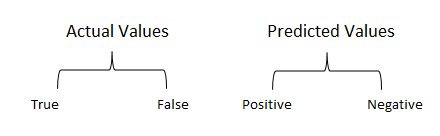
**Experiment No.5**

**Aim:** Data Analytics 2 Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given set.

**Theory:** When we get the data, after data cleaning, pre-processing, and wrangling, the first step we do is to feed it to an outstanding model and get output in probabilities. But how can we measure the effectiveness of our model. Better the effectiveness, better the performance, and that is exactly what we want. And it is where the Confusion matrix comes into the limelight. Confusion Matrix is a performance measurement for machine learning classification.

**What is Confusion Matrix and why you need it?**

Precision, Specificity, Accuracy, and most importantly AUC-ROC curves.



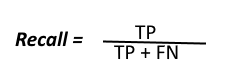
Actual vs Predicted values [Image ]

**How to Calculate Confusion Matrix for a 2-class classification problem?**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Y | Y pred | Output for threshold 0.6 | Recall | Precision | Accuracy |
| 0 | 0.5 | 0 | 1/2 | 2/3 | 4/7 |
| 1 | 0.9 | 1 |
| 0 | 0.7 | 1 |
| 1 | 0.7 | 1 |
| 1 | 0.3 | 0 |
| 0 | 0.4 | 0 |
| 1 | 0.5 | 0 |

Confusion Matrix [Image]

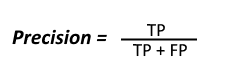
**Recall:**



The above equation can be explained by saying, from all the positive classes, how many we predicted correctly.

Recall should be high as possible.

**Precision:**

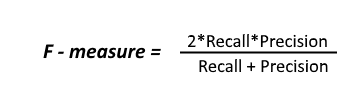


The above equation can be explained by saying, from all the classes we have predicted as positive, how many are actually positive.

Precision should be high as possible.

**Accuracy**

From all the classes (positive and negative), how many of them we have predicted correctly. In this case, it will be 4/7. Accuracy should be high as possible.

**F-measure:**

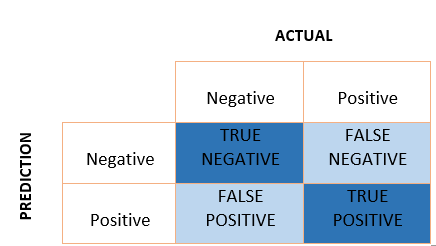
F1 Score [Image]

It is difficult to compare two models with low precision and high recall or vice versa. So to make them comparable, we use F-Score. F-score helps to measure Recall and Precision at the same time. It uses Harmonic Mean in place of Arithmetic Mean by punishing the extreme values more.

Accuracy performance metrics can be decisive when dealing with imbalanced data. The confusion matrix, precision, recall, and F1 score gives better intuition of prediction results as compared to accuracy.

**What is a confusion matrix?**

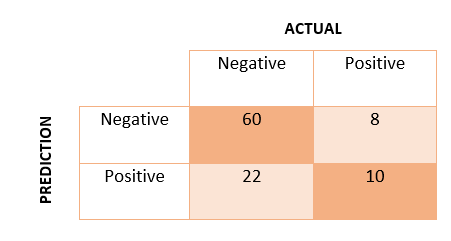
It is a matrix of size 2×2 for binary classification with actual values on one axis and predicted on another.



Confusion Matrix

The confusing terms in the confusion matrix are: **true positive, true negative, false negative and false positive** with an example.

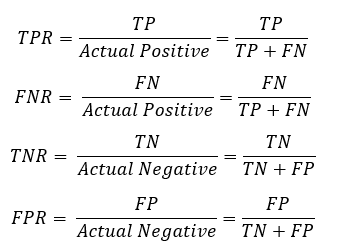
**EXAMPLE:** A machine learning model is trained to predict tumor in patients. The test dataset consists of 100 people.



