**Bài thực hành số 7.**

**Sử dụng thư viện ML**

**Link bài làm trên Databricks:** [Bài làm](https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/345358637122346/1160170899787476/5017236833160125/latest.html)

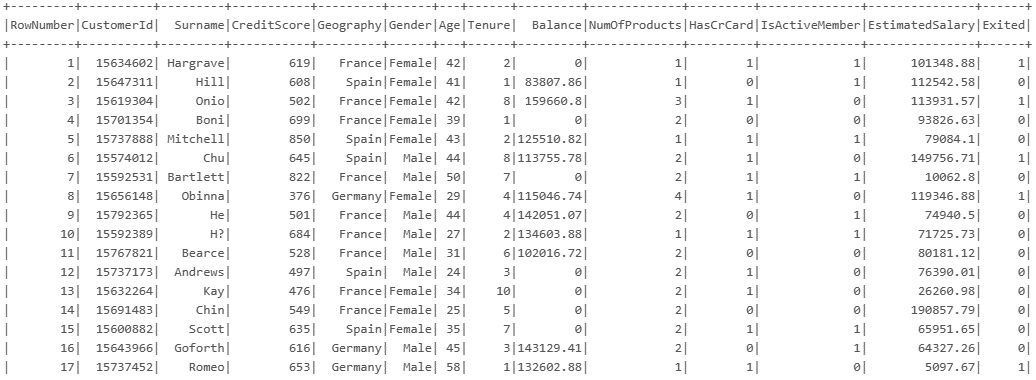
**I. Dự đoán khách hàng rời ngân hàng**

**a) Sử dụng PySpark để đọc dữ liệu vào DataFrame.**

df = spark.read.option('header',

'true').csv('/FileStore/Bigdata\_task07/Churn\_Modelling.csv')

df.show()



**Chuyển kiểu dữ liệu**

from pyspark.sql.types import DoubleType, IntegerType

list\_col\_to\_transtype\_double = ['CreditScore', 'Balance', 'EstimatedSalary']

list\_col\_to\_transtype\_int = ['Age', 'Tenure', 'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'Exited']

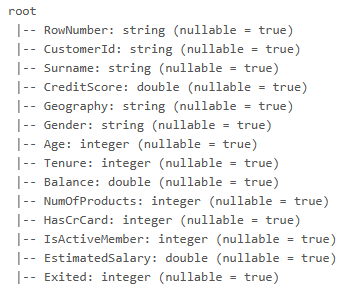
for c in list\_col\_to\_transtype\_double:

    df = df.withColumn(c, df[c].cast(DoubleType()))

for c in list\_col\_to\_transtype\_int:

    df = df.withColumn(c, df[c].cast(IntegerType()))

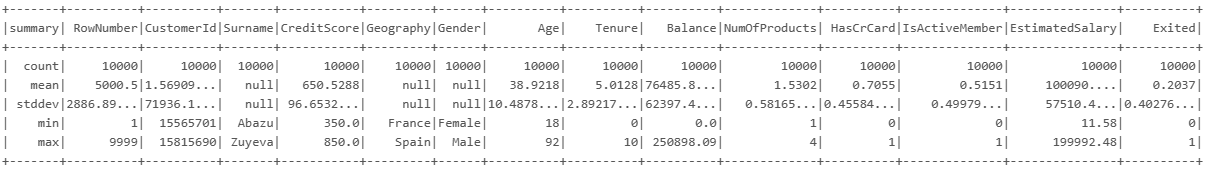
df.printSchema()



**b) Thực hiện một số thống kê, trực quan hóa để hiểu dữ liệu.**

**Thống kê cơ bản dữ liệu**

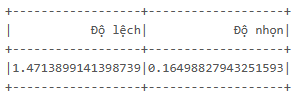
df.describe().show(truncate=10)

****

**Tính độ lệch và độ nhọn của trường Exited**

from pyspark.sql.functions import skewness, kurtosis

df.select(skewness('Exited').alias('Độ lệch'), kurtosis('Exited').alias('Độ nhọn')).show()

****

**Dùng ma trận tương quan để trực quan hóa dữ liệu**

import matplotlib.pyplot as plt

import seaborn as sns

df\_pandas = df.toPandas()

correlation\_matrix = df\_pandas.corr()

