First we Import the required Libs for the Model

In [4]:
```
import findspark
findspark.init()
import pyspark
from pyspark.sql.types import *
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('Amzon_Demo').getOrCreate()
from pyspark import SparkContext
sc= spark.sparkContext
import pandas as pd
import base64
import string
import re
from collections import Counter
from pyspark.sql.functions import isnan, when, count, col
from pyspark.sql.types import StructType
from pyspark.sql.functions import col, lower, regexp_replace, split
from pyspark.sql.functions import udf, col, lower, regexp_replace
from pyspark.ml.feature import Tokenizer, StopWordsRemover
import pyspark.sql.functions as f
from pyspark.ml.feature import RegexTokenizer, StopWordsRemover, CountVectorizer
from pyspark.ml import Pipeline
from pyspark.ml.feature import OneHotEncoder, StringIndexer, VectorAssembler
locale = sc._jvm.java.util.Locale
locale.setDefault(locale.forLanguageTag("en-US"))
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
```

Import the datasets from HDFS Hadoop

In [3]:
```
df = spark.read.csv("Roman1.csv")
df.show(10)
df.printSchema()
```

```
+--------------------+--------+----+
|                 _c0|     _c1| _c2|
+--------------------+--------+----+
|Sai kha ya her ki...|Positive|null|
|           sahi bt h|Positive|null|
|         Kya bt hai,|Positive|null|
|          Wah je wah|Positive|null|
|Are wha kaya bat hai|Positive|null|
|  Wah kya baat likhi|Positive|null|
|Wha Itni sari khu...|Positive|null|
|        Itni khubiya|Positive|null|
|Ya allah rehm far...|Positive|null|
|Please Everyone A...|Positive|null|
+--------------------+--------+----+
only showing top 10 rows

root
 |-- _c0: string (nullable = true)
 |-- _c1: string (nullable = true)
 |-- _c2: string (nullable = true)
```

Change the dataFrame columns names and then drop the unwatned columns

In [3]:
```
newNames  = ['Comment', 'Value', 'None']
dfRenamed = df.toDF(*newNames)
columns_to_drop = ['None']
df = dfRenamed.drop(*columns_to_drop)
df.printSchema()
df.show(10)
```

```
root
 |-- Comment: string (nullable = true)
 |-- Value: string (nullable = true)

+--------------------+--------+
|             Comment|   Value|
+--------------------+--------+
|Sai kha ya her ki...|Positive|
|           sahi bt h|Positive|
|         Kya bt hai,|Positive|
|          Wah je wah|Positive|
|Are wha kaya bat hai|Positive|
|  Wah kya baat likhi|Positive|
|Wha Itni sari khu...|Positive|
|        Itni khubiya|Positive|
|Ya allah rehm far...|Positive|
|Please Everyone A...|Positive|
|Ya mere rab tu br...|Positive|
|jaago Pakistani c...|Positive|
|kia kia jae .kon ...|Positive|
|afsos hota hai ga...|Positive|
|  Allah insaaf karey|Positive|
|ALLAH MUSALMAN PH...|Positive|
|  Je Thik Kaha Right|Positive|
|        jee ye to he|Positive|
|Maa cheez e ASE h...|Positive|
|  Wah wah yeh toh hy|Positive|
+--------------------+--------+
only showing top 20 rows
```

Count the null-values and then group them by the category

In [4]:
```
df.select([count(when(isnan(c), c)).alias(c) for c in df.columns]).show()

from pyspark.sql.functions import col
df.groupBy("Value") \
    .count() \
    .orderBy(col("count").desc()) \
    .show(10)
```

```
+-------+-----+
|Comment|Value|
+-------+-----+
|      0|    0|
+-------+-----+

+--------------------+-----+
|               Value|count|
+--------------------+-----+
|             Neutral| 8913|
|            Positive| 5974|
|            Negative| 5269|
|                null|  336|
| jo meine inko de...|    2|
| lakin mobile ko ...|    2|
| yaha jo bhee hID...|    2|
| aur Punjab bar c...|    1|
| there were some ...|    1|
| director un ke d...|    1|
| wanted salt"" au...|    1|
|                Apne|    1|
| Islam khata hay ...|    1|
|               laila|    1|
| Phantom aur Jagg...|    1|
| Ali Zafar ki tar...|    1|
| tou wapis karado...|    1|
|"" k name Saray D...|    1|
| wah kya bat hai ...|    1|
| par phassti nahi...|    1|
+--------------------+-----+
only showing top 20 rows
```

