Churn Prediction Analysis Steps

Step 1: Import necessary libraries

```
scala> import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.SparkSession

scala> import org.apache.spark.sql.functions._
import org.apache.spark.sql.functions._

scala> import org.apache.spark.ml.feature.{StringIndexer, VectorAssembler}
import org.apache.spark.ml.feature.{StringIndexer, VectorAssembler}

scala> import org.apache.spark.ml.classification.RandomForestClassifier
import org.apache.spark.ml.classification.RandomForestClassifier

scala> import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.Pipeline

scala> import org.apache.spark.ml.linalg.Vector
import org.apache.spark.ml.linalg.Vector

scala> import org.apache.spark.sql.functions.udf
import org.apache.spark.sql.functions.udf

scala> import org.apache.spark.ml.evaluation.BinaryClassificationEvaluator
import org.apache.spark.ml.evaluation.BinaryClassificationEvaluator

scala> import
org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator

scala> import org.apache.spark.ml.tuning.{ParamGridBuilder,
CrossValidator}
import org.apache.spark.ml.tuning.{ParamGridBuilder, CrossValidator}
```

Step 2: Initialize Spark session

```
scala> val spark = SparkSession.builder()
     .appName("TelcoChurnPrediction")
     .getOrCreate()
spark: org.apache.spark.sql.SparkSession =
org.apache.spark.sql.SparkSession@11bd85254
```

Step 3: Load data from Parquet

```
scala> val analyzedData = spark.read.parquet
("hdfs:///analyzed_telco_reviews/")
analyzedData: org.apache.spark.sql.DataFrame = [Username: string, Date:
string ... 16 more fields]
```

Step 4: Create churn risk and telco index

```
scala> val churnRisk = analyzedData.withColumn("ChurnRisk",
when(col("Rating") <= 2 && col("SentimentScore") < 0, 1.0).otherwise(0.0))
churnRisk: org.apache.spark.sql.DataFrame = [Username: string, Date:
string ... 17 more fields]

scala> val telcoIndexer = new StringIndexer()
     .setInputCol("Telco")
     .setOutputCol("TelcoIndex")
telcoIndexer: org.apache.spark.ml.feature.StringIndexer =
strIdx_ae677b12fd56
```

Step 5: Create VectorAssembler

```
scala> val assembler = new VectorAssembler()
     .setInputCols(Array("ReviewLength", "SentimentScore", "ResponseTime",
"TelcoIndex"))
     .setOutputCol("Features")
assembler: org.apache.spark.ml.feature.VectorAssembler =
vecAssembler_8e1498c4f2a
```

Step 6: Random forest model

```
scala> val randomForest = new RandomForestClassifier()
     .setLabelCol("ChurnRisk")
     .setFeaturesCol("Features")
     .setProbabilityCol("RawProbability")
     .setNumTrees(50)
randomForest: org.apache.spark.ml.classification.RandomForestClassifier =
rfc_5220993f5914
```

Step 7: Pipeline

```
scala> val pipeline = new Pipeline()
     .setStages(Array(telcoIndexer, assembler, randomForest))
pipeline: org.apache.spark.ml.Pipeline = pipeline_0d153371aeca
```

Step 8: Cast columns to numeric types

```
scala> val dataWithNumericFeatures = churnRisk
     .withColumn("ReviewLength", col("ReviewLength").cast("double"))
     .withColumn("ResponseTime", col("ResponseTime").cast("double"))
dataWithNumericFeatures: org.apache.spark.sql.DataFrame = [Username:
string, Date: string ... 17 more fields]
```

Step 9: Split data for train and test model

```
scala> val Array(train, test) =
dataWithNumericFeatures.randomSplit(Array(0.8, 0.2), seed = 42)
train: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Username:
string, Date: string ... 17 more fields]
test: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Username:
string, Date: string ... 17 more fields]
```

Step 10: Train model

```
scala> val model = pipeline.fit(train)
model: org.apache.spark.ml.PipelineModel = pipeline_0d153371aeca
```

Step 11: Make predictions with test

```
scala> val predictions = model.transform(test)
predictions: org.apache.spark.sql.DataFrame = [Username: string, Date:
string ... 22 more fields]
```

Step 12: UDF to extract churn probability

```
scala> val churnProb = udf((probability: Vector) => probability(1))
churnProb: org.apache.spark.sql.expressions.UserDefinedFunction =
SparkUserDefinedFunction(<function1>,DoubleType,Some(List(org.apache.spark
.ml.linalg.VectorUDT@3bfc3ba7)))
```

Step 13: Add ChurnProbability column

```
scala> val churnPredictions = predictions.withColumn("ChurnProbability",
churnProb(col("RamProbability")))
churnPredictions: org.apache.spark.sql.DataFrame = [Username: string,
Date: string ... 23 more fields]
```

Step 14: Show results

```
scala> churnPredictions .select("UserId", "TelcoIndex", "SentimentScore",
"ChurnRisk", "Features", "RamProbability", "Prediction",
"ChurnProbability").show()
```

Step 15: Save new predictions file to CSV and Parquet.

```
scala> churnPredictions
     .select("UserId", "Telco", "Date", "Day", "Sentiment", "Review",
"Rating", "ReviewLength", "ReviewCategory", "SentimentScore", "ChurnRisk",
"ChurnProbability")
     .write.option("header",
"true").csv("hdfs:///output/predicted_telco_reviews.csv")

scala> churnPredictions
     .write.mode("overwrite")
     .parquet("hdfs:///output/predicted_telco_reviews.parquet")
```