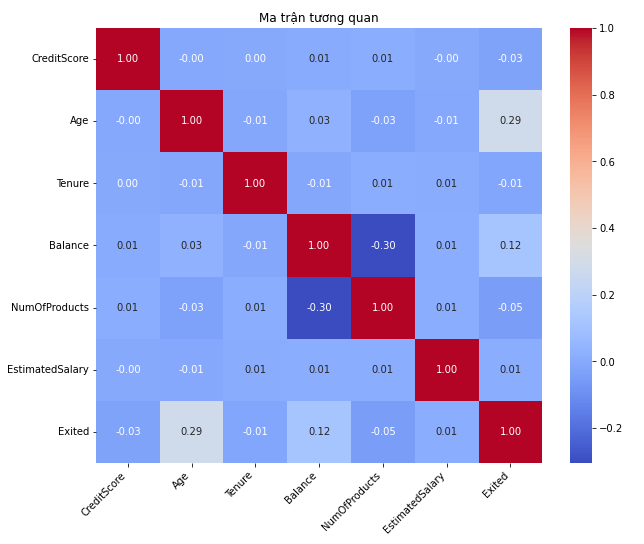
plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt=".2f")

plt.title("Ma trận tương quan")

plt.xticks(rotation=45, ha='right')

plt.show()

****

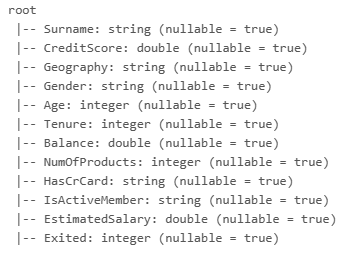
**c) Tiền xử lý dữ liệu, bao gồm: loại bỏ cột RowNumber, CustomerID, chuyển đổi giá**

**trị chuỗi thành số, chuyển đổi các biến độc lập thành vector.**

**Loại bỏ cột RowNumber và CustomerID**

df\_drop = df.drop('RowNumber').drop('CustomerID')

df\_drop.printSchema()

****

**Chuyển đổi giá trị chuỗi thành số (cho hai cột Geography và Gender)**

from pyspark.ml.feature import StringIndexer

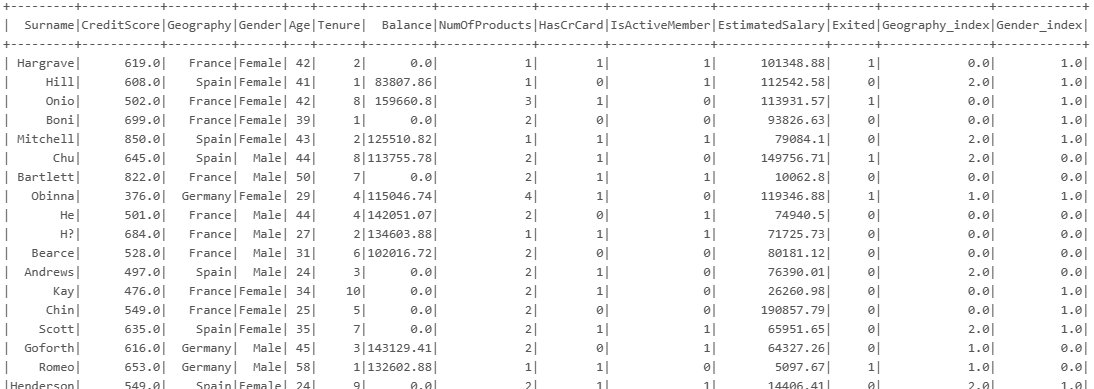
inputs = ['Geography', 'Gender']

outputs = ['Geography\_index', 'Gender\_index']

stringIndexer = StringIndexer(inputCols=inputs, outputCols=outputs)

df\_indexed = stringIndexer.fit(df\_drop).transform(df\_drop)

df\_indexed.show()

****

**Chuyển đổi các biến độc lập thành vector**

from pyspark.ml.feature import VectorAssembler

col\_assembler =  ["CreditScore"] + df\_indexed.columns[-10:]

assembler = VectorAssembler(inputCols=col\_assembler, outputCol='features')

