



KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)
(Deemed to be University)
SCHOOL OF COMPUTER APPLICATIONS

SPRING SEMESTER 2025-26

Date: 22.01.2026

- 1. Course Code:** BCA 3004 **2. Course title:** Introduction to Data Science (IDS) Lab **Marks = 4**

Assignment 3 (Performance Analysis)

Title of the Assignment:

1. Implement logistic regression using Python/R to perform classification on Iris.csv dataset.
 2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.
 3. Plot the AUROC curve for multi-class model.

Confusion Matrix Evaluation Metrics

Contingency table or Confusion matrix is often used to measure the performance of classifiers. A confusion matrix contains information about actual and predicted classifications done by a classification system. Performance of such systems is commonly evaluated using the data in the matrix.

The following table shows the confusion matrix for a two class classifier.

		predicted	
		n	
actual	P	TP	FN
		FP	TN
		N	

Confusion matrix

Here each row indicates the actual classes recorded in the test data set and the each column indicates the classes as predicted by the classifier.

Numbers on the descending diagonal indicate correct predictions, while the ascending diagonal concerns prediction errors.

Some Important measures derived from confusion matrix are:

- **Number of positive (Pos)** : Total number instances which are labelled as positive in a given dataset.
- **Number of negative (Neg)** : Total number instances which are labelled as negative in a given dataset.
- **Number of True Positive (TP)** : Number of instances which are actually labelled as positive and the predicted class by classifier is also positive.
- **Number of True Negative (TN)** : Number of instances which are actually labelled as negative and the predicted class by classifier is also negative.
- **Number of False Positive (FP)** : Number of instances which are actually labelled as negative and the predicted class by classifier is positive.
- **Number of False Negative (FN)** : Number of instances which are actually labelled as positive and the class predicted by the classifier is negative.

- **Accuracy:** Accuracy is calculated as the number of correctly classified instances divided by total number of instances.

The ideal value of accuracy is 1, and the worst is 0. It is also calculated as the sum of true positive and true negative (TP + TN) divided by the total number of instances.

$$acc = \frac{TP+TN}{TP+FP+TN+FN} = \frac{TP+TN}{Pos+Neg}$$

- **Error Rate:** Error Rate is calculated as the number of incorrectly classified instances divided by total number of instances.

The ideal value of accuracy is 0, and the worst is 1. It is also calculated as the sum of false positive and false negative (FP + FN) divided by the total number of instances.

$$err = \frac{FP+FN}{TP+FP+TN+FN} = \frac{FP+FN}{Pos+Neg} \quad \text{Or}$$

$$err = 1 - acc$$

- **Precision:** It is calculated as the number of correctly classified positive instances divided by the total number of instances which are predicted positive. It is also called confidence value. The ideal value is 1, whereas the worst is 0.

$$precision = \frac{TP}{TP+FP}$$

- **Recall:** .It is calculated as the number of correctly classified positive instances divided by the total number of positive instances. It is also called recall or sensitivity. The ideal value of sensitivity is 1, whereas the worst is 0 .

It is calculated as the number of correctly classified positive instances divided by the total number of positive instances.

$$recall = \frac{TP}{TP+FN}$$

Algorithm:

Step 1: Import the required Libraries.

Step 2: Import the Iris Dataset.

Step 3: Initialize the data frame.

Step 4: Perform Data Preprocessing.

Step 5: Use Logistic regression (Train the Machine) to create and evaluate model.

Step 6: Calculate the required evaluation parameters, define ROC curve and plot it.

Python Code:

Step 1: Import the required Libraries.

Step 2: Import the Iris Dataset.

Step 3: Initialize the data frame.

```
import numpy as np
import pandas as pd
col_names = ['Sepal_Length', 'Sepal_Width', 'Petal_Length', 'Petal_Width', 'Species']
df = pd.read_csv('iris.data', names = col_names)
df
```

Step 4: Perform Data Preprocessing. (If required)

- Convert Categorical to Numerical Values if applicable
- Check for Null Value
- Covariance Matrix to select the most promising features
- Divide the dataset into Independent(X) and Dependent(Y) variables.
- Split the dataset into training and testing datasets
- Scale the Features if necessary.

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
y = encoder.fit_transform(Label)
y
```

Step 5: Use Logistic regression (Train the Machine) to create and evaluate model.

```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import PowerTransformer
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.33,random_state=105)

lr = LogisticRegression()
lr.fit(X_train,y_train)
y_pred = lr.predict(X_test)
print(r2_score(y_test,y_pred))
```

Step 6: Calculate the required evaluation parameters. Define ROC curve for multi-class and plot it.

```
from sklearn.metrics import precision_score,confusion_matrix,
                           accuracy_score,recall_score,roc_auc_score,
                           roc_curve, auc
cm = confusion_matrix(y_test,y_pred)
cm
```

```
import matplotlib.pyplot as plt

def plot_roc_curve(true_y, y_prob):
    """
    plots the roc curve based of the probabilities
    """

    fpr, tpr, thresholds = roc_curve(true_y, y_prob)
    plt.plot(fpr, tpr)
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
```

y_pred

y_test

```

# Binarize the labels for multi-class classification
y_bin = label_binarize(y, classes=[0, 1, 2])
n_classes = 3

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y_bin, test_size=0.5, random_state=42)

# Train a classifier
classifier = OneVsRestClassifier(LogisticRegression())
y_score = classifier.fit(X_train, y_train).decision_function(X_test)

# Compute ROC curve and ROC area for each class
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Plot AUC curves for each class
plt.figure()
colors = ['blue', 'green', 'red']
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=2, label=f'Class {i} (AUC = {roc_auc[i]:.2f})')

plt.plot([0, 1], [0, 1], color='gray', lw=2, 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('AUROC Curve for Iris Dataset')
plt.legend(loc='lower right')
plt.show()

```

Conclusion:

In this way we have done data analysis using logistic regression and evaluate the performance of model.

Viva Questions:

- 1) Consider the binary classification task with two classes positive and negative.

Find out TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall

N = 165	Predicted YES	Predicted NO
Actual YES	TP = 150	FN = 10
Actual NO	FP = 20	TN = 100

- 2) Comment on whether the model is best fit or not based on the calculated values.
- 3) Write python code for the preprocessing mentioned in step 4. and Explain every step in detail.