BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO CUỐI KÌ

Môn học: BDA Machine Learning at Scale

Đề tài: Customer Classification

GVHD:	ThS. Quách Đình Hoàng	
SVTH1:	Trần Trọng Nghĩa	(17133040)
SVTH2:	Vũ Anh Thái Dương	(17133010)
SVTH3:	Nguyễn Quang Nhật	(17133043)
SVTH4:	Đỗ Viết Hải	(17133017)

THÀNH PHỐ HỒ CHÍ MINH, 1/2021

MỤC LỤC

TỔNG QUAN ĐỀ TÀI

Mô tả đề tài

Homesite, nhà cung cấp hàng đầu về bảo hiểm nhà ở. Homesite muốn tìm ra với mức thu nhập và mức giá bảo hiểm như thế nào thì khách hàng sẽ mua bảo hiểm nhà ở của họ. Từ đó điều chỉnh mức giá niêm yết bảo hiểm phù hợp để gia tăng doanh số. Sử dụng cơ sở dữ liệu ẩn danh thông tin về khách hàng và hoạt động bán bảo hiểm của công ty, bao gồm thông tin về giá trị tài sản và mức giá bảo hiểm. Tập dữ liệu có thể được tìm thấy tại đây (https://www.kaggle.com/c/homesite-quote-conversion)

Mô tả tập dữ liệu

Tập train

Tập dữ liệu mô tả về lượng lớn khách hàng quan tâm việc mua bảo hiểm nhà ở của Homesite. Tập dữ liệu gồm 299 cột:

• Một cột QuoteNumber: Khách hàng tiềm năng.

• Một cột QuoteConversion_Flag: Cho biết liệu khách hàng có mua gói bảo hiểm nhà ở của Homesite (0 là mua 1 là không mua).

• Các cột còn lại chứa các thông tin cụ thể về thông tin khác hàng như: Thông tin bảo hiểm cụ thể (specific coverage information), thông tin bán hàng (sales information), thông tin cá nhân (personal information), thông tin tài sản (property information) và thông tin địa chỉ (geographic information).

Tập train có 260753 dòng, mỗi dòng tương ứng với một khách hàng. Tỷ lệ khách hàng mua bảo hiểm chiếm 81.2% và khách hàng không mua chiếm 18,8%.

Tập test

Tập test chứa thông tin tương tự như tập train nhưng không có cột QuoteConversion_Flag. Tập test có dòng 173836 dòng. Nhiệm vụ của chúng tôi là dự đoán QuoteConversion_Flag cho mỗi QuoteNumber trong tập test.

Thuật toán

Nhận thấy đây là bài toán phân loại nhị phân (Binary Classification) nên nhóm sẽ dùng các thuật toán như sau để thực hiện bài toán phân loại sử dụng thư viện pyspark.

• Logistic Regression: Sử dụng hàm logit để dự đoán xác suất.

• Decision Tree: Tìm biến độc lập quan trọng nhất để tạo nhóm. Decision tree là tên đại diện cho một nhóm thuật toán phát triển dựa trên decision tree. Ở đó, mỗi node của cây sẽ là các thuộc tính, và các nhánh là giá trị lựa chọn của thuộc tính đó. Bằng cách đi theo các giá trị thuộc tính trên cây, decision tree sẽ cho ta biết giá trị dự đoán.

• Random Forest: Thuật toán này xây dựng nhiều cây quyết định và hợp nhất chúng lại với nhau.

Đánh giá mô hình

Gini

Gini là tỷ số giữa đường cong ROC và đường $diagnolline$ & diện tích của tam giác trên (the area of the above triangle). Vì vậy, chúng ta có thể tính toán $Gini$ theo công thức sau:

$$Gini = 2 * AUC - 1$$

Trong đó, AUC (Area Under the Curve) là diện tích nằm dưới ROC curve. ROC là một đường cong biểu diễn xác suất. Chỉ số AUC càng gần 1 thì mô hình càng phân loại chính xác. AUC càng gần 0.5 thì hiệu suất phân loại càng tệ còn nếu gần 0 thì mô hình sẽ phân loại ngược kết quả (phân loại dương tính thành âm tính và ngược lại). Ví dụ , $Gini$ đạt giá trị trên 0.6 thì mô hình tốt.