Confusion Matrix for tumor detection

**True Positive** **(TP)** — model correctly predicts the positive class (prediction and actual both are positive). In the above example, **10 people** who have tumors are predicted positively by the model.  
**True Negative (TN)** — model correctly predicts the negative class (prediction and actual both are negative). In the above example, **60 people** who don’t have tumors are predicted negatively by the model.  
**False Positive (FP)** — model gives the wrong prediction of the negative class (predicted-positive, actual-negative). In the above example, **22 people** are predicted as positive of having a tumor, although they don’t have a tumor. FP is also called a **TYPE I** error.  
**False Negative (FN)** — model wrongly predicts the positive class (predicted-negative, actual-positive). In the above example, **8 people** who have tumors are predicted as negative. FN is also called a **TYPE II** error.

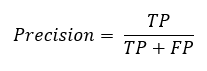
With the help of these four values, we can calculate True Positive Rate (TPR), False Negative Rate (FPR), True Negative Rate (TNR), and False Negative Rate (FNR).



Even if data is imbalanced, we can figure out that our model is working well or not. For that, **the values of TPR and TNR should be high, and FPR and FNR should be as low as possible.** With the help of TP, TN, FN, and FP, other performance metrics can be calculated.

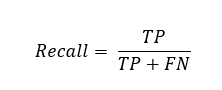
**Precision, Recall:** Both precision and recall are crucial for information retrieval, where positive class mattered the most as compared to negative. Because while searching something on the web, the model does not care about something **irrelevant** and **not retrieved** (this is the true negative case). Therefore only TP, FP, FN are used in Precision and Recall.

**Precision:** Out of all the positive predicted what percentage is truly positive.



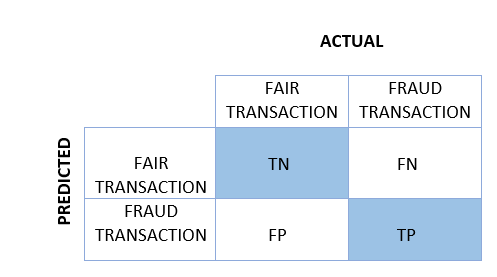
The precision value lies between 0 and 1.

**Recall**: Out of the total positive, what percentage are predicted positive. It is the same as TPR (true positive rate).



**How are precision and recall useful?** .

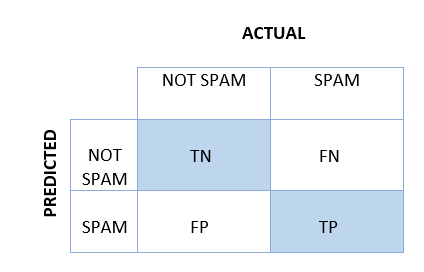
**EXAMPLE 1- Credit card fraud detection**



Confusion Matrix for Credit Card Fraud Detection

We do not want to **miss any fraud transactions**. Therefore, we **want False-Negative to be as low as possible.** In these situations, we can compromise with the low precision, but recall should be high. Similarly, in the medical application, we don’t want to miss any patient. Therefore we focus on having a high recall. So we have discussed when the recall is important than precision. **But, when is the precision more important than recall?**

**EXAMPLE 2 — Spam detection**



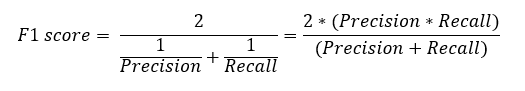
Confusion Matrix for Spam detection

In the detection of spam mail, it is okay if any spam mail remains undetected (false negative), but what if we miss any critical mail because it is classified as spam (false positive). In this situation, **False Positive** should be as low as possible. Here, precision is more vital as compared to recall.

When comparing different models, it will be difficult to decide which is better (high precision and low recall or vice-versa). Therefore, there should be a metric that combines both of these. One such metric is the F1 score.

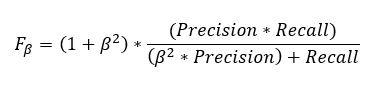
**F1 Score:**

It is the harmonic mean of precision and recall. It takes both false positive and false negatives into account. Therefore, it performs well on an imbalanced dataset.



F1 score gives the same weightage to recall and precision.

There is a **weighted F1 score** in which we can give different weightage to recall and precision. As discussed in the previous section, different problems give different weightage to recall and precision.



**Beta represents how many times recall is more important than precision**. If the recall is twice as important as precision, the value of Beta is 2.

# Confusion Matrix – An Overview with Python and R

# ****Introduction:****

To develop a [machine learning](https://www.mygreatlearning.com/blog/what-is-machine-learning/) classification model, we first collect data, then perform data exploration, data pre-processing, and cleaning. After completing all these processes, we apply the classification technique to achieve predictions from that model. This is a brief idea about how we develop a machine learning model. Before finalizing the classifier model, we have to be sure if it is performing well or not. Confusion Matrix measures the [performance of a classifier](https://www.mygreatlearning.com/blog/model-evaluation-techniques-for-machine-learning-classification-models/) to check efficiency and precision in predicting results.

## ****Confusion Matrix**** ****Definition:****

A confusion matrix is used to judge the performance of a classifier on the test dataset for which we already know the actual values. Confusion matrix is also termed as Error matrix. It consists of a count of correct and incorrect values broken down by each class. It not only tells us the error made by classifier but also tells us what type of error the classifier made. So, we can say that a confusion matrix is a performance measurement technique of a classifier model where output can be two classes or more. It is a table with four different groups of true and predicted values.

## ****Terminologies in Confusion Matrix:****

The confusion matrix shows us how our classifier gets confused while predicting. In a confusion matrix we have four important terms which are:

1. **True Positive (TP)**
2. **True Negative (TN)**
3. **False Positive (FP)**
4. **False Negative (FN)**

We will explain these terms with the help of visualisation of the confusion matrix:

This is what a confusion matrix looks like. This is a case of a 2-class confusion matrix. On one side of the table, there are predicted values and on one side there are the actual values.

Let’s discuss the above terms in detail:

### True Positive (TP)

Both actual and predicted values are Positive.

### True Negative (TN)

Both actual and predicted values are Negative.

### False Positive (FP)

The actual value is negative but we predicted it as positive.

### False Negative (FN)

The actual value is positive but we predicted it as negative.

## ****Performance Metrics:****

Confusion matrix not only used for finding the errors in prediction but is also useful to find some important performance metrics like Accuracy, Recall, Precision, F-measure. We will discuss these terms one by one.

### Accuracy:

As the name suggests, the value of this metric suggests the accuracy of our classifier in predicting results.

It is defined as:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

A 99% accuracy can be good, average, poor or dreadful depending upon the problem.

### Precision

Precision is the measure of all actual positives out of all predicted positive values.

It is defined as:

Precision = TP / (TP + FP)

### Recall

Recall is the measure of positive values that are predicted correctly out of all actual positive values.

It is defined as:

Recall = TP / (TP + FN)

High Value of Recall specifies that the class is correctly known (because of a small number of False Negative).

### F-measure

It is hard to compare classification models which have low precision and high recall or vice versa. So, for comparing the two classifier models we use F-measure. F-score helps to find the metrics of Recall and Precision in the same interval. Harmonic Mean is used instead of Arithmetic Mean.

F-measure is defined as:

F-measure = 2 \* Recall \* Precision / (Recall + Precision)

The F-Measure is always closer to the Precision or Recall, whichever has a smaller value.

## ****Calculation of 2-class confusion matrix****

Let us derive a confusion matrix and interpret the result using simple mathematics.

Let us consider the actual and predicted values of y as given below:

|  |  |  |
| --- | --- | --- |
| **Actual y** | **Y predicted** | **Predicted y with threshold 0.5** |
| 1 | 0.7 | 1 |
| 0 | 0.1 | 0 |
| 0 | 0.6 | 1 |
| 1 | 0.4 | 0 |
| 0 | 0.2 | 0 |

Now, if we make a confusion matrix from this, it would look like:

|  |  |  |
| --- | --- | --- |
| N=5 | **Predicted 1** | **Predicted 0** |
| **Actual: 1** | 1 (TP) | 1 (FN) |
| **Actual: 0** | 1 (FP) | 2 (TN) |

This is our derived confusion matrix. Now we can also see all the four terms used in the above confusion matrix. Now we will find all the above-defined performance metrics from this confusion matrix.

### Accuracy:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

So, Accuracy = (1+2) / (1+2+1+1)

                        = 3/5 which is 60%.

So, the accuracy from the above confusion matrix is 60%.

### Precision

Precision = TP / (TP + FP)

                 = 1 / (1+1)

                 =1 / 2 which is 50%.

So, the precision is 50%.

### Recall

Recall = TP / (TP + FN)

           = 1 / (1+1)

           = ½ which is 50%

So, the Recall is 50%.

### F-measure

F-measure = 2 \* Recall \* Precision / (Recall + Precision)

                    = 2\*0.5\*0.5 / (0.5+0.5)

                    = 0.5

So, the F-measure is 50%.

## ****Confusion Matrix in Python****

In this section, we will derive all performance metrics for a confusion matrix using [**Python**](https://www.mygreatlearning.com/blog/python-tutorial-for-beginners-a-complete-guide/)**.**

import os

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt # Importing the required libraries

import seaborn as sns

%matplotlib inline

os.chdir("C:\\Users\\ABC\\Desktop\\bank")

df=pd.read\_csv("bank.csv", delimiter=";",header='infer')

df.head()

df.columns # Columns in the dataset

df.shape # There are 4521 rows and 17 columns in data

df.info () # Checking info of data

df.dtypes # Checking the data types of variables in data

df.describe() # Summary statistics of numerical columns in data

df.isnull().sum() # Checking the missing value in data. We can see that there is no missing value in data.

df.corr() # Correlation matrix

sns.heatmap(df.corr()) # Visualization of Correlation matrix Using heatmap

**As we see, not a single feature is correlated completely with class, hence requires a combination of features.**

sns.countplot(y='job', data= df)

sns.countplot(x='marital', data= df)

sns.countplot(x='y', data= df)

from sklearn import preprocessing

from sklearn.preprocessing import LabelEncoder

from sklearn import model\_selection

from sklearn.linear\_model import LogisticRegression

from sklearn import metrics

from sklearn.metrics import accuracy\_score,confusion\_matrix

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier

**Sklearn offers a very effective technique for encoding the classes of a categorical variable into numeric format. LabelEncoder encodes classes with values between 0 and n\_classes-1**

le = preprocessing.LabelEncoder()

df.job = le.fit\_transform(df.job)

df.marital = le.fit\_transform(df.marital)

df.default = le.fit\_transform(df.default)

df.education = le.fit\_transform(df.education)

df.housing = le.fit\_transform(df.housing)

df.loan = le.fit\_transform(df.loan)

df.contact = le.fit\_transform(df.contact)

df.month = le.fit\_transform(df.month)

df.poutcome = le.fit\_transform(df.poutcome)

df.y = le.fit\_transform(df.y)

X= df.drop(["y"],axis=1)

y= df ["y"] #### X consists of all independent variables and y has the dependent variable.

print(X.shape,y.shape)

### Train and Test split:

Now, we will split the data into training and testing sets. We will train the model with training data and will test the performance of our model on the test data which will be unknown for the model.

**Here, we split data in train and test in 70:30.**

X\_train, X\_test, y\_train, y\_test = model\_selection.train\_test\_split(X, y, test\_size=0.3, random\_state=42)

Print (X\_train.shape,X\_test.shape, y\_train.shape, y\_test.shape)

model\_log=LogisticRegression (max\_iter=1000, random\_state=42)

model\_log.fit (X\_train, y\_train)

pred=model\_log.predict (X\_test)

accuracy\_score (y\_test, pred)

confusion\_matrix (y\_test, pred)

[[1175 30]

[ 121 31]]

Print (classification\_report (y\_test, prediction\_log))

precision recall f1-score support

0 0.91 0.98 0.94 1205

1 0.51 0.20 0.29 152

accuracy 0.89 1357

macro avg 0.71 0.59 0.62 1357

weighted avg 0.86 0.89 0.87 1357

## ****Confusion Matrix study in R****

Library (dplyr)

Library (ggplot2)

library (DataExplorer)

df=read.csv("adult.csv")

head(df)

summary(df)

colSums (is.na(df)) # Checking if there is any missing value or not column wise

### Changing? into a new category ‘Missing’

df $workclass = ifelse (df $workclass=='?', 'Missing', as.character (df $workclass))

df $workclass = as.factor (df $workclass)

df $occupation = ifelse (df $occupation=='?', 'Missing', as.character(df $occupation))

df $occupation = as.factor (df $occupation)

df $native.country = ifelse(df $native.country== '?', 'Missing',as.character(df $native.country))

df $native.country = as.factor (df $native.country)

summary(df)

str(df)