Clean the Dataframe & Changing the emi to real value to be considerd in our model

In [5]:
```
def clean_text(c):
    c = lower(c)
    c = regexp_replace(c, '😊|😉|😁|😌|😂|😄|♥|😋|😎|😃|😈|😍|🎁|❤|😆|👌|😳|😩|💖|😔|💋|💘', " 111 ")
    c = regexp_replace(c, '😒|😞|😫|😑|😭|😢|😩|😐|😓|👊|😦|😟', " 222 ")


    return c

def clean_text_2(c):

   c = regexp_replace(c, "^rt ", " ")
   c = regexp_replace(c, "(https?\://)\S+", " ")
   c = regexp_replace(c, "[^a-zA-Z0-9\\s]", " ")
   c = regexp_replace(c, "[^\w\s#@/:%.,_-]", " ")
   #c = regexp_replace(c, "[0-9]", " ")


   return c

data_1 = dfRenamed.select((clean_text(col("Comment")).alias("Comment")), ((clean_text(col("Value")).alias("Va
lue"))))
data_2 = data_1.select((clean_text_2(col("Comment")).alias("Comment")), ((clean_text_2(col("Value")).alias("V
alue"))))
```

Let's Check the Categories again and clean the select only the Categories we want

In [15]:
```
from pyspark.sql.functions import col

data_2.groupBy("Value") \
    .count() \
    .orderBy(col("count").desc()) \
    .show(20)

df2 = data_2.filter(data_2.Comment. isNotNull() & (data_2.Value == "negative")==True)
df3 = data_2.filter(data_2.Comment. isNotNull() & (data_2.Value == "positive")==True)
df4 = data_2.filter(data_2.Comment. isNotNull() & (data_2.Value == "neutral")==True)

print((df2.count(), len(df2.columns)))
df2.show(5)
print((df3.count(), len(df3.columns)))
df3.show(5)
print((df4.count(), len(df4.columns)))
df4.show(5)
```

```
+--------------------+-----+
|               Value|count|
+--------------------+-----+
|             neutral| 8913|
|            positive| 5974|
|            negative| 5269|
|                null|  336|
| lakin mobile ko ...|    2|
| jo meine inko de...|    2|
| yaha jo bhee hid...|    2|
| playing by heart...|    1|
| stop copying  111  |    1|
| there were some ...|    1|
| islam khata hay ...|    1|
| director un ke d...|    1|
| bar asosi aishio...|    1|
| tomb raider   me...|    1|
| singh is king au...|    1|
| sense should pre...|    1|
|    hasad na kar    |    1|
|     medora lipstick|    1|
| 1st time user don t|    1|
| tou wapis karado...|    1|
+--------------------+-----+
only showing top 20 rows

(5269, 2)
+--------------------+--------+
|             Comment|   Value|
+--------------------+--------+
|asif momin hakir ...|negative|
|phely jaa kr naha...|negative|
|ye to bilkul thk ...|negative|
|dukh hi dukh zind...|negative|
|or ya assa he hot...|negative|
+--------------------+--------+
only showing top 5 rows

(5974, 2)
+--------------------+--------+
|             Comment|   Value|
+--------------------+--------+
|sai kha ya her ki...|positive|
```

```
|           sahi bt h|positive|
|         kya bt hai |positive|
|          wah je wah|positive|
|are wha kaya bat hai|positive|
+--------------------+--------+
only showing top 5 rows

(8912, 2)
+--------------------+-------+
|             Comment|  Value|
+--------------------+-------+
|          hakeqat hy|neutral|
|aor aisy bahut km...|neutral|
|        jee ye to he|neutral|
|hmm jysa kro gy w...|neutral|
|ye kia hoa raha h...|neutral|
|ghreeb k ghr subh...|neutral|
|      ye kia bat hoe|neutral|
|ri8 but naseeb ki...|neutral|
|  ya post sabka laiy|neutral|
|khabi khabi mera ...|neutral|
+--------------------+-------+
only showing top 10 rows
```