Confusion Matrix

Confusion Matrix là một bảng được sử dụng để mô tả hiệu suất của một mô hình phân loại. Một số định nghĩa:

a. Accuracy: Tỷ lệ tổng số dự đoán đúng.

b. Precision (Positive Predictive Value): Tỷ lệ các trường hợp dương tính được xác định đúng.

$$Precision = \frac{TP}{TP + FP}$$

c. Negative Predictive Value: Tỷ lệ các trường hợp âm tính được xác định đúng.

$$NegativePredictiveValue = \frac{TN}{FN + TN}$$

d. Sensitivity (Recall): Tỷ lệ các trường hợp dương tính thực tế được xác định đúng.

$$Recall = \frac{TP}{TP + FN}$$

e. Specificity : Tỷ lệ các trường hợp âm tính thực tế được xác định đúng.

$$Specificity = \frac{TN}{FP + TN}$$

TIỀN XỬ LÝ DỮ LIỆU

Khởi tạo, import thư viện, load dữ liệu vào Spark DataFrame

Khởi tạo PySpark

In [1]:
```
import findspark
findspark.init()

import pyspark

from pyspark import SparkContext
sc = SparkContext.getOrCreate()

#Spark DataFrame
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
```

Import thư viện cần thiết

In [2]:
```
from pyspark.sql.types import IntegerType, StringType, DoubleType, ShortType,
DecimalType
import pyspark.sql.functions as func
from pyspark.sql.functions import isnull
from pyspark.sql.functions import isnan, when, count, col, round
from pyspark.sql.functions import mean
from pyspark.sql.types import Row
import matplotlib.pyplot as plt
from pyspark.sql.functions import udf


# Pandas DF operation
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from numpy import array

# Model và Evaluation (đánh giá)
from pyspark.ml.feature import VectorAssembler, VectorIndexer, OneHotEncoder,
StringIndexer
from pyspark.sql.functions import when
from pyspark.sql import functions as F
from pyspark.sql.functions import avg
from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression, RandomForestClassifi
er, GBTClassifier
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.evaluation import BinaryClassificationEvaluator, MulticlassCla
ssificationEvaluator
from pyspark.mllib.evaluation import BinaryClassificationMetrics
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import log_loss
from pyspark.sql import Window
from pyspark.sql.functions import rank,sum,col
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorSlicer

window = Window.rowsBetween(Window.unboundedPreceding,Window.unboundedFollowin
g)
```

Load dữ liệu vào Spark DataFrame

In [3]:
```
# Chọn đường dẫn đến tập data train
file_type = 'text'
path=r'train.csv'
delimeter=','
```

In [4]:
```
def load_data(file_type):
    """input type of file "text" or "parquet" and Return pyspark dataframe"""
    if file_type =="text": # đầu vào ở dạng text
        df = spark.read.option("header", "true") \
                       .option("delimeter",delimeter)\
                       .option("inferSchema", "true") \
                       .csv(path)
    else:
        df= spark.read.parquet("example.parquet")
    return df
```

In [5]:
```
df = load_data(file_type)
```

In [6]:
```
# Chọn đường dẫn đến tập data test
file_type = 'text'
path=r'test.csv'
delimeter=','
```

In [7]:
```
test_data = load_data(file_type)
```

In [8]:
```
help(load_data)
```

```
Help on function load_data in module __main__:

load_data(file_type)
    input type of file "text" or "parquet" and Return pyspark dataframe
```

Kiểm tra dữ liệu

Chúng tôi tiến hành kiểm tra thuộc tính của các cột trong dataframe và kiểm tra các cột nào có giá trị bị thiếu (null) trên tập train và tập test .