### Creating a new column target based on income column

df $target=ifelse (df $income == '>50K', 1, 0)

df $target=as.factor (df $target)

For checking outliers:

boxplot (df $capital.gain)

head (sort (df $capital.gain, decreasing = T),10)

boxplot (df $capital.loss)

boxplot (df $hours.per.week)

Changing Age column into 3 categories:

df $age=ifelse (df $age <= 30, 'Young', ifelse (df $age>30 & df $age <= 50, 'Mid-Age', 'Old'))

df $age=as.factor (df $age)

summary (df$age)

# Remove column income

df =select (df, -income)

**Splitting data into test and train:**

set.seed (1000)

index=sample (nrow (df), 0.70\*nrow (df), replace=F)

train= df [index,]

test= df [-index,]

table(train$target)/22792

table(test$target)/9769

#### Applying logistic regression:

mod=glm(target~.,data=train,family='binomial')

summary(mod)

step (mod,direction = 'both')

#### 2nd Iteration based on function call given by step function:

mod1=glm (formula = target ~ age + workclass + fnlwgt + education +

marital.status + occupation + relationship + race + sex +

capital.gain + capital.loss + hours.per.week + native.country,

family = "binomial", data = train)

summary(mod1)

#### Changing significant categorical var levels into dummies:

train$age\_Young\_d = ifelse (train$age== 'Young', 1, 0)

test$age\_Young\_d = ifelse (test$age== 'Young', 1, 0)

train$workclassLocalgov\_d = ifelse (train$workclass== 'Local-gov', 1, 0)

test$workclassLocalgov\_d = ifelse (test$workclass== 'Local-gov', 1, 0)

train$workclassMissing\_d = ifelse (train$workclass== 'Missing', 1, 0)

test$workclassMissing\_d = ifelse (test$workclass== 'Missing', 1, 0)

test$workclassPrivate\_d = ifelse (test$workclass== 'Private', 1, 0)

train$workclassPrivate\_d = ifelse (train$workclass== 'Private', 1, 0)

train$workclassSelfempnotinc\_d = ifelse (train$workclass== 'Self-emp-not-inc', 1, 0)

test$workclassSelfempnotinc\_d = ifelse (test$workclass== 'Self-emp-not-inc', 1, 0)

test$workclassSelfempinc\_d = ifelse (test$workclass== 'Self-emp-inc', 1, 0)

train$workclassSelfempinc\_d = ifelse (train$workclass== 'Self-emp-inc', 1, 0)

train$workclassStategov\_d = ifelse (train$workclass== 'State-gov', 1, 0)

test$workclassStategov\_d = ifelse (test$workclass== 'State-gov', 1, 0)

train$education1st\_4th\_d = ifelse (train$education== '1st-4th', 1, 0)

test$education1st\_4th\_d = ifelse (test$education== '1st-4th', 1, 0)

train$educationAssocacdm\_d = ifelse (train$education== 'Assoc-acdm', 1, 0)

test$educationAssocacdm\_d = ifelse (test$education== 'Assoc-acdm', 1, 0)

train$educationAssocvoc\_d = ifelse (train$education== 'Assoc-voc', 1, 0)

test$educationAssocvoc\_d = ifelse (test$education== 'Assoc-voc',1, 0)

train$educationBachelors\_d = ifelse (train$education== 'Bachelors', 1, 0)

test$educationBachelors\_d = ifelse (test$education== 'Bachelors', 1, 0)

train$educationDoctorate\_d = ifelse (train$education== 'Doctorate', 1, 0)

test$educationDoctorate\_d = ifelse (test$education== 'Doctorate', 1, 0)

train$educationHSgrad\_d = ifelse (train$education== 'HS-grad', 1, 0)

test$educationHSgrad\_d = ifelse (test$education== 'HS-grad', 1, 0)

train$educationMasters\_d = ifelse (train$education== 'Masters', 1, 0)