Connecting all the data and make it as one dataframe

In [7]:
```
print((df2.count(), len(df2.columns)))
df2.show(5)
print((df3.count(), len(df3.columns)))
df3.show(5)
print((df4.count(), len(df4.columns)))
df4.show(5)

df5 = df2.union(df3)
df_final=df5.union(df4)

print((df_final.count(), len(df_final.columns)))
df_final.show(5)
```

```
(5269, 2)
+--------------------+--------+
|             Comment|   Value|
+--------------------+--------+
|asif momin hakir ...|negative|
|phely jaa kr naha...|negative|
|ye to bilkul thk ...|negative|
|dukh hi dukh zind...|negative|
|or ya assa he hot...|negative|
+--------------------+--------+
only showing top 5 rows

(5974, 2)
+--------------------+--------+
|             Comment|   Value|
+--------------------+--------+
|sai kha ya her ki...|positive|
|           sahi bt h|positive|
|         kya bt hai |positive|
|          wah je wah|positive|
|are wha kaya bat hai|positive|
+--------------------+--------+
only showing top 5 rows

(8912, 2)
+--------------------+-------+
|             Comment|  Value|
+--------------------+-------+
|          hakeqat hy|neutral|
|aor aisy bahut km...|neutral|
|        jee ye to he|neutral|
|hmm jysa kro gy w...|neutral|
|ye kia hoa raha h...|neutral|
+--------------------+-------+
only showing top 5 rows

(20155, 2)
+--------------------+--------+
|             Comment|   Value|
+--------------------+--------+
|asif momin hakir ...|negative|
|phely jaa kr naha...|negative|
|ye to bilkul thk ...|negative|
```

```
|dukh hi dukh zind...|negative|
|or ya assa he hot...|negative|
+--------------------+--------+
only showing top 5 rows
```

Now we can tokenize the dataframe and delete the less than 3 char tokens

In [8]:
```
from pyspark.ml.feature import RegexTokenizer, StopWordsRemover, CountVectorizer
from pyspark.ml.classification import LogisticRegression
# regular expression tokenizer
regexTokenizer = RegexTokenizer(inputCol="Comment", outputCol="words_token", pattern="\\W")
df_words_token = regexTokenizer.transform(df_final).select('Value', 'words_token')

df222 = df_words_token.withColumn("Comment", f.expr("filter(words_token, x -> not(length(x) < 3))")).where(f.
size(f.col("Comment")) > 0).drop("words_token")
print((df222.count(), len(df222.columns)))
df222.show(5)
```

```
(19895, 2)
+--------+--------------------+
|   Value|             Comment|
+--------+--------------------+
|negative|[asif, momin, hak...|
|negative|[phely, jaa, naha...|
|negative|[bilkul, thk, kah...|
|negative|[dukh, dukh, zind...|
|negative|[assa, hotta, jas...|
|negative|[twadi, bahan, no...|
|negative|[agree, main, sth...|
|negative|[ghareeb, insan, ...|
|negative|[plz, police, tha...|
|negative|[sindh, police, t...|
+--------+--------------------+
only showing top 10 rows
```

Now lets prepare our dataframe to our machine learning models by removing unwanted words and doing bags of words