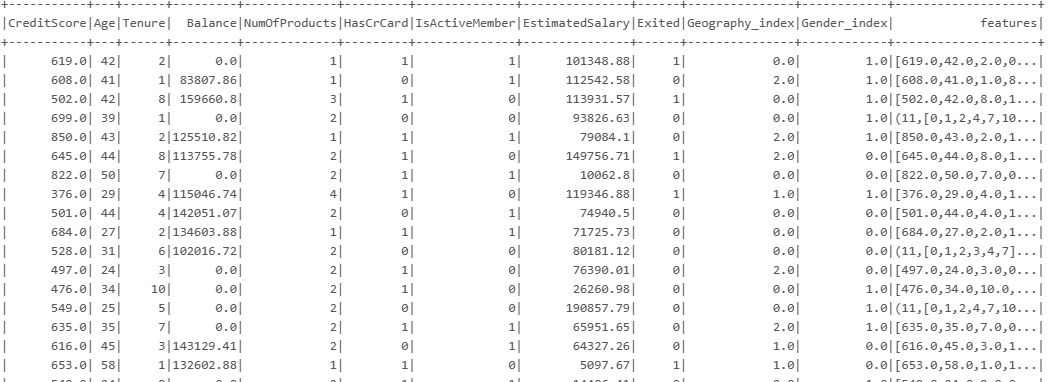
df\_assembler = assembler.transform(df\_indexed)

# Lọc lấy các cột cần thiết trước khi train mô hình

col\_filter\_to\_train\_model = col\_assembler + ['features']

data = df\_assembler.select(col\_filter\_to\_train\_model)

data.show()

****

**d) Chia dữ liệu thành tập huấn luyện và tập kiểm tra với tỉ lệ 70/30.**

data\_train, data\_test = data.randomSplit([0.7, 0.3], seed=666)

print('train: ', data\_train.count())

print('test: ', data\_test.count())

****

**e) Sử dụng Logistic Regression để huấn luyện mô hình dự đoán khách hàng có rời khỏi ngân hàng không?**

from pyspark.ml.classification import LogisticRegression

lr = LogisticRegression(featuresCol='features', labelCol='Exited')

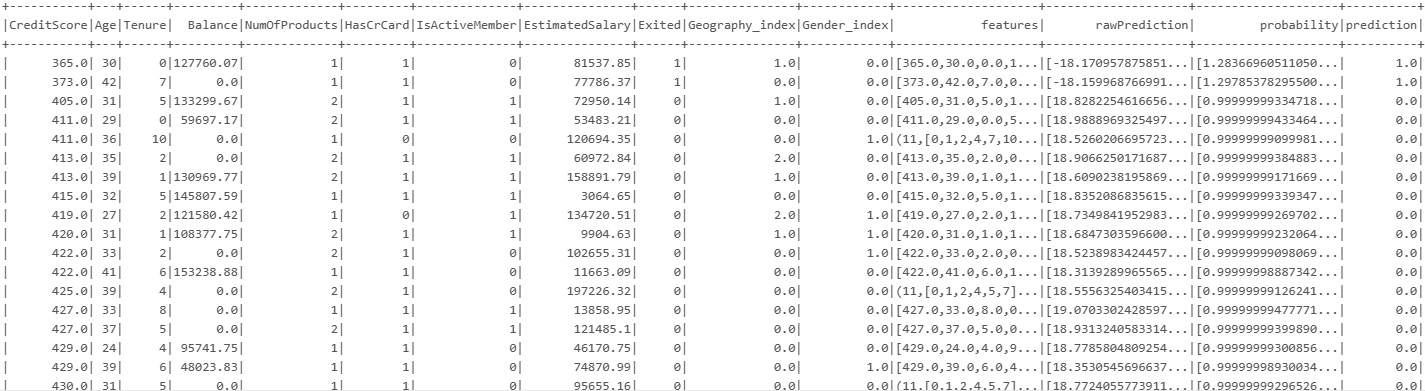
model = lr.fit(data\_train)

**f) Đánh giá hiệu suất của mô hình trên tập kiểm tra bằng độ chính xác, độ phủ, độ chính xác cân bằng, F1 score và AUC.**

