**Task 5: Classification Algorithm: Text classification with pipeline**

**Objective:** Demonstrate how pipeline helps to compac the code

#Prepared input dataset

training = spark.createDataFrame([

("We had a perfectly pleasant stay here in December.", 1),

("Stayed at this hotel again and it was as good as last year. Great service, perfect location and very clean.", 1),

("Very negative experience when trying to get a refund from my travelocity booking that was cancelled due to a weather delay .", 0),

("Staff is very green young, restaurant is nice but forget it worst service ever. breakfast, burned toasts, cold eggs waited forever to be seated and for the food. ", 0)

], ["review", "label"])

#Tokenize sentences

from pyspark.ml.feature import Tokenizer

tokenizer = Tokenizer(inputCol="review", outputCol="words")

#Generate hash for each word

from pyspark.ml.feature import HashingTF

hashingTF = HashingTF(inputCol=tokenizer.getOutputCol(), outputCol="features")

#Create LogisticRegression instance

from pyspark.ml.classification import LogisticRegression

lr = LogisticRegression(maxIter=10)

#Create ML pipeline

from pyspark.ml import Pipeline

lr\_pipeline = Pipeline(stages=[tokenizer, hashingTF, lr])

**#Train the model**

model = lr\_pipeline.fit(training)

#Predict

test = spark.createDataFrame([

("Very negative experience. Not impressed. I won’t be staying here again and I recommend you don’t either.",),

("Great service, perfect location and very clean.",),

("The staff were all excellent. Super pleasant and courteous.",),

("worst service ever. waited forever to be seated and for the food.",)

], ["review"])

prediction = model.transform(test\_htf)