Then we will create pipeline to push the dataset into it

In [9]:
```
from pyspark.ml.feature import RegexTokenizer, StopWordsRemover, CountVectorizer
from pyspark.ml.classification import LogisticRegression
# regular expression tokenizer


# stop words
remover = StopWordsRemover()
stopwords = remover.getStopWords()
stopwordsRemover = StopWordsRemover(inputCol="Comment", outputCol="filtered").setStopWords(stopwords)

countVectors = CountVectorizer(inputCol="filtered", outputCol="features", vocabSize=10000, minDF=3)

label_stringIdx = StringIndexer(inputCol = "Value", outputCol = "label")
pipeline = Pipeline(stages=[stopwordsRemover, countVectors, label_stringIdx])

# Fit the pipeline to training documents.
pipelineFit = pipeline.fit(df222)
dataset = pipelineFit.transform(df222)
dataset.show(20)
# bag of words count
```

```
+--------+--------------------+--------------------+--------------------+-----+
|   Value|             Comment|            filtered|            features|label|
+--------+--------------------+--------------------+--------------------+-----+
|negative|[asif, momin, hak...|[asif, momin, hak...|(8753,[1,10,82,95...|  2.0|
|negative|[phely, jaa, naha...|[phely, jaa, naha...|(8753,[193,1392,2...|  2.0|
|negative|[bilkul, thk, kah...|[bilkul, thk, kah...|(8753,[2,75,123,1...|  2.0|
|negative|[dukh, dukh, zind...|[dukh, dukh, zind...|(8753,[0,1315],[1...|  2.0|
|negative|[assa, hotta, jas...|[assa, hotta, jas...|(8753,[5,28,108,1...|  2.0|
|negative|[twadi, bahan, no...|[twadi, bahan, no...|(8753,[1,2,4,5,6,...|  2.0|
|negative|[agree, main, sth...|[agree, main, sth...|(8753,[9,12,124,3...|  2.0|
|negative|[ghareeb, insan, ...|[ghareeb, insan, ...|(8753,[2,7,12,53,...|  2.0|
|negative|[plz, police, tha...|[plz, police, tha...|(8753,[92,190,115...|  2.0|
|negative|[sindh, police, t...|[sindh, police, t...|(8753,[7,15,33,81...|  2.0|
|negative|[pakistan, tabah,...|[pakistan, tabah,...|(8753,[1,15,160,5...|  2.0|
|negative|[kia, hogya, poli...|[kia, hogya, poli...|(8753,[9,19,71,15...|  2.0|
|negative|[nawaz, sharif, p...|[nawaz, sharif, p...|(8753,[68,115,137...|  2.0|
|negative|[police, walon, m...|[police, walon, m...|(8753,[35,68,190,...|  2.0|
|negative|[police, wala, ak...|[police, wala, ak...|(8753,[9,69,98,12...|  2.0|
|negative|[police, per, lan...|[police, per, lan...|(8753,[16,30,33,1...|  2.0|
|negative|[mary, khyal, pol...|[mary, khyal, pol...|(8753,[190,977,28...|  2.0|
|negative|[coda, kro, polic...|[coda, kro, polic...|(8753,[184,190,44...|  2.0|
|negative|[mere, sat, bhi, ...|[mere, sat, bhi, ...|(8753,[4,6,22,50,...|  2.0|
|negative|[bherwe, tujhe, a...|[bherwe, tujhe, a...|(8753,[25,65,119,...|  2.0|
+--------+--------------------+--------------------+--------------------+-----+
only showing top 20 rows
```

Lets divide our datasets into trainsets and testsets for testing

In [10]:
```
# split the data and count them
(trainingData, testData) = dataset.randomSplit([0.8, 0.2], seed = 100)

print("Training Dataset Count: " + str(trainingData.count()))
print("Test Dataset Count: " + str(testData.count()))
```

```
Training Dataset Count: 15940
Test Dataset Count: 3955
```

Lets test our models with the algorithms we got in pyspark and do Multiclass-Classification-Evaluator