test$educationMasters\_d = ifelse (test$education=='Masters', 1, 0)

train$educationProfschool\_d = ifelse (train$education== 'Prof-school', 1, 0)

test$educationProfschool\_d = ifelse (test$education== 'Prof-school', 1, 0)

train$educationSomecollege\_d = ifelse (train$education== 'Some-college', 1, 0)

test$educationSomecollege\_d = ifelse (test$education== 'Some-college', 1, 0)

train$marital.statusMarriedAFspouse\_d = ifelse (train$marital.status== 'Married-AF-spouse',1,0)

test$marital.statusMarriedAFspouse\_d = ifelse (test$marital.status== 'Married-AF-spouse',1,0)

train$marital.statusMarriedcivspouse\_d = ifelse (train$marital.status== 'Married-civ-spouse',1,0)

test$marital.statusMarriedcivspouse\_d = ifelse (test$marital.status== 'Married-civ-spouse',1,0)

train$marital.statusNevermarried\_d = ifelse (train$marital.status== 'Never-married', 1, 0)

test$marital.statusNevermarried\_d = ifelse (test$marital.status== 'Never-married', 1, 0)

train$marital.statusWidowed\_d = ifelse (train$marital.status== 'Widowed', 1, 0)

test$marital.statusWidowed\_d = ifelse (test$marital.status== 'Widowed', 1, 0)

train$occupationExecmanagerial\_d = ifelse (train$occupation== 'Exec-managerial', 1, 0)

test$occupationExecmanagerial\_d = ifelse (test$occupation== 'Exec-managerial', 1,0)

train$occupationFarmingfishing\_d = ifelse (train$occupation== 'Farming-fishing', 1, 0)

test$occupationFarmingfishing\_d = ifelse (test$occupation== 'Farming-fishing', 1, 0)

train$occupationHandlerscleaners\_d = ifelse (train$occupation== 'Handlers-cleaners', 1, 0)

test$occupationHandlerscleaners\_d = ifelse (test$occupation== 'Handlers-cleaners', 1, 0)

train$occupationMachineopinspct\_d = ifelse (train$occupation== 'Machine-op-inspct', 1, 0)

test$occupationMachineopinspct\_d = ifelse (test$occupation== 'Machine-op-inspct', 1, 0)

train$occupationOtherservice\_d = ifelse (train$occupation== 'Other-service', 1, 0)

test$occupationOtherservice\_d = ifelse (test$occupation== 'Other-service', 1, 0)

train$occupationProfspecialty\_d = ifelse (train$occupation== 'Prof-specialty', 1, 0)

test$occupationProfspecialty\_d = ifelse (test$occupation== 'Prof-specialty', 1, 0)

train$occupationProtectiveserv\_d = ifelse (train$occupation== 'Protective-serv', 1, 0)

test$occupationProtectiveserv\_d = ifelse (test$occupation== 'Protective-serv', 1, 0)

train$occupationSales\_d = ifelse (train$occupation== 'Sales', 1, 0)

test$occupationSales\_d = ifelse (test$occupation== 'Sales', 1, 0)

train$occupationTechsupport\_d = ifelse (train$occupation== 'Tech-support', 1, 0)

test$occupationTechsupport\_d = ifelse (test$occupation== 'Tech-support', 1, 0)

train$relationshipOwnchild\_d = ifelse (train$relationship== 'Own-child', 1, 0)

test$relationshipOwnchild\_d = ifelse (test$relationship== 'Own-child', 1, 0)

train$relationshipWife\_d = ifelse (train$relationship== 'Wife', 1, 0)

test$relationshipWife\_d = ifelse (test$relationship== 'Wife', 1, 0)

train$raceAsianPacIslander\_d=ifelse (train$race=='Asian-Pac-Islander', 1, 0)

test$raceAsianPacIslander\_d=ifelse (test$race=='Asian-Pac-Islander', 1, 0)

train$raceWhite\_d=ifelse (train$race== 'White', 1, 0)

test$raceWhite\_d=ifelse (test$race=='White',1, 0)

train$native. countryColumbia\_d=ifelse(train$native.country=='Columbia',1,0)

test$native. countryColumbia\_d=ifelse(test$native.country=='Columbia',1,0)

train$native. countrySouth\_d=ifelse(train$native.country=='South',1,0)

test$native. countrySouth\_d=ifelse(test$native.country=='South',1,0)

