National University of Computer & Emerging Sciences, Karachi

FALL 2024, LAB MANUAL – 06

EVALUATION METRICS

COURSE CODE :	**AL3002**
INSTRUCTOR :	**Usama Bin Umar**

OBJECTIVE
1. Understanding of Evaluation Metrics
2. Different Metrics for Classification and Regression
3. How to split dataset
4. Cross Validation

EVALUATION METRICS

Evaluation metrics in machine learning are used to assess the performance of a model or algorithm by quantifying how well it is doing its intended task. The choice of evaluation metrics depends on the specific problem you are trying to solve and the nature of your data. Here are some commonly used evaluation metrics in machine learning:

- Accuracy
- Precision
- Recall
- F1 Score
- R2 Score
- ROC AUC Curve
- MSE
- Rand_Score

These are just a few of the many evaluation metrics used in machine learning. The choice of metric depends on the problem at hand and the goals of your analysis. Selecting the most appropriate metric based on your specific use case is essential to accurately assess the model's performance.
Evaluation Metrics are different for different tasks. The Evaluating parameters are different for classification and Regression and the same applies for Clustering. Creating an AI based ML or DL model will be beneficial but it would be great to measure the performance of ML or DL models. For measuring the performance of the model different metrics have been used for different purpose
Here are the main Evaluation Metrics used for Classification, Regression, and Clustering.

CLASSIFICATION	REGRESSION	CLUSTERING
Accuracy Score	Mean Absolute Error	Rand Score
F1 Score	Mean Square Error	Adjusted Square Error
Precision	Max Error	Homogeneity Score
Recall	R2 Score	Completeness Score
ROC_AUC Score	Explained Variance Score	Mutual Info Score
Balanced Accuracy Score	D2 Absoulute Score	V Measure Score

CLASSIFICATION METRICS

Before learning different metric for Classification, the best way is to understand the concept of True Positive, True Negative, False Positive and False Negative. In machine learning, especially in the context of classification algorithms, these terms are used to evaluate the performance of a model. They are typically used in a confusion matrix, which is a table used to evaluate the performance of a classification algorithm

True Positive (TP):

The instances that were actually positive and were correctly predicted as positive by the model.
Example: If the model correctly predicts 100 people as having a disease out of 150 people who actually have the disease, then TP = 100.

True Negative (TN):

The instances that were actually negative and were correctly predicted as negative by the model.
Example: If the model correctly predicts 200 people as not having a disease out of 250 people who do not have the disease, then TN = 200.

False Positive (FP):

The instances that were actually negative but were incorrectly predicted as positive by the model. Also known as a Type I error.
Example: If the model incorrectly predicts 50 people as having a disease out of 250 people who do not have the disease, then FP = 50.

False Negative (FN):

The instances that were actually positive but were incorrectly predicted as negative by the model. Also known as a Type II error.
Example: If the model incorrectly predicts 50 people as not having a disease out of 150 people who have the disease, then FN = 50.

These terms are essential in assessing the accuracy, precision, recall, and F1-score of a classification model. Here's how they are calculated:

Accuracy

Accuracy is the most straightforward metric and is used for classification problems. It measures the ratio of correctly predicted instances to the total number of instances. However, accuracy may not be a good choice when dealing with imbalanced datasets.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

This formula in its more expanded version

$$Accuracy = \frac{TrueNegatives + TruePositive}{TruePositive + FalsePositive + TrueNegative + FalseNegative}$$

Model Testing Accuracy

```
teacKNN=accuracy_score(test_y,PredictionDT)
testingAccKNN=teacKNN*100
print(testingAccKNN)
```

Precision

Precision is a metric that measures the proportion of true positive predictions among all positive predictions. It is particularly useful when the cost of false positives is high.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Precision

```
from sklearn.metrics import precision_score
precision_score(y_test,prediction)
```

Recall (Sensitivity or True Positive Rate)

Recall measures the proportion of true positive predictions among all actual positives. It is important when you want to ensure that you catch as many positive cases as possible.

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Recall or Sensitivity

```
from sklearn.metrics import recall_score
recall_score(y_test,prediction)
```

F1-Score

The F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall and is useful when you want to consider both false positives and false negatives.

$$\text{F1 Score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

F1 Score

```
from sklearn.metrics import f1_score
f1_score(y_test,prediction)
```

Specificity (True Negative Rate)

Specificity measures the proportion of true negative predictions among all actual negatives. It is important in situations where false positives are costly.