In [11]:
```
from pyspark.ml.classification import LogisticRegression
lr = LogisticRegression(maxIter=20, regParam=0.3, elasticNetParam=0)
lrModel = lr.fit(trainingData)
predictions = lrModel.transform(testData)
predictions.filter(predictions['prediction'] == 0) \
    .select("Comment","Value","probability","label","prediction") \
    .orderBy("probability", ascending=False) \
    .show(n = 10, truncate = 30)

from pyspark.ml.evaluation import MulticlassClassificationEvaluator
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction")
evaluator.evaluate(predictions)
```

```
+------------------------------+--------+------------------------------+-----+----------+
|                       Comment|   Value|                   probability|label|prediction|
+------------------------------+--------+------------------------------+-----+----------+
|[sonal, chauhanne, telugu, ...|positive|[0.9948291268712804,0.00481...|  1.0|       0.0|
|[ohh, got, real, competitio...| neutral|[0.9814307339348136,0.01152...|  0.0|       0.0|
|[1973, mein, harvard, unive...| neutral|[0.9779763800686536,0.01588...|  0.0|       0.0|
|[brigadier, hesiyat, unhon,...| neutral|[0.9759593428164509,0.01918...|  0.0|       0.0|
|[sri, lanka, batsman, kumar...|positive|[0.9754151876781293,0.01338...|  1.0|       0.0|
|[jab, guzishta, aik, saal, ...| neutral|[0.9712226647832201,0.02275...|  0.0|       0.0|
|[primary, school, number, w...| neutral|[0.9657880870496699,0.01650...|  0.0|       0.0|
|[chunache, pir, pagara, pak...| neutral|[0.9655016609864637,0.01896...|  0.0|       0.0|
|[tahum, maut, years, bad, a...| neutral|[0.9622694992836397,0.01044...|  0.0|       0.0|
|[tasaneef, mein, sirf, book...| neutral|[0.9620653338915889,0.02257...|  0.0|       0.0|
+------------------------------+--------+------------------------------+-----+----------+
only showing top 10 rows
```

Out[11]: 0.6238593500842845

Lets add HashingTF and IDF Methodes to our LogisticRegression Model

In [12]:
```
from pyspark.ml.feature import HashingTF, IDF
hashingTF = HashingTF(inputCol="filtered", outputCol="rawFeatures", numFeatures=10000)
idf = IDF(inputCol="rawFeatures", outputCol="features", minDocFreq=2) #minDocFreq: remove sparse terms
pipeline = Pipeline(stages=[stopwordsRemover, hashingTF, idf, label_stringIdx])
pipelineFit = pipeline.fit(df222)
dataset = pipelineFit.transform(df222)
(trainingData, testData) = dataset.randomSplit([0.8, 0.2], seed = 100)
lr = LogisticRegression(maxIter=20, regParam=0.3, elasticNetParam=0)
lrModel = lr.fit(trainingData)
predictions = lrModel.transform(testData)
predictions.filter(predictions['prediction'] == 0) \
    .select("Comment","Value","probability","label","prediction") \
    .orderBy("probability", ascending=False) \
    .show(n = 10, truncate = 30)

from pyspark.ml.evaluation import MulticlassClassificationEvaluator
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction")
evaluator.evaluate(predictions)
```

```
+------------------------------+--------+------------------------------+-----+----------+
|                       Comment|   Value|                   probability|label|prediction|
+------------------------------+--------+------------------------------+-----+----------+
|[begum, liaquat, batan, far...| neutral|[0.9850723268672922,0.00944...|  0.0|       0.0|
|[1973, mein, harvard, unive...| neutral|[0.9674589746691525,0.02654...|  0.0|       0.0|
|     [smja, inhy, momina, kis]|negative|[0.9592677565775155,0.01254...|  2.0|       0.0|
|[zulfiqar, ali, bhutto, nay...| neutral|[0.9569974927345575,0.01956...|  0.0|       0.0|
|     [kiya, kehh, sakte, henn]| neutral|[0.952319260852304,0.020049...|  0.0|       0.0|
|[rice, isqaat, hamal, khila...|negative|[0.9505715375953411,0.01304...|  2.0|       0.0|
|[come, tuhadee, phuphee, lu...|negative|[0.9495398376280736,0.02369...|  2.0|       0.0|
|[helen, keller, 1880, mein,...| neutral|[0.9487525020127384,0.01849...|  0.0|       0.0|
|[anab, altaf, hussain, nay,...|positive|[0.9468301617032723,0.03737...|  1.0|       0.0|
|[zilla, aur, tehsil, counce...|negative|[0.9462576116635412,0.02488...|  2.0|       0.0|
+------------------------------+--------+------------------------------+-----+----------+
only showing top 10 rows
```

