Assignment - 01

Topic: Implementation of evaluation measures in Machine Learning Approach

- **Accuracy:**

Definition: Accuracy is the proportion of correctly predicted cases out of the total cases. In this context, the accuracy is 48.35% that indicates 48.35% of the predictions were correct.

Formula:

```
accuracy = (TP + TN) / (TP + TN + FP + FN)
```

- **Precision:**

Definition: Precision measures the proportion of true positive predictions among all positive predictions. With a precision of 33.6%, this means 33.6% of the predicted "Malignant" cases were actually malignant.

Formula:

```
precision = TP / (TP + FP)
```

- **Recall (Sensitivity):**

Definition: Recall, also known as sensitivity, is the proportion of true positive predictions out of all actual positive cases. With a recall of 51.03%, this indicates that 51.03% of actual "Malignant" cases were correctly identified.

Formula:

```
recall = TP / (TP + FN)
```

- **F1 Score:**

Definition: The F1 score is the harmonic mean of precision and recall. It balances the two metrics, making it useful in cases of class imbalance. An F1 score of 40.52% indicates a balanced trade-off between precision and recall.

Formula:

```
f1 = 2 * (precision * recall) / (precision + recall)
```

- **Specificity:**

Definition: Specificity measures the proportion of true negative predictions among all actual negative cases. A 46.94% specificity means 46.94% of the actual "Benign" cases were correctly identified.

Formula:

```
specificity = TN / (TN + FP)
```

- **F-beta Score (beta=0.5):**

Definition: The F-beta score with beta set to 0.5 gives more weight to precision than recall. In this case, it emphasizes the accuracy of positive predictions. A score of 36.07% indicates this balance.

Formula:

```
beta_05 = 0.5
fbeta_05 = (1 + beta_05**2) * (precision * recall) / (beta_05**2 *
precision + recall)
```

Assignment - 02

Topic: Implementation of Python Machine Learning Libraries

- **NumPy:**

NumPy is a fundamental library for scientific computing in Python. It provides support for arrays, matrices, and a large collection of mathematical functions to operate on these data structures efficiently.

- **SciPy:**

SciPy is a Python library used for scientific and technical computing. It builds on NumPy by adding modules for optimization, linear algebra, integration, interpolation, eigenvalue problems, and other tasks common in science and engineering.

- **Scikit-learn:**

Scikit-learn is a robust machine learning library for Python. It offers simple and efficient tools for data mining and data analysis, including classification, regression, clustering, and dimensionality reduction, built on NumPy, SciPy, and matplotlib.

- **Theano:**

Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently. It is primarily used for deep learning and offers tight integration with NumPy.

- **TensorFlow:**

TensorFlow is an open-source library for numerical computation and large-scale machine learning. It uses data flow graphs to build models and is highly flexible for building and training various types of machine learning and deep learning models.

- **Keras:**

Keras is a high-level neural networks API written in Python. It is capable of running on top of TensorFlow, Theano, or Microsoft Cognitive Toolkit (CNTK). Keras simplifies the process of building and training deep learning models.

- **PyTorch:**

PyTorch is an open-source deep learning library for Python, primarily developed by Facebook's AI Research lab. It provides tensors and dynamic neural networks with strong GPU acceleration and is known for its flexibility and ease of use.

- **Pandas:**

Pandas is an open-source data analysis and manipulation library for Python. It provides data structures like DataFrames to efficiently handle and analyze structured data, making data cleaning, transformation, and analysis straightforward.

- **Matplotlib:**

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is widely used for plotting graphs and charts, offering various plot types to visualize data in different formats.