**Dùng mô hình để dự đoán trên tập dữ liệu data\_test**

test\_model = model.transform(data\_test)

test\_model.show()

****

**Kiểm tra bằng độ chính xác của mô hình:**

from pyspark.ml.evaluation import BinaryClassificationEvaluator

from pyspark.ml.evaluation import MulticlassClassificationEvaluator

evaluator = BinaryClassificationEvaluator(labelCol="Exited", rawPredictionCol="probability")

accuracy = evaluator.evaluate(test\_model)

print('Độ chính xác của mô hình: ', accuracy)

****

**Kiểm tra bằng độ phủ:**

evaluator\_recall = MulticlassClassificationEvaluator(labelCol="Exited",

predictionCol="prediction", metricName="weightedRecall")

recall = evaluator\_recall.evaluate(test\_model)

print("Độ phủ:", recall)

****

**Kiểm tra bằng F1 Score:**

evaluator\_f1 = MulticlassClassificationEvaluator(labelCol="Exited", predictionCol="prediction",

metricName="f1")

f1\_score = evaluator\_f1.evaluate(test\_model)

print('F1 Score: ', f1\_score)

****

**Kiểm tra bằng AUC:**

evaluator\_auc = BinaryClassificationEvaluator(labelCol="Exited", rawPredictionCol="rawPrediction",

metricName="areaUnderROC")

auc = evaluator\_auc.evaluate(test\_model)

print("AUC:", auc)

****

**g) Tạo pipeline để xây dựng mô hình từ bước chuẩn hóa dữ liệu đến chọn mô hình học máy. Sử dụng pipeline để huấn luyện mô hình từ dữ liệu huấn luyện.**

**Tạo pipeline để xây dựng mô hình từ bước chuẩn hóa dữ liệu đến chọn mô hình học máy**

from pyspark.ml import Pipeline

pipeline = Pipeline(stages=[assembler, lr])

**Sử dụng pipeline để huấn luyện mô hình từ dữ liệu huấn luyện.**

data\_train2, data\_test2 = df\_indexed.randomSplit([0.7, 0.3], seed=666)

model2 = pipeline.fit(data\_train2)

print(model2)

****

**h) Lưu mô hình xây dựng bao gồm dữ liệu đã huấn luyện.**

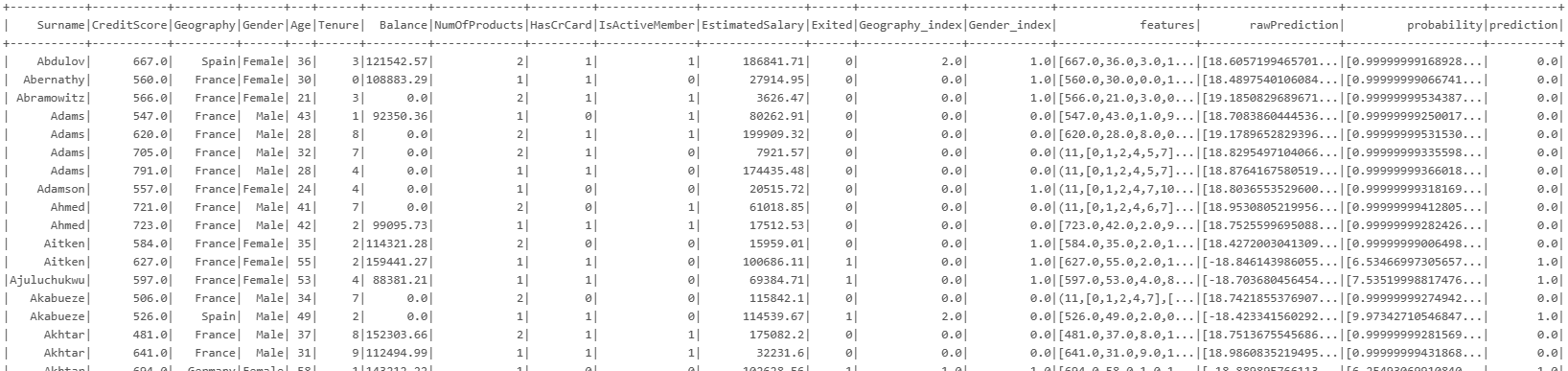
pipelinePath = '/FileStore/Model'

pipeline.write().overwrite().save(pipelinePath)