In [9]:
```
type(df)
```

Out[9]:
```
pyspark.sql.dataframe.DataFrame
```

In [10]:
```
type(test_data)
```

Out[10]:
```
pyspark.sql.dataframe.DataFrame
```

Do dữ liệu cũng khá dài nên chúng tôi chỉ hiển thị 1 dòng dữ liệu trong DataFrame.

In [11]:
```
# Thực hiện trên tập train
# df.show(1)
```

In [12]:
```
# Thực hiện tương tự trên tập test
#test_data.show(1)
```

In [13]:
```
# Kiểm tra thuộc tính của các cột trong DataFrame và kiểm tra các cột nào có g
iá trị bị thiếu (null)
# Kiểm tra trên tập train
len(df.columns), df.printSchema()
```

```
root
 |-- QuoteNumber: integer (nullable = true)
 |-- Original_Quote_Date: string (nullable = true)
 |-- QuoteConversion_Flag: integer (nullable = true)
 |-- Field6: string (nullable = true)
 |-- Field7: integer (nullable = true)
 |-- Field8: double (nullable = true)
 |-- Field9: double (nullable = true)
 |-- Field10: string (nullable = true)
 |-- Field11: double (nullable = true)
 |-- Field12: string (nullable = true)
 |-- CoverageField1A: integer (nullable = true)
 |-- CoverageField1B: integer (nullable = true)
 |-- CoverageField2A: integer (nullable = true)
 |-- CoverageField2B: integer (nullable = true)
 |-- CoverageField3A: integer (nullable = true)
 |-- CoverageField3B: integer (nullable = true)
 |-- CoverageField4A: integer (nullable = true)
 |-- CoverageField4B: integer (nullable = true)
 |-- CoverageField5A: integer (nullable = true)
 |-- CoverageField5B: integer (nullable = true)
 |-- CoverageField6A: integer (nullable = true)
 |-- CoverageField6B: integer (nullable = true)
 |-- CoverageField8: string (nullable = true)
 |-- CoverageField9: string (nullable = true)
 |-- CoverageField11A: integer (nullable = true)
 |-- CoverageField11B: integer (nullable = true)
 |-- SalesField1A: integer (nullable = true)
 |-- SalesField1B: integer (nullable = true)
 |-- SalesField2A: integer (nullable = true)
 |-- SalesField2B: integer (nullable = true)
 |-- SalesField3: integer (nullable = true)
 |-- SalesField4: integer (nullable = true)
 |-- SalesField5: integer (nullable = true)
 |-- SalesField6: integer (nullable = true)
 |-- SalesField7: string (nullable = true)
 |-- SalesField8: integer (nullable = true)
 |-- SalesField9: integer (nullable = true)
 |-- SalesField10: integer (nullable = true)
 |-- SalesField11: integer (nullable = true)
 |-- SalesField12: integer (nullable = true)
 |-- SalesField13: integer (nullable = true)
 |-- SalesField14: integer (nullable = true)
 |-- SalesField15: integer (nullable = true)
 |-- PersonalField1: integer (nullable = true)
 |-- PersonalField2: integer (nullable = true)
 |-- PersonalField4A: integer (nullable = true)
 |-- PersonalField4B: integer (nullable = true)
 |-- PersonalField5: integer (nullable = true)
 |-- PersonalField6: integer (nullable = true)
 |-- PersonalField7: string (nullable = true)
 |-- PersonalField8: integer (nullable = true)
 |-- PersonalField9: integer (nullable = true)
 |-- PersonalField10A: integer (nullable = true)
 |-- PersonalField10B: integer (nullable = true)
 |-- PersonalField11: integer (nullable = true)
 |-- PersonalField12: integer (nullable = true)
```

```
|-- PersonalField13: integer (nullable = true)
|-- PersonalField14: integer (nullable = true)
|-- PersonalField15: integer (nullable = true)
|-- PersonalField16: string (nullable = true)
|-- PersonalField17: string (nullable = true)
|-- PersonalField18: string (nullable = true)
|-- PersonalField19: string (nullable = true)
|-- PersonalField22: integer (nullable = true)
|-- PersonalField23: integer (nullable = true)
|-- PersonalField24: integer (nullable = true)
|-- PersonalField25: integer (nullable = true)
|-- PersonalField26: integer (nullable = true)
|-- PersonalField27: integer (nullable = true)