Assignment - 03

Topic: Implementation of **Support Vector Machine (SVM)**, **K-Nearest Neighbors (K-NN), and k-Means Clustering** by using Breast Cancer Dataset by UCI Machine Learning Repository

Step-by-Step Code Implementation:

1. Load the dataset
2. Data Cleaning
3. Data Preprocessing
4. Train the SVM model/k-NN model
5. Evaluate the model

Explanation:

- **Load the dataset:**

  Read the dataset using pandas.
- **Data Cleaning:**
  1. Replace missing values ('?') with NaN and drop rows with NaN values.
  2. Convert the 'BareNuc' column to integer type.
- **Data Preprocessing:**
  1. Separate features (X) and target variable (y).
  2. Split the data into training and testing sets.
  3. Standardize the features using StandardScaler.
- **Train the SVM model / K-NN model:**
  1. Use a linear kernel SVM from sklearn.svm.SVC. Or, Use the k-Nearest Neighbors classifier from sklearn.neighbors. KNeighborsClassifier with n_neighbors=5.
  2. Fit the model on the training data.
- **Evaluate the model:**
  1. Predict on the test set.
  2. Calculate accuracy, confusion matrix, and classification report.
  3. Display evaluation metrics and plot the confusion matrix using matplotlib and seaborn.

K-Means Clustering on Breast Cancer Dataset: Results and Analysis

Cluster Statistics:

After performing k-means clustering with `n_clusters=2` on the breast cancer dataset, the cluster statistics are as follows:

- **Cluster for 2**: 230 data points
- **Cluster for 4**: 453 data points

This indicates that the k-means algorithm divided the dataset into two clusters, with 230 samples in the first cluster and 453 samples in the second cluster.

Evaluation Metrics:

To evaluate the performance of the clustering, we mapped the clusters to the original class labels (benign and malignant) based on the majority class in each cluster. The results are:

- **Accuracy**: 95.75%

Confusion Matrix:

Here:

- **True Positives (TP)**: 220 (malignant samples correctly identified)
- **True Negatives (TN)**: 434 (benign samples correctly identified)
- **False Positives (FP)**: 10 (benign samples incorrectly identified as malignant)
- **False Negatives (FN)**: 19 (malignant samples incorrectly identified as benign)

Analysis:

- **High Accuracy**: The model achieves a high accuracy of 95.75%, indicating that it correctly classifies the majority of samples.
- **PCA** is used to reduce the dimensionality of the dataset while preserving most of the variance in the data.
- By reducing the number of dimensions, PCA simplifies the visualization of high-dimensional data and allows for easier interpretation of clustering results.

Assignment - 04

Naive Bayes on Breast Cancer Dataset: Results and Analysis

Steps:

1. Loads and cleans the dataset.
2. Selects the relevant features and splits the data into training and testing sets.
3. Standardizes the features.
4. Applies PCA to reduce the dimensionality to 2 components for visualization.
5. Trains a Naive Bayes classifier on the scaled training data.
6. Predicts the class labels for the test data.
7. Prints the accuracy and confusion matrix.
8. Visualizes the classification results using a scatter plot where the points are colored based on their predicted class.

Output:

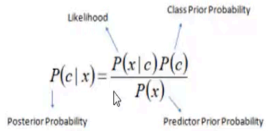
Accuracy: Accuracy is the percentage of correctly predicted instances among all instances.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Visualization:

- **Purpose:** The scatter plot visualizes the classification results, showing how the data points are distributed in the 2D PCA-transformed space, with colors representing the predicted class labels.
- **PCA:** Principal Component Analysis (PCA) was used to reduce the dimensionality of the dataset to 2 components, allowing for an effective 2D visualization.
- **Plot Description:**
    - Each point represents a data instance.
    - Points are colored based on their predicted class label (using a color map, e.g., 'coolwarm').
    - The plot helps in understanding how well the Naive Bayes classifier has separated the two classes in the reduced 2D space.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:



$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

Above,

- $P(c|x)$ is the posterior probability of *class* (c, *target*) given *predictor* (x, *attributes*).
- $P(c)$ is the prior probability of *class*.
- $P(x|c)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(x)$ is the prior probability of *predictor*.

	A	B	C	D	E
1	Days	Outlook	Temperature	Routine	Wear Coat?
2	D1	Sunny	Cold	Indoor	No
3	D2	Sunny	Warm	Outdoor	No
4	D3	Cloudy	Warm	Indoor	No
5	D4	Sunny	Warm	Indoor	No
6	D5	Cloudy	Cold	Indoor	Yes
7	D6	Cloudy	Cold	Outdoor	Yes
8	D7	Sunny	Cold	Outdoor	Yes

Predict (Cloudy, Warm, Outdoor) ___ ?