**i) Mở lại mô hình đã lưu và dự đoán lại cho dữ liệu test.**

loadedPipeline = Pipeline.load(pipelinePath)

loadedPipeline.fit(data\_train2).transform(data\_test2).show()

****

**II. Lựa chọn thuộc tính xây dựng mô hình**

**a) Từ việc hiểu dữ liệu, phân tích độ tương quan giữa các thuộc tính hãy chọn các thuộc tính cần thiết cho việc xây dựng mô hình.**

**Phân tích độ tương quan giữa các thuộc tính với cột Exited**

data\_view = data.drop('features').toPandas()

correlation\_matrix = data\_view.corr()

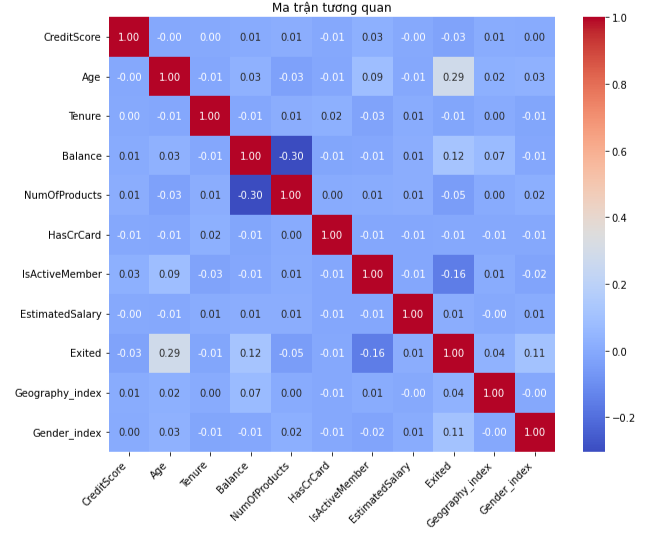
plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt=".2f")

plt.title("Ma trận tương quan")

plt.xticks(rotation=45, ha='right')

plt.show()

****

**🡪**  Từ ma trận trên có thể chọn những cột tương quan mạnh với cột Exited như sau:

Age, Balance, IsActiveMember, Gender

**Xây dựng mô hình**

df\_sec = data.select('Age', 'Balance', 'IsActiveMember', 'Gender\_index', 'Exited')

df\_sec.show()

assembler\_sec = VectorAssembler(inputCols=['Age', 'Balance', 'IsActiveMember', 'Gender\_index'], outputCol='Features')

df\_asembler\_sec = assembler\_sec.transform(df\_sec)

lr\_sec = LogisticRegression(featuresCol='Features', labelCol='Exited')

data\_train\_sec, data\_test\_sec = df\_asembler\_sec.randomSplit([0.7, 0.3], seed=666)

model\_sec = lr\_sec.fit(data\_train\_sec)

**Dùng mô hình để dự đoán**

test\_model\_sec = model\_sec.transform(data\_test\_sec)

**Đánh giá mô hình**

accuracy = evaluator.evaluate(test\_model\_sec)

print('Độ chính xác của mô hình: ', accuracy)

****

**🡺 Đánh giá:** Từ độ chính xác của mô hình có thể thấy việc lựa chọn các thuộc tính chưa chặt chẽ, độ chính xác của mô hình thấp hơn.

**III. Thay đổi tham số của mô hình**

**Thay đổi tham số của mô hình LogisticRegression như: maxIter, regParam rồi so sánh**

**với kết quả ban đầu.**

lr = LogisticRegression(featuresCol='features', labelCol='Exited', maxIter=10, regParam=0.05)

model = lr.fit(data\_train)

test\_model = model.transform(data\_test)

**Đánh giá độ chính xác của mô hình so với kết quả ban đầu.**

accuracy3 = evaluator.evaluate(test\_model)

print('Độ chính xác mô hình ban đầu: ', accuracy)

print('Độ chính xác của mô hình sau khi thay đổi các tham số: ', accuracy3)

****

**🡺 So với kết quả ban đầu:** Độ chính xác không thay đổi