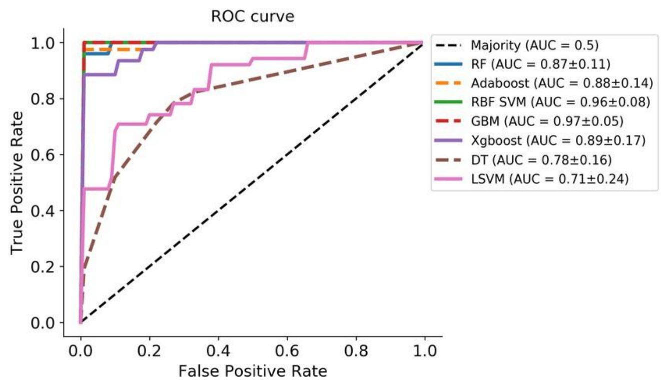
$$\text{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}}$$

Specificity

```
from sklearn.metrics import confusion_matrix
tn, fp, fn, tp = confusion_matrix(y_test,prediction).ravel()
specificity = (tn / (tn + fp))
print (specificity)
```

ROC Curve and AUC

Receiver Operating Characteristic (ROC) curves are used for binary classification problems. They plot the true positive rate against the false positive rate at various thresholds. The Area Under the ROC Curve (AUC) summarizes the overall performance of the model, with a higher AUC indicating better performance.



ROC AUC CURVE

```
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_test,prediction)

# Calculate ROC-AUC score
roc_auc = roc_auc_score(y_test,prediction)

# Plot ROC curve with shaded area under the curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC Curve (AUC = {roc_auc:.2f})')
plt.fill_between(fpr, tpr, color='skyblue', alpha=0.4)
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve with AUC Area')
plt.legend(loc='lower right')
plt.show()
```

CONFUSION MATRIX

A confusion matrix is a table used to evaluate the performance of a classification algorithm. It is a summary of the actual vs. predicted classifications, and it is particularly useful for understanding the behavior of the algorithm when dealing with multiple classes.

CONFUSION MATRIX

```
from sklearn.metrics import confusion_matrix

# Generate confusion matrix
conf_matrix = confusion_matrix(y_test,prediction)

# Display the confusion matrix using a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='g', cmap='Blues')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

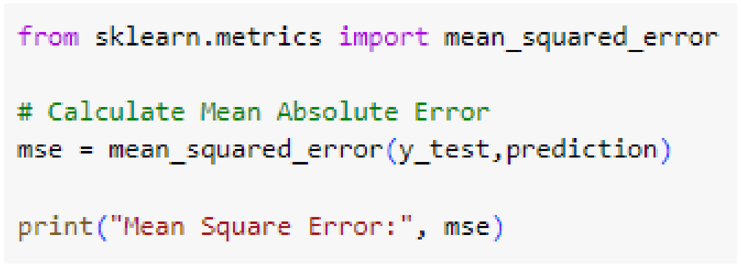
REGRESSION METRICS

Mean Absolute Error (MAE)

MAE is a regression metric that measures the average absolute difference between the predicted and actual values. It is less sensitive to outliers compared to Mean Squared Error (MSE).

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |x_i - x|$$

Mean Absolute Error



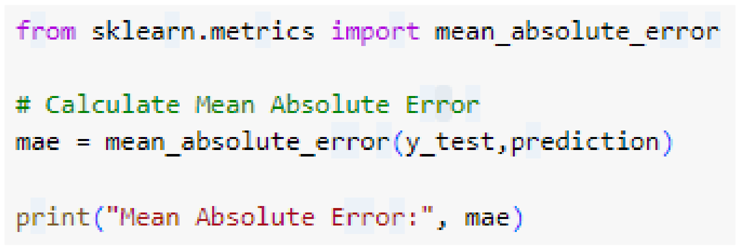
```python
from sklearn.metrics import mean_squared_error

# Calculate Mean Absolute Error
mse = mean_squared_error(y_test,prediction)

print("Mean Square Error:", mse)
```

Mean Squared Error (MSE)

MSE is another regression metric that measures the average of the squared differences between predicted and actual values.

$$\mathrm{MSE} = \frac{1}{n}\sum (Y_i - \hat{Y}_i)^2$$

Mean Square Error

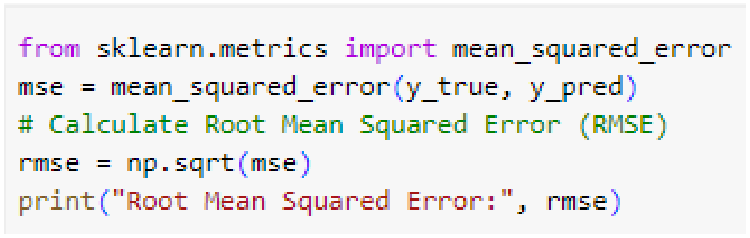


