In [2]:

```
# Author : Wish MKN
#
#
#pip install pyspark
#
#
```

```
Collecting pyspark
  Downloading pyspark-3.2.0.tar.gz (281.3 MB)
Collecting py4j==0.10.9.2
  Downloading py4j-0.10.9.2-py2.py3-none-any.whl (198 kB)
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py): started
  Building wheel for pyspark (setup.py): finished with status 'done'
  Created wheel for pyspark: filename=pyspark-3.2.0-py2.py3-none-any.whl siz
e=281805913 sha256=5f2938d9184f773b50dc8912b1fe3fdc122661e13e24ad196f5e041d1
14a3586
  Stored in directory: c:\users\wish\appdata\local\pip\cache\wheels\23\f6\d3
\110e53bd43baeb8d7d38049733d48e39cbecd056f01dba7ee8
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9.2 pyspark-3.2.0
Note: you may need to restart the kernel to use updated packages.
```

In [1]:

```
from pyspark.sql import SparkSession
```

an entry point to PySpark to work with Resilient Distributed Dataset, DataFrame these are immutable

In [2]:

```
spark = SparkSession.builder.appName("Model_Prediction_Log").getOrCreate()

# entry point to spark sql
```

In [3]:

```
user_interaction_df = spark.read.csv("dataframe.csv", inferSchema = True, header = True)
```

Data imported

and displaying schema to get overview about dataset and printing the dataset using .show() and get column names

In [4]:

```
user_interaction_df.printSchema()
```

```
root
 |-- user_id: string (nullable = true)
 |-- month_interaction_count: integer (nullable = true)
 |-- week_interaction_count: integer (nullable = true)
 |-- day_interaction_count: integer (nullable = true)
 |-- cancelled_within_week: integer (nullable = true)
```

In [5]:

```
user_interaction_df.show()
```

```
+--------+----------------------+---------------------+------------------
--+--------------------+
| user_id|month_interaction_count|week_interaction_count|day_interaction_cou
nt|cancelled_within_week|
+--------+----------------------+---------------------+------------------
--+--------------------+
|66860ae6|                    41|                    9|
0|                   1|
|249803f8|                    25|                    9|
2|                   0|
|32ed74cc|                    21|                    2|
1|                   1|
|7ed76e6a|                    22|                    5|
2|                   0|
|46c81f43|                    32|                    8|
2|                   0|
|cf0f185e|                    26|                    4|
0|                   1|
|568275b3|                    29|                    5|
1|                   1|
|86a060ec|                    33|                    7|
1|                   1|
|c0c07290|                    35|                   10|
0|                   0|
|709dc1da|                    36|                   11|
1|                   0|
+--------+----------------------+---------------------+------------------
--+--------------------+
```

In [6]:

```
user_interaction_df.columns
```

Out[6]:

```
['user_id',
 'month_interaction_count',
 'week_interaction_count',
 'day_interaction_count',
 'cancelled_within_week']
```

Features

'month_interaction_count', 'week_interaction_count', 'day_interaction_count',

Label

'cancelled_within_week'

In [7]:

```
from pyspark.ml.feature import VectorAssembler
```

In [8]:

```
assembler = VectorAssembler(inputCols = ['month_interaction_count','week_interaction_count'
                                         'cancelled_within_week'], outputCol = "features")
```

Created features using the VectorAssembler and then in final dataset we merge all features column with the cancelled_within_week column which is our label column

In [9]:

```
output = assembler.transform(user_interaction_df)
```

In [10]:

```
user_interaction_df_final = output.select("features","cancelled_within_week")
```

In [11]:

```
user_interaction_df_final.show()
```

```
+------------------+--------------------+
|          features|cancelled_within_week|
+------------------+--------------------+
| [41.0,9.0,0.0,1.0]|                   1|
| [25.0,9.0,2.0,0.0]|                   0|
| [21.0,2.0,1.0,1.0]|                   1|
| [22.0,5.0,2.0,0.0]|                   0|
| [32.0,8.0,2.0,0.0]|                   0|
| [26.0,4.0,0.0,1.0]|                   1|
| [29.0,5.0,1.0,1.0]|                   1|
| [33.0,7.0,1.0,1.0]|                   1|
|[35.0,10.0,0.0,0.0]|                   0|
|[36.0,11.0,1.0,0.0]|                   0|
+------------------+--------------------+
```

In [12]:

```
train, test = user_interaction_df_final.randomSplit([0.7,0.3], seed = 42)
```

dividing the dataset into train and test dataset by giving 70% as training and 30% as testing

In [13]:

```
from pyspark.ml.classification import LogisticRegression
```

importing the LogisticRegression package and creating LR (logistic regression) model

In [14]:

```
LR = LogisticRegression(labelCol = "cancelled_within_week", elasticNetParam=1)
LR.setRegParam(0.1)
LR.setThreshold(0.6)
```

Out[14]:

```
LogisticRegression_13e6d2b88312
```

LR model created with required below :

1. L1 regularization panelty 'elasticNetParam'=1 , the alpha value 1 = L1 penalty and 0 = L2 penalty

2. regularization parameter 0.1 , the lamda value

3. threshold 0.6, if robability is equal more than 0.6 then the label column will have a value of 1 otherwise 0

