```
# Import necessary libraries
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.classification import LogisticRegression, RandomForestClassifier, DecisionTreeClassifier, GBTClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator


spark = SparkSession.builder \
    .appName("Customer Churn Prediction") \
    .getOrCreate()


# upload the file
from google.colab import files
uploaded = files.upload()
```

Choose Files customer c… dataset.csv

- **customer ch pred dataset.csv**(text/csv) - 684858 bytes, last modified: 10/15/2024 - 100% done

```
Saving customer ch pred dataset.csv to customer ch pred dataset.csv


# reading the file
data = spark.read.csv("/content/customer ch pred dataset.csv", header=True, inferSchema=True)
data.show(5)
```

```
+---------+----------+--------+-----------+---------+------+---+------+---------+-------------+---------+--------------+-----------
|RowNumber|CustomerId| Surname|CreditScore|Geography|Gender|Age|Tenure|  Balance|NumOfProducts|HasCrCard|IsActiveMember|EstimatedSal
+---------+----------+--------+-----------+---------+------+---+------+---------+-------------+---------+--------------+-----------
|        1|  15634602|Hargrave|        619|   France|Female| 42|     2|      0.0|            1|        1|             1|      101348
|        2|  15647311|    Hill|        608|    Spain|Female| 41|     1| 83807.86|            1|        0|             1|      112542
|        3|  15619304|    Onio|        502|   France|Female| 42|     8| 159660.8|            3|        1|             0|      113931
|        4|  15701354|    Boni|        699|   France|Female| 39|     1|      0.0|            2|        0|             0|       93826
|        5|  15737888|Mitchell|        850|    Spain|Female| 43|     2|125510.82|            1|        1|             1|        7908
+---------+----------+--------+-----------+---------+------+---+------+---------+-------------+---------+--------------+-----------
only showing top 5 rows
```

```
# data preprocessing
data = data.na.drop()


# Feature Selection
features = ["Age", "Balance", "EstimatedSalary"]
label = 'Exited'


# Vector Assembler - Combines the feature columns into a single feature vector.
assembler = VectorAssembler(inputCols=features, outputCol='features')
data = assembler.transform(data)


# Splitting the data
train_data, test_data = data.randomSplit([0.8, 0.2], seed=42)


# Model Training
lr = LogisticRegression(labelCol=label, featuresCol='features')
lr_model = lr.fit(train_data)
lr_model
```

```
LogisticRegressionModel: uid=LogisticRegression_a6310e3e2989, numClasses=2, numFeatures=3


rf = RandomForestClassifier(labelCol=label, featuresCol='features')
rf_model = rf.fit(train_data)
rf
```

```
RandomForestClassifier_cd4829cf4196


# Model Evaluation
def evaluate_model(model):
    predictions = model.transform(test_data)
    evaluator = MulticlassClassificationEvaluator(labelCol=label, predictionCol="prediction", metricName="f1")
    f1_score = evaluator.evaluate(predictions)
    accuracy = predictions.filter(predictions[label] == predictions['prediction']).count() / float(predictions.count())
    return f1_score, accuracy


# Evaluate each model
models = {'Logistic Regression': lr_model, 'Random Forest': rf_model}
```

```
for name, model in models.items():
    f1, acc = evaluate_model(model)
    print(f"{name} - F1 Score: {f1}, Accuracy: {acc}")
```

```
Logistic Regression - F1 Score: 0.7172505907808155, Accuracy: 0.7782404997397189
Random Forest - F1 Score: 0.7829679174964488, Accuracy: 0.8105153565851119
```

```
# Make predictions
predictions = lr_model.transform(test_data)
predictions.select("Exited", "prediction", "probability").show(10)
```

```
+------+----------+--------------------+
|Exited|prediction|         probability|
+------+----------+--------------------+
|     1|       0.0|[0.70889640764895...|
|     0|       0.0|[0.77926941422053...|
|     0|       0.0|[0.70819241617441...|
|     0|       0.0|[0.93615992557578...|
|     0|       0.0|[0.94707654641019...|
|     0|       0.0|[0.82046485628250...|
|     0|       0.0|[0.90661028865742...|
|     1|       0.0|[0.71245275549516...|
|     0|       0.0|[0.85380344206407...|
|     1|       0.0|[0.88887603342805...|
+------+----------+--------------------+
only showing top 10 rows
```

```
spark.stop()
```

```
Start coding or generate with AI.
```