```python
from sklearn.metrics import mean_absolute_error

# Calculate Mean Absolute Error
mae = mean_absolute_error(y_test,prediction)

print("Mean Absolute Error:", mae)
```

Root Mean Squared Error (RMSE)
RMSE is the square root of the MSE. It is often used to provide a more interpretable scale for the error in regression tasks.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \|y(i) - \hat{y}(i)\|^2}{N}},$$

Root Mean Square Error

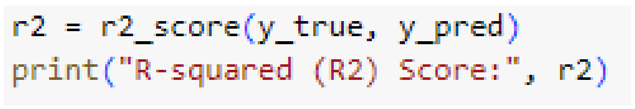


```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_true, y_pred)
# Calculate Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)
print("Root Mean Squared Error:", rmse)
```

R-squared (Coefficient of Determination)
R-squared measures the proportion of the variance in the dependent variable that is predictable from the independent variables. It ranges from 0 to 1, with higher values indicating a better fit of the model to the data.

$$R^2 = \frac{SSR}{SST} = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2}$$

R2 Score



```
r2 = r2_score(y_true, y_pred)
print("R-squared (R2) Score:", r2)
```

HOW TO SPLIT DATASET

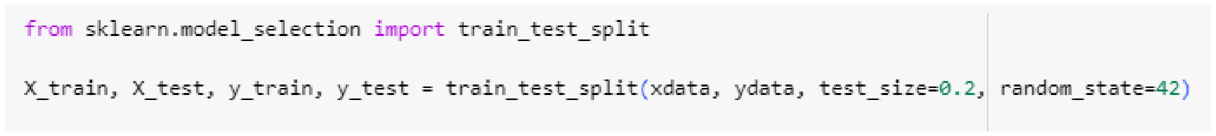
In machine learning, splitting a dataset into training and testing sets is a crucial step for evaluating the performance of a model. There are several methods to split a dataset, each serving different purposes. Here are some common techniques:

- Train Test Split
- Cross Validation
- K Fold Cross Validation
- Time Series Split
- LOOCV

Train Test Split

In the earlier lab, we have already covered how to split dataset using Train Test Split.
The dataset is divided into a training set and a testing set. The training set is used to train the model, and the testing set is used to evaluate its performance.

Train Test Split



```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(xdata, ydata, test_size=0.2, random_state=42)
```

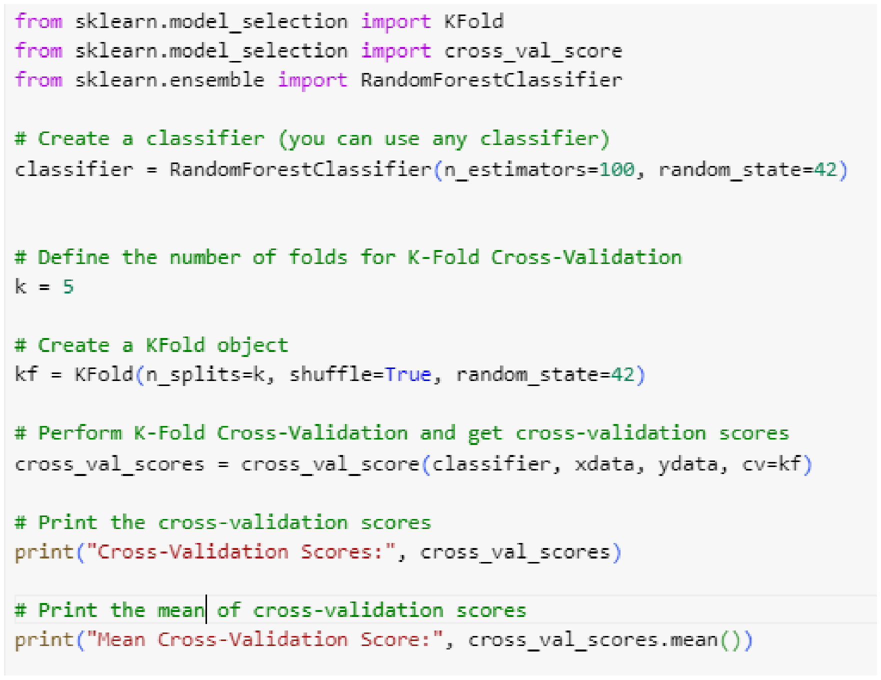
Random State

In scikit-learn, the random_state parameter is used to control the randomness in algorithms that involve randomization. Many machine learning algorithms use randomness or random initializations, and setting the random_state parameter ensures reproducibility. When you set the random_state to a specific integer value, the results will be the same every time you run the code. This is particularly useful when you want to achieve consistent and reproducible results, especially during the development and debugging phase

Cross Validation

Cross-validation is a resampling technique used in machine learning to assess the performance and generalizability of a predictive model. The primary goal of cross-validation is to provide an unbiased estimate of how well a model is expected to perform on an independent dataset.

Cross Validation