Out[12]: 0.5884725888089297

Lets do CrossOver Methodes hopfully we will add some more accurcy to our model

In [13]:
```
pipeline = Pipeline(stages=[stopwordsRemover, countVectors, label_stringIdx])
pipelineFit = pipeline.fit(df222)
dataset = pipelineFit.transform(df222)
(trainingData, testData) = dataset.randomSplit([0.8, 0.2], seed = 100)
lr = LogisticRegression(maxIter=20, regParam=0.3, elasticNetParam=0)
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
# Create ParamGrid for Cross Validation
paramGrid = (ParamGridBuilder()
             .addGrid(lr.regParam, [0.1, 0.3, 0.5]) # regularization parameter
             .addGrid(lr.elasticNetParam, [0.0, 0.1, 0.2]) # Elastic Net Parameter (Ridge = 0)
             .build())
# Create 5-fold CrossValidator
cv = CrossValidator(estimator=lr, \
                    estimatorParamMaps=paramGrid, \
                    evaluator=evaluator, \
                    numFolds=5)
cvModel = cv.fit(trainingData)

predictions = cvModel.transform(testData)
# Evaluate best model
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction")
evaluator.evaluate(predictions)
```

Out[13]: 0.6219636491887839

its dose not change that much. Lets try another classfication algorithem - NaiveBayes

In [14]:
```
from pyspark.ml.classification import NaiveBayes
nb = NaiveBayes(smoothing=5)
model = nb.fit(trainingData)
predictions = model.transform(testData)
predictions.filter(predictions['prediction'] == 0) \
    .select("Comment","Value","probability","label","prediction") \
    .orderBy("probability", ascending=False) \
    .show(n = 10, truncate = 30)


from pyspark.ml.evaluation import MulticlassClassificationEvaluator
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction")
evaluator.evaluate(predictions)
```

```
+------------------------------+--------+------------------------------+-----+----------+
|                       Comment|   Value|                   probability|label|prediction|
+------------------------------+--------+------------------------------+-----+----------+
|[zulfiqar, ali, bhutto, nay...| neutral|[0.9999674360403801,3.24917...|  0.0|       0.0|
|[phupo, thanks, bolin, meri...|positive|[0.9998634583068682,9.71751...|  1.0|       0.0|
|[meri, tarf, apny, parents,...| neutral|[0.9997063494520431,1.27534...|  0.0|       0.0|
|[inshallah, hum, dono, kaam...|positive|[0.9996429337791103,1.87496...|  1.0|       0.0|
|[pictures, screenshot, maar...|negative|[0.9995571475107323,2.18137...|  2.0|       0.0|
|[hahahahahah, 111, 111, 111...| neutral|[0.9995073048266958,3.93420...|  0.0|       0.0|
|[111, 111, 111, thnkew, den...| neutral|[0.999439563014437,1.928399...|  0.0|       0.0|
|[uncle, anty, thanks, kehna...| neutral|[0.9992770363538133,5.75694...|  0.0|       0.0|
|[hahaha, new, wala, pehno, ...| neutral|[0.9991893573978425,1.51595...|  0.0|       0.0|
|[111, bhi, thank, you, bol,...| neutral|[0.9990677568758084,6.12351...|  0.0|       0.0|
+------------------------------+--------+------------------------------+-----+----------+
only showing top 10 rows
```

Out[14]: 0.648457283205311

We notice we get some improvements in our model. Considering the given data we have in roman urdu 65 accuracy its good for multiclassfiction text sentiment problem

In [ ]:
```
#limitation of the data used
# 1- The data we used was entirely missy contain numbers emojis and other non-english characters
# 2- The data used for foreign Language written in english characters which make it difficult for us to use t
echniques such as
# StopWordRemover and Normalization such as stemming and lemmatization
# 3- The datasets collected  not accurate. We find lots of negative comments contain happy words such as ( lo
l, hahhahha,
# happy emoj, and postive words in urdu language )
# 4- The datasets unbalanced (6000 postive comments, 6000 negative comments and around 8000 neutral comments
 )
# however there is lots of numbers and none-sense datasets (Comments) in netural comments...after clean it ca
rfully
# the data become somehow close in terms of numbers (Balanced) becasue lots of null datasets deleted.
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: