**Bài thực hành số 7.**

**Sử dụng thư viện ML**

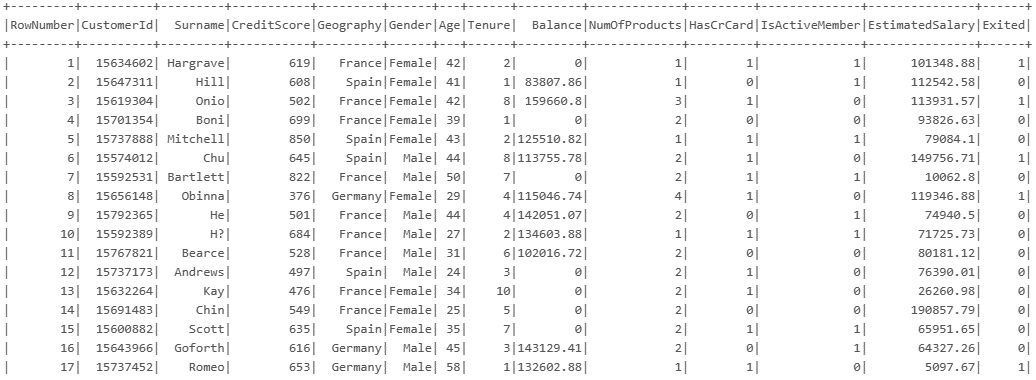
**Link bài làm trên Databricks:** [Bài làm](https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/345358637122346/1160170899787476/5017236833160125/latest.html)

**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']

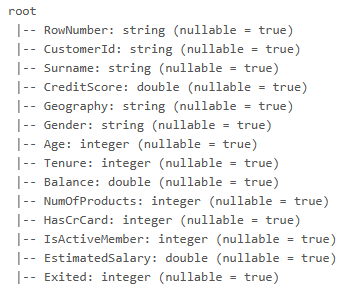
for c in list\_col\_to\_transtype\_double:

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

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

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

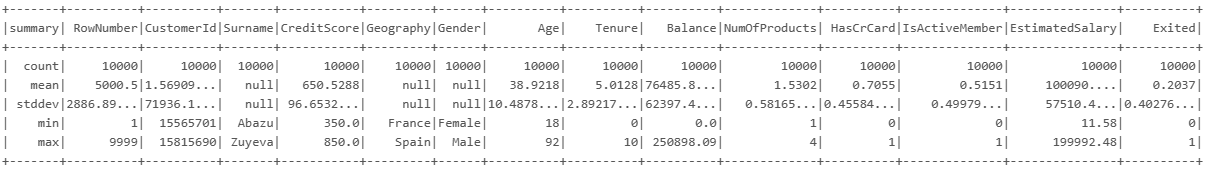
df.printSchema()



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

**Thống kê cơ bản dữ liệu**

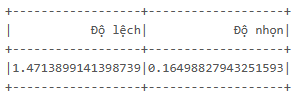
df.describe().show(truncate=10)

****

**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()

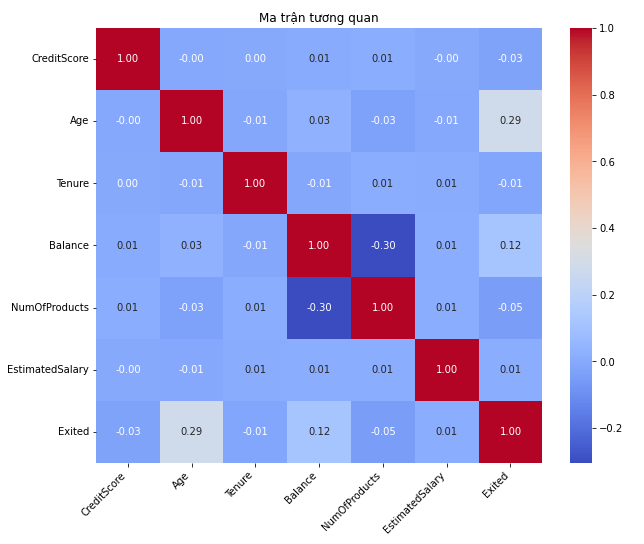
plt.figure(figsize=(10, 8))

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

plt.title("Ma trận tương quan")

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

plt.show()

****

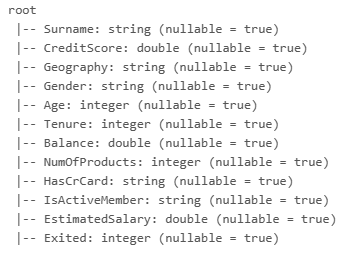
**c) Tiền xử lý dữ liệu, bao gồm: loại bỏ cột RowNumber, CustomerID, chuyển đổi giá**

**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

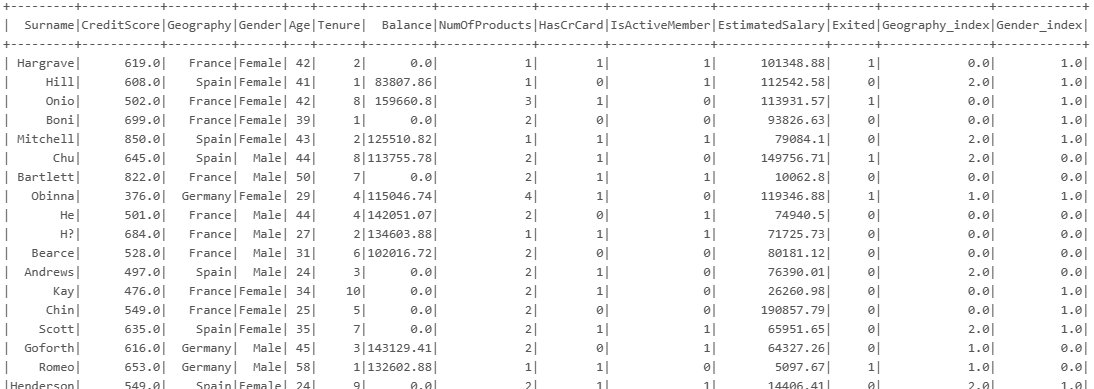
inputs = ['Geography', 'Gender']

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')

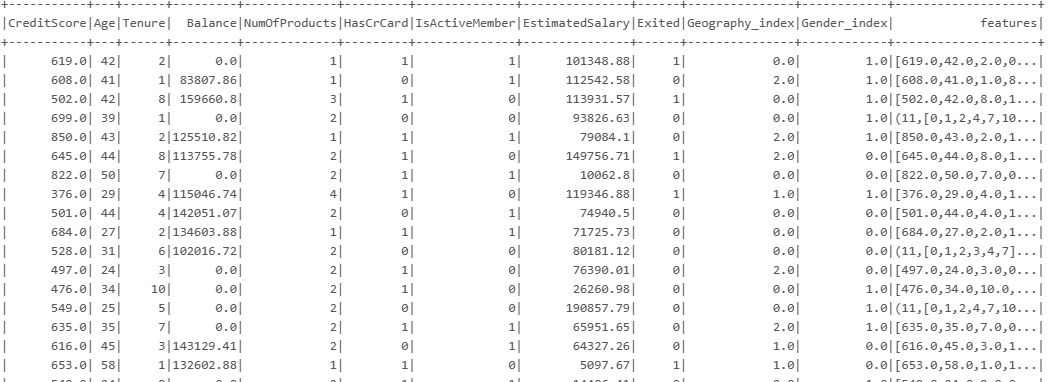
df\_assembler = assembler.transform(df\_indexed)

# 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')

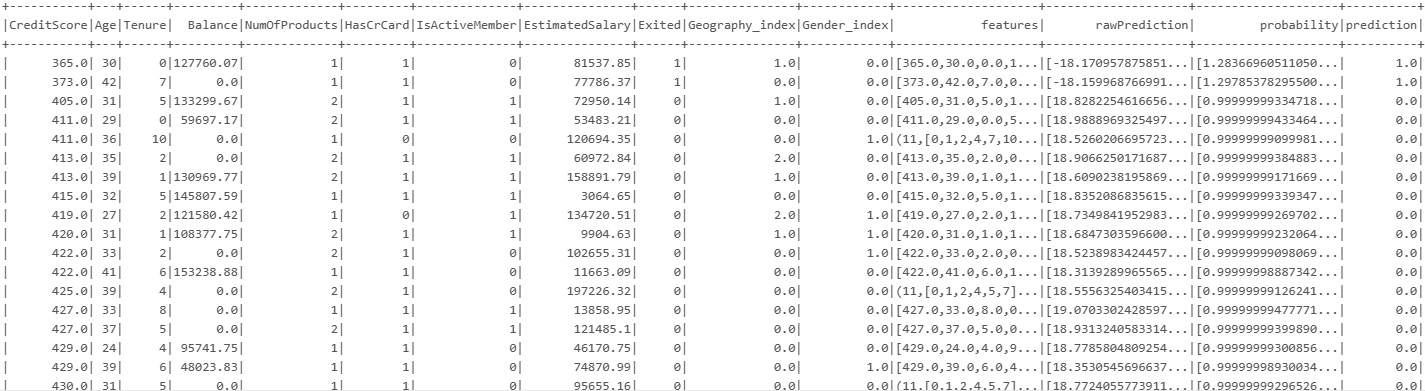
model = lr.fit(data\_train)

**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)

****