|-- PersonalField28: integer (nullable = true)
|-- PersonalField29: integer (nullable = true)
|-- PersonalField30: integer (nullable = true)
|-- PersonalField31: integer (nullable = true)
|-- PersonalField32: integer (nullable = true)
|-- PersonalField33: integer (nullable = true)
|-- PersonalField34: integer (nullable = true)
|-- PersonalField35: integer (nullable = true)
|-- PersonalField36: integer (nullable = true)
|-- PersonalField37: integer (nullable = true)
|-- PersonalField38: integer (nullable = true)
|-- PersonalField39: integer (nullable = true)
|-- PersonalField40: integer (nullable = true)
|-- PersonalField41: integer (nullable = true)
|-- PersonalField42: integer (nullable = true)
|-- PersonalField43: integer (nullable = true)
|-- PersonalField44: integer (nullable = true)
|-- PersonalField45: integer (nullable = true)
|-- PersonalField46: integer (nullable = true)
|-- PersonalField47: integer (nullable = true)
|-- PersonalField48: integer (nullable = true)
|-- PersonalField49: integer (nullable = true)
|-- PersonalField50: integer (nullable = true)
|-- PersonalField51: integer (nullable = true)
|-- PersonalField52: integer (nullable = true)
|-- PersonalField53: integer (nullable = true)
|-- PersonalField54: integer (nullable = true)
|-- PersonalField55: integer (nullable = true)
|-- PersonalField56: integer (nullable = true)
|-- PersonalField57: integer (nullable = true)
|-- PersonalField58: integer (nullable = true)
|-- PersonalField59: integer (nullable = true)
|-- PersonalField60: integer (nullable = true)
|-- PersonalField61: integer (nullable = true)
|-- PersonalField62: integer (nullable = true)
|-- PersonalField63: integer (nullable = true)
|-- PersonalField64: integer (nullable = true)
|-- PersonalField65: integer (nullable = true)
|-- PersonalField66: integer (nullable = true)
|-- PersonalField67: integer (nullable = true)
|-- PersonalField68: integer (nullable = true)
|-- PersonalField69: integer (nullable = true)
|-- PersonalField70: integer (nullable = true)
|-- PersonalField71: integer (nullable = true)
```

```
|-- PersonalField72: integer (nullable = true)
|-- PersonalField73: integer (nullable = true)
|-- PersonalField74: integer (nullable = true)
|-- PersonalField75: integer (nullable = true)
|-- PersonalField76: integer (nullable = true)
|-- PersonalField77: integer (nullable = true)
|-- PersonalField78: integer (nullable = true)
|-- PersonalField79: integer (nullable = true)
|-- PersonalField80: integer (nullable = true)
|-- PersonalField81: integer (nullable = true)
|-- PersonalField82: integer (nullable = true)
|-- PersonalField83: integer (nullable = true)
|-- PersonalField84: integer (nullable = true)
|-- PropertyField1A: integer (nullable = true)
|-- PropertyField1B: integer (nullable = true)
|-- PropertyField2A: integer (nullable = true)
|-- PropertyField2B: integer (nullable = true)
|-- PropertyField3: string (nullable = true)
|-- PropertyField4: string (nullable = true)
|-- PropertyField5: string (nullable = true)
|-- PropertyField6: integer (nullable = true)
|-- PropertyField7: string (nullable = true)
|-- PropertyField8: integer (nullable = true)
|-- PropertyField9: integer (nullable = true)
|-- PropertyField10: integer (nullable = true)
|-- PropertyField11A: integer (nullable = true)
|-- PropertyField11B: integer (nullable = true)
|-- PropertyField12: integer (nullable = true)
|-- PropertyField13: integer (nullable = true)
|-- PropertyField14: string (nullable = true)
|-- PropertyField15: integer (nullable = true)
|-- PropertyField16A: integer (nullable = true)
|-- PropertyField16B: integer (nullable = true)
|-- PropertyField17: integer (nullable = true)
|-- PropertyField18: integer (nullable = true)
|-- PropertyField19: integer (nullable = true)
|-- PropertyField20: integer (nullable = true)
|-- PropertyField21A: integer (nullable = true)
|-- PropertyField21B: integer (nullable = true)
|-- PropertyField22: integer (nullable = true)
|-- PropertyField23: integer (nullable = true)
|-- PropertyField24A: integer (nullable = true)
|-- PropertyField24B: integer (nullable = true)
|-- PropertyField25: double (nullable = true)
|-- PropertyField26A: integer (nullable = true)
|-- PropertyField26B: integer (nullable = true)
|-- PropertyField27: integer (nullable = true)
|-- PropertyField28: string (nullable = true)
|-- PropertyField29: integer (nullable = true)
|-- PropertyField30: string (nullable = true)
|-- PropertyField31: string (nullable = true)
|-- PropertyField32: string (nullable = true)
|-- PropertyField33: string (nullable = true)
|-- PropertyField34: string (nullable = true)
|-- PropertyField35: integer (nullable = true)
|-- PropertyField36: string (nullable = true)
|-- PropertyField37: string (nullable = true)
```

```
|-- PropertyField38: string (nullable = true)
|-- PropertyField39A: integer (nullable = true)
|-- PropertyField39B: integer (nullable = true)
|-- GeographicField1A: integer (nullable = true)
|-- GeographicField1B: integer (nullable = true)
|-- GeographicField2A: integer (nullable = true)
|-- GeographicField2B: integer (nullable = true)
|-- GeographicField3A: integer (nullable = true)
|-- GeographicField3B: integer (nullable = true)
|-- GeographicField4A: integer (nullable = true)
|-- GeographicField4B: integer (nullable = true)
|-- GeographicField5A: integer (nullable = true)
|-- GeographicField5B: integer (nullable = true)
|-- GeographicField6A: integer (nullable = true)
|-- GeographicField6B: integer (nullable = true)
|-- GeographicField7A: integer (nullable = true)
|-- GeographicField7B: integer (nullable = true)
|-- GeographicField8A: integer (nullable = true)
|-- GeographicField8B: integer (nullable = true)
|-- GeographicField9A: integer (nullable = true)
|-- GeographicField9B: integer (nullable = true)
|-- GeographicField10A: integer (nullable = true)
|-- GeographicField10B: integer (nullable = true)
|-- GeographicField11A: integer (nullable = true)
|-- GeographicField11B: integer (nullable = true)
|-- GeographicField12A: integer (nullable = true)
|-- GeographicField12B: integer (nullable = true)
|-- GeographicField13A: integer (nullable = true)
|-- GeographicField13B: integer (nullable = true)
|-- GeographicField14A: integer (nullable = true)
|-- GeographicField14B: integer (nullable = true)
|-- GeographicField15A: integer (nullable = true)
|-- GeographicField15B: integer (nullable = true)
|-- GeographicField16A: integer (nullable = true)
|-- GeographicField16B: integer (nullable = true)
|-- GeographicField17A: integer (nullable = true)
|-- GeographicField17B: integer (nullable = true)
|-- GeographicField18A: integer (nullable = true)
|-- GeographicField18B: integer (nullable = true)
|-- GeographicField19A: integer (nullable = true)
|-- GeographicField19B: integer (nullable = true)
|-- GeographicField20A: integer (nullable = true)
|-- GeographicField20B: integer (nullable = true)
|-- GeographicField21A: integer (nullable = true)
|-- GeographicField21B: integer (nullable = true)
|-- GeographicField22A: integer (nullable = true)
|-- GeographicField22B: integer (nullable = true)
|-- GeographicField23A: integer (nullable = true)
|-- GeographicField23B: integer (nullable = true)
|-- GeographicField24A: integer (nullable = true)
|-- GeographicField24B: integer (nullable = true)
|-- GeographicField25A: integer (nullable = true)
|-- GeographicField25B: integer (nullable = true)
|-- GeographicField26A: integer (nullable = true)
|-- GeographicField26B: integer (nullable = true)
|-- GeographicField27A: integer (nullable = true)
|-- GeographicField27B: integer (nullable = true)
```

```
|-- GeographicField28A: integer (nullable = true)
|-- GeographicField28B: integer (nullable = true)
|-- GeographicField29A: integer (nullable = true)
|-- GeographicField29B: integer (nullable = true)
|-- GeographicField30A: integer (nullable = true)
|-- GeographicField30B: integer (nullable = true)
|-- GeographicField31A: integer (nullable = true)
|-- GeographicField31B: integer (nullable = true)
|-- GeographicField32A: integer (nullable = true)
|-- GeographicField32B: integer (nullable = true)
|-- GeographicField33A: integer (nullable = true)
|-- GeographicField33B: integer (nullable = true)
|-- GeographicField34A: integer (nullable = true)
|-- GeographicField34B: integer (nullable = true)
|-- GeographicField35A: integer (nullable = true)
|-- GeographicField35B: integer (nullable = true)
|-- GeographicField36A: integer (nullable = true)
|-- GeographicField36B: integer (nullable = true)
|-- GeographicField37A: integer (nullable = true)
|-- GeographicField37B: integer (nullable = true)
|-- GeographicField38A: integer (nullable = true)
|-- GeographicField38B: integer (nullable = true)
|-- GeographicField39A: integer (nullable = true)
|-- GeographicField39B: integer (nullable = true)
|-- GeographicField40A: integer (nullable = true)
|-- GeographicField40B: integer (nullable = true)
|-- GeographicField41A: integer (nullable = true)
|-- GeographicField41B: integer (nullable = true)
|-- GeographicField42A: integer (nullable = true)
|-- GeographicField42B: integer (nullable = true)
|-- GeographicField43A: integer (nullable = true)
|-- GeographicField43B: integer (nullable = true)
|-- GeographicField44A: integer (nullable = true)
|-- GeographicField44B: integer (nullable = true)
|-- GeographicField45A: integer (nullable = true)
|-- GeographicField45B: integer (nullable = true)
|-- GeographicField46A: integer (nullable = true)
|-- GeographicField46B: integer (nullable = true)
|-- GeographicField47A: integer (nullable = true)
|-- GeographicField47B: integer (nullable = true)
|-- GeographicField48A: integer (nullable = true)
|-- GeographicField48B: integer (nullable = true)
|-- GeographicField49A: integer (nullable = true)
|-- GeographicField49B: integer (nullable = true)
|-- GeographicField50A: integer (nullable = true)
|-- GeographicField50B: integer (nullable = true)
|-- GeographicField51A: integer (nullable = true)
|-- GeographicField51B: integer (nullable = true)
|-- GeographicField52A: integer (nullable = true)
|-- GeographicField52B: integer (nullable = true)
|-- GeographicField53A: integer (nullable = true)
|-- GeographicField53B: integer (nullable = true)
|-- GeographicField54A: integer (nullable = true)
|-- GeographicField54B: integer (nullable = true)
|-- GeographicField55A: integer (nullable = true)
|-- GeographicField55B: integer (nullable = true)
|-- GeographicField56A: integer (nullable = true)
```

```
 |-- GeographicField56B: integer (nullable = true)
 |-- GeographicField57A: integer (nullable = true)
 |-- GeographicField57B: integer (nullable = true)
 |-- GeographicField58A: integer (nullable = true)
 |-- GeographicField58B: integer (nullable = true)
 |-- GeographicField59A: integer (nullable = true)
 |-- GeographicField59B: integer (nullable = true)
 |-- GeographicField60A: integer (nullable = true)
 |-- GeographicField60B: integer (nullable = true)
 |-- GeographicField61A: integer (nullable = true)
 |-- GeographicField61B: integer (nullable = true)
 |-- GeographicField62A: integer (nullable = true)
 |-- GeographicField62B: integer (nullable = true)
 |-- GeographicField63: string (nullable = true)
 |-- GeographicField64: string (nullable = true)
```

Out[13]: (299, None)

Do các khá là nhiều cột dữ liệu nên nếu các bạn muốn chạy để kiểm tra trên tập test thì các bạn mở comment bên dưới ra.

In [14]:
```
# Tương tự chúng tôi kiểm tra trên tập test
#test_data.printSchema(), len (test_data.columns)
```

Đổi tên các cột ‘QuoteConversion_Flag’ thành ‘label’ và 'QuoteNumber' thành 'Id' để thuận tiện cho việc xây dựng mô hình.

In [15]:
```
# Đổi tên cột 'QuoteConversion_Flag' thành 'label' trên tập train
df = df.withColumnRenamed('QuoteConversion_Flag','label')
# Đổi tên cột 'QuoteNumber' thành 'Id' trên tập train
df = df.withColumnRenamed('QuoteNumber','Id')
```

In [16]:
```
# Đổi tên cột 'QuoteNumber' thành 'Id' trên tập test
test_data = test_data.withColumnRenamed('QuoteNumber','Id')
```

In [17]:
```
# Kiểm tra lại các thuộc tính của tập train
#len(df.columns), df.printSchema()
```

Dữ liệu ngày tháng không liên quan đến việc phân loại nên chúng tôi sẽ xóa cột này khỏi các tập dữ liệu.

In [18]:
```
# xóa cột ngày tháng Original_Quote_Date từ tập train
df_final=df.drop('Original_Quote_Date')

# xóa cột ngày tháng Original_Quote_Date từ tập test
test_data=test_data.drop('Original_Quote_Date')
```

Thống kê các khách hàng đã mua bảo hiểm (0) và không mua bảo hiểm (1) trong tập train. Ở tập train có đến 81.2% khách hàng là mua bảo hiểm và 18.8% khách hàng là không mua bảo hiểm.

In [19]:
```
target_percent=df_final.groupBy('label').count().sort(col("count").desc())\
                       .withColumn('total',sum(col('count')).over(window))\
                       .withColumn('Percent',col('count')*100/col('total'))
```

In [20]:
```
target_percent.show()
```

```
+-----+------+------+-----------------+
|label| count| total|          Percent|
+-----+------+------+-----------------+
|    0|211859|260753|81.24892139304245|
|    1| 48894|260753|18.75107860695754|
+-----+------+------+-----------------+
```

Xác định các categorical variable (biến phân loại) và nummerical variable (biến số)

Categorical variable là các biến có chứa giá trị string chứ không phải là giá trị số.

Numerical variable là các biến chứa giá trị số (integer và double).

Xác định các categorical variable (biến phân loại) và nummerical variable (biến số) trong tập train