```python
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier

# Create a classifier (you can use any classifier)
classifier = RandomForestClassifier(n_estimators=100, random_state=42)


# Define the number of folds for K-Fold Cross-Validation
k = 5

# Create a KFold object
kf = KFold(n_splits=k, shuffle=True, random_state=42)

# Perform K-Fold Cross-Validation and get cross-validation scores
cross_val_scores = cross_val_score(classifier, xdata, ydata, cv=kf)

# Print the cross-validation scores
print("Cross-Validation Scores:", cross_val_scores)

# Print the mean of cross-validation scores
print("Mean Cross-Validation Score:", cross_val_scores.mean())
```

LAB TASKS

Task 1:

You are working for a telecommunications company. The company has collected data on 500 customers, including features like usage duration, monthly charges, and customer feedback. Your task is to build a customer churn prediction model to identify customers who are likely to leave the service.
Click here to download the description and dataset

- Evaluate your model using Confusion Matrix, Calculate its f1 Score and Plot AUC ROC curve.
- Compare the results of Accuracy Score with TP, TR, FP, FN and briefly discuss in the text cell whether accuracy score measure is effective for your dataset or not.
- Plot the confusion Matrix
- Check whether you have done your preprocessing effectively

Task 2:

You are working on a disease prediction model based on patient health records. The dataset contains 500 patients' data, including symptoms, age, and medical history. The task is to predict whether a patient has the disease (1) or not (0).
Click here to download the description and dataset

Using Train-Test Split:

- Split the dataset into a training set (80%) and a testing set (20%).
- Train a random forest and KNN classifier on the training set.
- Calculate and print accuracy, precision, recall, and F1-score on the testing set.
- Using Stratified K-Fold Cross-Validation:

Implement 10-fold stratified cross-validation using a KNN and RF classifier.

- Calculate the average precision across all folds.
- Ensure that the positive class is preserved in each fold.
- Compare the result of mean Validation score with Train Test Split using Accuracy Score.
- Plot ROC AUC Curve for RF with train test split and Cross Validation

Additional Evaluation

- Use balanced accuracy score and top K Accuracy Score from Sklearn.

Task 3:

Imagine you are working for a winery and your task is to build a machine learning model to classify wines into categories based on their quality. The dataset you have contains various features related to the chemical composition of the wines. The target variable, 'quality', with higher values indicating better quality. Your goal is to create a model that can accurately predict the quality of wines based on their features. Click here to download dataset
Features: Various chemical properties of wines (e.g., alcohol content, acidity, etc.).
Target Variable: 'quality' - an integer value indicating wine quality.

Data Exploration and Preprocessing

- **Data Loading**: Load the wine quality dataset into your Notebook Python environment.
- **Data Exploration**: Explore the dataset to understand its structure, features, and the distribution of wine qualities.
- **Data Preprocessing:** Handle any missing or inconsistent data. Normalize or scale the features if necessary.

Model Building and Evaluation

- **Feature Selection:** Choose relevant features that might influence wine quality.
- **Train-Test Split:** Split the dataset using Stratified Cross Validation
- **Model Selection**: Select appropriate classification algorithms for the task (e.g., Random Forest, Random Forest, or KNN).
- **Model Training:** Train the selected models using the training data.
- **Model Evaluation:** Evaluate the models using appropriate metrics (accuracy, precision, recall, F1-score) on the test data. Choose the model with the best performance.

Visualization and Interpretation

Visualization: Create visualizations (e.g., feature importance plots, confusion matrices) to better understand your model's behavior and predictions.
Interpretation: Interpret the model's predictions. For example, which features contribute most to high-quality wines?

Additional Challenges:
Imbalanced Classes: If the dataset has imbalanced classes, explore techniques such as oversampling, under sampling, or using different evaluation metrics to compare with or without handling imbalance dataset.
Classification: If the wine qualities are divided into more than two categories, modify your approach for binary classification.

Task 4:

- Apply Evaluation Metrics to the previously covered labs
- Accuracy Score , Classification Report
- Confusion Matrix
- Precision, Recall and Specificity
- Calculate TP, TR , FP, FN
- F1 Score
- Plot AUC ROC Curve

Remember to document your process, explain your decisions, and provide clear justifications for your chosen methods and models.