulin:** 2-Hour serum insulin (mu U/ml)
- **BMI:** Body Mass Index (weight in kg / height in m²)
- **DiabetesPedigreeFunction:** A score showing the likelihood of diabetes based on family history
- **Age:** Age of the patient (in years)
- **Outcome:** Target variable — 0 = No diabetes, 1 = Has diabetes



Step 1: Data loading

```
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, ConfusionMatrixDisplay
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] df = pd.read_csv('/content/diabetes.csv')
df.head()
```



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1



Step 2: Data preprocessing:

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

# Handle missing values in features (replace zeros with NaN where invalid)
cols_with_zero = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']
X[cols_with_zero] = X[cols_with_zero].replace(0, np.nan)

# Fill NaNs with mean values
X = X.fillna(X.mean())

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

Step 3: Handle Class Imbalance

```
from imblearn.over_sampling import SMOTE

# Apply SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_scaled, y)
```

We used SMOTE (Synthetic Minority Over-sampling Technique) to balance the dataset. It creates synthetic examples of the minority class by interpolating between existing ones. This helps prevent the model from being biased toward the majority class during training.

Step 4: Train, Validation and Test Split should be 70/20/10, Train and Test split must be randomly done:

```
# First split (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)

# Second split (20% of remaining for validation)
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)

print(f"Train: {X_train.shape}, Validation: {X_val.shape}, Test: {X_test.shape}")
```

Train: (700, 8), Validation: (200, 8), Test: (100, 8)

Step 5: Train SVM with Hyperparameter Tuning

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

grid = GridSearchCV(SVC(), param_grid, refit=True, cv=5, scoring='accuracy')
grid.fit(X_train, y_train)

print("Best Parameters:", grid.best_params_)
print("Validation Accuracy:", grid.score(X_val, y_val))
```

Best Parameters: {'C': 10, 'gamma': 'auto', 'kernel': 'rbf'}
Validation Accuracy: 0.8

In this step, we are training an SVM (Support Vector Machine) model to classify the data. We use GridSearchCV to test different combinations of parameters like C, kernel, and gamma.

It finds the best combination that gives the highest accuracy through cross-validation. This helps us choose the most effective settings for the SVM model automatically.

Step 6: Evaluate on Test Data

```
# Best model from GridSearchCV
best_model = grid.best_estimator_
y_pred = best_model.predict(X_test)

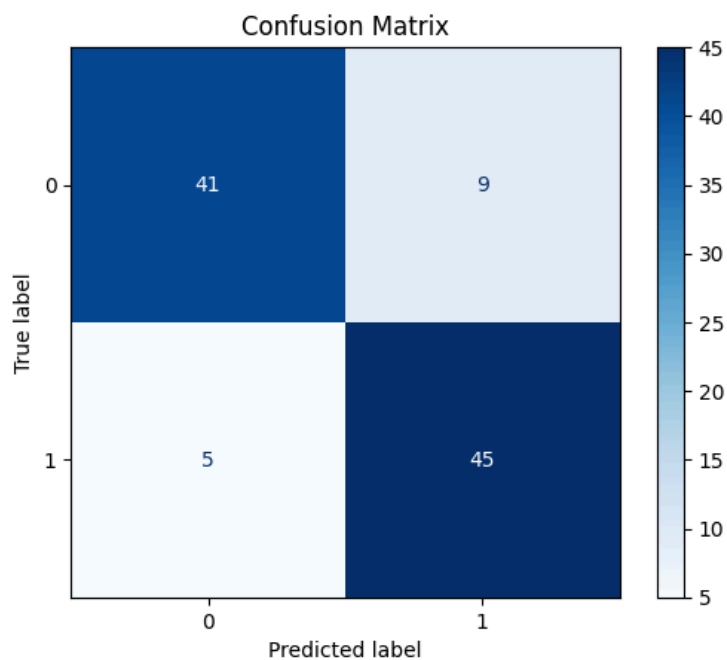
# Accuracy
print("Test Accuracy:", accuracy_score(y_test, y_pred))