#### 3rd iteration by using significant dummy vars:

mod2=glm (formula=target~age\_Young\_d+workclassLocalgov\_d+workclassMissing\_d+workclassPrivate\_d+

workclassSelfempinc\_d+workclassSelfempnotinc\_d+workclassStategov\_d+fnlwgt+

education1st\_4th\_d+educationAssocacdm\_d+educationAssocvoc\_d+educationBachelors\_d+educationDoctorate\_d+

educationHSgrad\_d+educationMasters\_d+educationProfschool\_d+educationSomecollege\_d+marital. statusWidowed\_d+

marital. statusMarriedAFspouse\_d+marital. statusNevermarried\_d+marital.statusMarriedcivspouse\_d+

occupationExecmanagerial\_d+occupationFarmingfishing\_d+occupationHandlerscleaners\_d+occupationMachineopinspct\_d+

occupationOtherservice\_d+occupationProfspecialty\_d+occupationProtectiveserv\_d+occupationSales\_d+

occupationTechsupport\_d+relationshipWife\_d+relationshipOwnchild\_d+raceWhite\_d+raceAsianPacIslander\_d+

sex+capital. gain+capital. loss+hours.per. week+native.countryColumbia\_d+native.countrySouth\_d,

data=train, family='binomial')

summary(mod2)

#### Again, getting some insignificant vars. So, to remove those:

mod3=glm (formula=target~age\_Young\_d+workclassLocalgov\_d+workclassMissing\_d+workclassPrivate\_d+workclassSelfempinc\_d+workclassSelfempnotinc\_d+workclassStategov\_d+fnlwgt+education1st\_4th\_d+educationAssocacdm\_d+educationAssocvoc\_d+educationBachelors\_d+educationDoctorate\_d+educationHSgrad\_d+educationMasters\_d+educationProfschool\_d+educationSomecollege\_d+marital. statusWidowed\_d+ marital. statusMarriedAFspouse\_d+marital. statusNevermarried\_d+marital.statusMarriedcivspouse\_d+occupationExecmanagerial\_d+occupationFarmingfishing\_d+occupationHandlerscleaners\_d+occupationMachineopinspct\_d+occupationOtherservice\_d+occupationProfspecialty\_d+occupationProtectiveserv\_d+occupationSales\_d+occupationTechsupport\_d+relationshipWife\_d+relationshipOwnchild\_d+raceWhite\_d+sex+capital.gain+capital.loss+hours.per.week+native.countryColumbia\_d+native.countrySouth\_d, data=train, family='binomial')

summary(mod3)

**# checking VIF value for this model to check multicollinearity**

library(car)

library(caret)

library(e1071)

vif(mod3)

# now all variables are significant and vif value is also okay so this model mod3 is finalized

# Taking top 5 factors most influencing the target variable

head(sort(abs(mod3$coefficients), decreasing = T),6)

### Model Validation:

table(data$target)/nrow(data)

pred<-predict (mod3, type="response”, newdata=test)

pred<-ifelse (pred>=0.24, 1, 0)

pred=as.factor (pred)

#### Confusion matrix is for checking model accuracy:

confusionMatrix (pred, test$target, positive="1")

Output:

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 5934 374

1 1477 1984

Accuracy: 0.8105 95%

CI: (0.8026, 0.8183)

No Information Rate: 0.7586

P-Value [Acc > NIR]: < 2.2e-16

Kappa: 0.5538

Mcnemar's Test P-Value: < 2.2e-16

Sensitivity: 0.8414

Specificity: 0.8007

Pos Pred Value: 0.5732

Neg Pred Value: 0.9407

Prevalence: 0.2414

Detection Rate: 0.2031

Detection Prevalence: 0.3543

Balanced Accuracy: 0.8210

'Positive' Class: 1

**Conclusion:** Confusion matrix, precision, recall, and F1 score provides better insights into the prediction as compared to accuracy performance metrics. Applications of precision, recall, and F1 score is in information retrieval, word segmentation, named entity recognition, and many more.