Predict (cloudy, warm, outdoor)

* P(c = Yes | x = cloudy, warm, outdoor)
* P(c = No | x = cloudy, warm, outdoor)

Step 1: $P(c=Yes|x) = \frac{P(x|c=Yes) * P(c=Yes)}{P(x)}$

$= \frac{P(cloudy|c=Yes) * P(warm|c=Yes) * P(outdoor|c=Yes) * P(c=Yes)}{P(cloudy) \cdot P(warm) \cdot P(outdoor)}$

$= \frac{\frac{2}{3} * \frac{0}{3} * \frac{2}{3} * \frac{3}{7}}{\frac{3}{7} * \frac{3}{7} * \frac{3}{7}}$

$= 0$

Step 2:

$P(c=No|x) = \frac{P(x|c=No) * P(c=No)}{P(x)}$

$= \frac{P(cloudy|c=No) * P(warm|c=No) * P(outdoor|c=No) * P(c=No)}{P(cloudy) * P(warm) * P(outdoor)}$

$= \frac{\frac{1}{4} * \frac{3}{4} * \frac{1}{4} * \frac{4}{7}}{\frac{3}{7} * \frac{3}{7} * \frac{3}{7}}$

$= \frac{49}{144}$

$= .34$

$\therefore P(c=No|x) > P(c=Yes|x)$

So, Predicted output for (cloudy, warm, outdoor) is "No".

Assignment - 05

Decision Tree:

Steps to follow:

1. Choose a target Attribute.
2. Calculate Information Gain (I.G) of that Target Attribute.
3. Calculate Entropy of other attributes using the following formula.

$$E(A) = \sum_{i=1}^{N} \frac{P_i + N_i}{P+N} \times I(P_i N_i)$$

4. Subtract E(A) from I.G. of each attribute for find out the Gain.

$$I.G = \frac{-P}{P+N} \log_2\left(\frac{P}{P+N}\right) - \frac{N}{P+N} \log_2\left(\frac{N}{P+N}\right)$$

Gain = I.G. − E(A)

Decision Tree:

Weather	Humidity	Wind	Play
Sunny	High	Weak	No
Sunny	Normal	Weak	No
Sunny	Normal	Strong	No
Cloudy	High	Weak	No
Cloudy	High	Strong	No
Cloudy	Normal	Strong	Yes
Cloudy	Normal	Weak	Yes
Rainy	High	Weak	Yes
Rainy	Normal	Strong	Yes
Rainy	Normal	Weak	Yes

• Information Gain for Play: .

$$I.G. = -\frac{P}{P+N}\log_2\frac{P}{P+N} - \frac{N}{P+N}\log_2\frac{N}{P+N}$$

$$= -\frac{5}{5+5}\log_2\left(\frac{5}{5+5}\right) - \frac{5}{5+5}\log_2\left(\frac{5}{5+5}\right)$$

$$= -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right)$$

$$= -\frac{1}{2}\log_2(2^{-1}) - \frac{1}{2}\log_2(2^{-1})$$

$$= \frac{1}{2} + \frac{1}{2}$$

$$= 1$$

• For (weather):

	No	Yes
Sunny	3	0
Cloudy	2	2
Rainy	0	3

$$IG(\text{Sunny}) = -\frac{3}{3}\log_2\left(\frac{3}{3}\right) - \frac{0}{3}\log_2\left(\frac{0}{3}\right) = 0 \times \frac{3}{10} = 0$$

$$I.G(\text{Cloudy}) = -\frac{2}{4}\log_2\left(\frac{2}{4}\right) - \left(\frac{2}{4}\right)\log_2\left(\frac{2}{4}\right)$$

$$= -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right) = 1 \times \frac{4}{10} = 0.4$$

$$I.G(\text{Rainy}) = -\frac{0}{3}\log\left(\frac{0}{3}\right) - \frac{3}{3}\log_2\left(\frac{3}{3}\right)$$

$$= 0 \times \frac{3}{10} = 0$$

$$E(\text{weather}) = 0 + 0.4 + 0 = 0.4$$

$\therefore$ (weather) Gain $= 1 - 0.4 = 0.6$