# Classification report
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix")
plt.show()
```

🔗 Test Accuracy: 0.86

Classification Report:				
	precision	recall	f1-score	support
0	0.89	0.82	0.85	50
1	0.83	0.90	0.87	50
accuracy			0.86	100
macro avg	0.86	0.86	0.86	100
weighted avg	0.86	0.86	0.86	100



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

Ans:

Objective

To train a regression model using Ridge Regression with Polynomial Feature Expansion to predict real estate prices based on several features (like distance to MRT station, number of convenience stores, etc.). We aim to:

- Tune hyperparameters (alpha and polynomial degree)
- Use a modular, object-oriented design
- Evaluate using R², Adjusted R², and MSE
- Visualize predicted vs actual prices

Step-by-Step Explanation

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

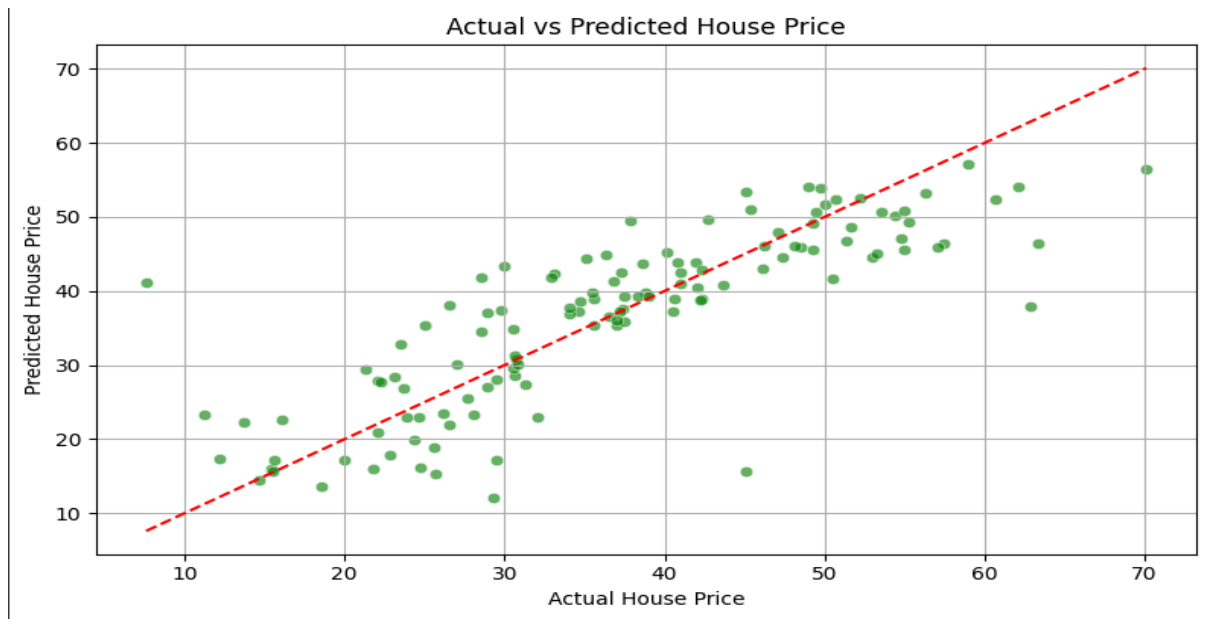
```
best_model = reg_model.train(X_train, y_train)
y_test_actual, y_pred = reg_model.evaluate(best_model, X_test, y_test)
```

- 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).

Ans:

<https://www.kaggle.com/datasets/tawfikelmetwally/wine-dataset>

Key Features & Importance

1. **Alcohol** – Positively correlated with quality; affects flavor and body.
2. **Malic Acid** – Impacts sourness; moderate levels improve taste.
3. **Ash / Alcalinity of Ash** – Reflect mineral content and stability.
4. **Magnesium** – Needed for fermentation; minimal effect on taste.
5. **Total Phenols & Flavanoids** – Strong indicators of quality; affect flavor, astringency.

6. **Nonflavanoid Phenols** – May reduce quality due to bitterness.
7. **Proanthocyanins** – Impact mouthfeel and aging.
8. **Color Intensity / Hue** – Affect visual appeal, perception of richness.
9. **OD280/OD315** – Indicates phenolic content; quality marker.
10. **Proline** – Highly correlated with quality; reflects grape richness.

Handling Missing Data

Although the original dataset typically has no missing data, if present:

Common Imputation Techniques:

Technique	Pros	Cons
Mean/Median	Simple, fast	May distort variance
Mode	Works for categorical-like numeric data	Not ideal for continuous data
KNN	Considers similarity	Slower, sensitive to irrelevant features
Multivariate (e.g., MICE)	Captures feature relations	Slower, assumes linearity
Dropping	Quick when missingness is low	Loses data, may bias model

Use **median** for simple cases, **KNN/MICE** for better accuracy.

Always **fit imputers on training data**, apply to test.

Feature Scaling

Since features are continuous and vary in scale:

from sklearn.preprocessing import StandardScaler

```
scaler = StandardScaler()
```

```
X_scaled = scaler.fit_transform(X)
```

Standardization improves model performance, especially for SVM, KNN, logistic regression, etc.

Summary

- **Important Features:** Alcohol, Flavanoids, Proline, Total Phenols, Color Intensity
- **Best Imputation:** Median for simplicity, KNN/MICE for accuracy
- **Always Standardize:** Ensures equal treatment of features