In [15]:

```
LogRModel = LR.fit(train)
```

Train the model on training data

In [16]:

```
LogRModel_summary = LogRModel.summary
```

assign model summary object to another variable so later we can access it seprately

In [17]:

```
LogRModel_summary.predictions.show()
```

```
C:\Users\Wish\anaconda3\lib\site-packages\pyspark\sql\context.py:125: Future
Warning: Deprecated in 3.0.0. Use SparkSession.builder.getOrCreate() instea
d.
  warnings.warn(

+------------------+--------------------+--------------------+-----------
--------+----------+
|          features|cancelled_within_week|       rawPrediction|         pro
bability|prediction|
+------------------+--------------------+--------------------+-----------
--------+----------+
| [21.0,2.0,1.0,1.0]|                 1.0|[-2.0954021468336...|[0.109544512
02978...|       1.0|
| [22.0,5.0,2.0,0.0]|                 0.0|[2.09540277225602...|[0.890455548
97672...|       0.0|
| [26.0,4.0,0.0,1.0]|                 1.0|[-2.0954021468336...|[0.109544512
02978...|       1.0|
| [29.0,5.0,1.0,1.0]|                 1.0|[-2.0954021468336...|[0.109544512
02978...|       1.0|
| [32.0,8.0,2.0,0.0]|                 0.0|[2.09540277225602...|[0.890455548
97672...|       0.0|
|[35.0,10.0,0.0,0.0]|                 0.0|[2.09540277225602...|[0.890455548
97672...|       0.0|
+------------------+--------------------+--------------------+-----------
--------+----------+
```

In [19]:

```
from pyspark.ml.evaluation import BinaryClassificationEvaluator
```

BinaryClassificationEvaluator as we have values of 0 or 1 if probability equal or greater than 0.6 threshold value

In [20]:

```
pred_labels = LogRModel.evaluate(test)
```

Model now runs on the test dataset. We compare the cancelled_within_week with prediction

compare actual with predicted to get rough idea of how good our model can be

In [21]:

```
pred_labels.predictions.show()
```

```
+------------------+--------------------+--------------------+-----------
--------+----------+
|          features|cancelled_within_week|       rawPrediction|         pro
bability|prediction|
+------------------+--------------------+--------------------+-----------
--------+----------+
| [25.0,9.0,2.0,0.0]|                   0|[2.09540277225602...|[0.890455548
97672...|       0.0|
| [33.0,7.0,1.0,1.0]|                   1|[-2.0954021468336...|[0.109544512
02978...|       1.0|
|[36.0,11.0,1.0,0.0]|                   0|[2.09540277225602...|[0.890455548
97672...|       0.0|
| [41.0,9.0,0.0,1.0]|                   1|[-2.0954021468336...|[0.109544512
02978...|       1.0|
+------------------+--------------------+--------------------+-----------
--------+----------+
```

In [22]:

```
evaluation_Res = BinaryClassificationEvaluator(rawPredictionCol = "prediction", labelCol =
```

Evaluating our model

In [23]:

```
auc = evaluation_Res.evaluate(pred_labels.predictions)
```

In [24]:

```
auc
```

Out[24]:

```
1.0
```

get AUC (Area under the curve) value,

When AUC = 1, then the classifier is able to perfectly distinguish between all the Positive and the Negative class points correctly

How Does the AUC-ROC Curve Work?

A ROC curve, a higher X-axis value indicates a higher number of False positives than True negatives. While a higher Y-axis value indicates a higher number of True positives than False negatives. So, the choice of the threshold depends on the ability to balance between False positives and False negatives.

In [25]:

```
#<new data with same columns but without cancelled_within_week column>
# <new_data_name> = spark.read.csv("dataframe.csv", inferSchema = True, header = True)
```

In [30]:

```
final_model = LR.fit(user_interaction_df_final)   # set model on complete data set that we
```

In [33]:

```
user_interaction_df_valid = assembler.transform(user_interaction_df)

# can test for new data set from here which does not have
# cancelled withing week column
# so instead we put assebler.transform(<new data set after importing with same columns>)
```

In [34]:

```
user_interaction_df_valid.printSchema()
```

```
root
 |-- user_id: string (nullable = true)
 |-- month_interaction_count: integer (nullable = true)
 |-- week_interaction_count: integer (nullable = true)
 |-- day_interaction_count: integer (nullable = true)
 |-- cancelled_within_week: integer (nullable = true)
 |-- features: vector (nullable = true)
```

In [35]:

```
res = final_model.transform(user_interaction_df_valid)
```

In [36]:

```
res.select('user_id','prediction').show()
```

```
+--------+----------+
| user_id|prediction|
+--------+----------+
|66860ae6|       1.0|
|249803f8|       0.0|
|32ed74cc|       1.0|
|7ed76e6a|       0.0|
|46c81f43|       0.0|
|cf0f185e|       1.0|
|568275b3|       1.0|
|86a060ec|       1.0|
|c0c07290|       0.0|
|709dc1da|       0.0|
+--------+----------+
```

In [ ]:

```

```