In [21]:

```
# Xác định Categorical variable
cat_cols = [item[0] for item in df_final.dtypes if item[1].startswith('string'
)]
print("cat_cols:", cat_cols)

# Xác định Numerical variable
num_cols = [item[0] for item in df_final.dtypes if item[1].startswith('int') |
item[1].startswith('double')]
print("num_cols:", num_cols)
```

```
cat_cols: ['Field6', 'Field10', 'Field12', 'CoverageField8', 'CoverageField
9', 'SalesField7', 'PersonalField7', 'PersonalField16', 'PersonalField17', 'P
ersonalField18', 'PersonalField19', 'PropertyField3', 'PropertyField4', 'Prop
ertyField5', 'PropertyField7', 'PropertyField14', 'PropertyField28', 'Propert
yField30', 'PropertyField31', 'PropertyField32', 'PropertyField33', 'Property
Field34', 'PropertyField36', 'PropertyField37', 'PropertyField38', 'Geographi
cField63', 'GeographicField64']
num_cols: ['Id', 'label', 'Field7', 'Field8', 'Field9', 'Field11', 'CoverageF
ield1A', 'CoverageField1B', 'CoverageField2A', 'CoverageField2B', 'CoverageFi
eld3A', 'CoverageField3B', 'CoverageField4A', 'CoverageField4B', 'CoverageFie
ld5A', 'CoverageField5B', 'CoverageField6A', 'CoverageField6B', 'CoverageFiel
d11A', 'CoverageField11B', 'SalesField1A', 'SalesField1B', 'SalesField2A', 'S
alesField2B', 'SalesField3', 'SalesField4', 'SalesField5', 'SalesField6', 'Sa
lesField8', 'SalesField9', 'SalesField10', 'SalesField11', 'SalesField12', 'S
alesField13', 'SalesField14', 'SalesField15', 'PersonalField1', 'PersonalFiel
d2', 'PersonalField4A', 'PersonalField4B', 'PersonalField5', 'PersonalField
6', 'PersonalField8', 'PersonalField9', 'PersonalField10A', 'PersonalField10
B', 'PersonalField11', 'PersonalField12', 'PersonalField13', 'PersonalField1
4', 'PersonalField15', 'PersonalField22', 'PersonalField23', 'PersonalField2
4', 'PersonalField25', 'PersonalField26', 'PersonalField27', 'PersonalField2
8', 'PersonalField29', 'PersonalField30', 'PersonalField31', 'PersonalField3
2', 'PersonalField33', 'PersonalField34', 'PersonalField35', 'PersonalField3
6', 'PersonalField37', 'PersonalField38', 'PersonalField39', 'PersonalField4
0', 'PersonalField41', 'PersonalField42', 'PersonalField43', 'PersonalField4
4', 'PersonalField45', 'PersonalField46', 'PersonalField47', 'PersonalField4
8', 'PersonalField49', 'PersonalField50', 'PersonalField51', 'PersonalField5
2', 'PersonalField53', 'PersonalField54', 'PersonalField55', 'PersonalField5
6', 'PersonalField57', 'PersonalField58', 'PersonalField59', 'PersonalField6
0', 'PersonalField61', 'PersonalField62', 'PersonalField63', 'PersonalField6
4', 'PersonalField65', 'PersonalField66', 'PersonalField67', 'PersonalField6
8', 'PersonalField69', 'PersonalField70', 'PersonalField71', 'PersonalField7
2', 'PersonalField73', 'PersonalField74', 'PersonalField75', 'PersonalField7
6', 'PersonalField77', 'PersonalField78', 'PersonalField79', 'PersonalField8
0', 'PersonalField81', 'PersonalField82', 'PersonalField83', 'PersonalField8
4', 'PropertyField1A', 'PropertyField1B', 'PropertyField2A', 'PropertyField2
B', 'PropertyField6', 'PropertyField8', 'PropertyField9', 'PropertyField10',
'PropertyField11A', 'PropertyField11B', 'PropertyField12', 'PropertyField13',
'PropertyField15', 'PropertyField16A', 'PropertyField16B', 'PropertyField17',
'PropertyField18', 'PropertyField19', 'PropertyField20', 'PropertyField21A',
'PropertyField21B', 'PropertyField22', 'PropertyField23', 'PropertyField24A',
'PropertyField24B', 'PropertyField25', 'PropertyField26A', 'PropertyField26
B', 'PropertyField27', 'PropertyField29', 'PropertyField35', 'PropertyField39
A', 'PropertyField39B', 'GeographicField1A', 'GeographicField1B', 'Geographic
Field2A', 'GeographicField2B', 'GeographicField3A', 'GeographicField3B', 'Geo
graphicField4A', 'GeographicField4B', 'GeographicField5A', 'GeographicField5
B', 'GeographicField6A', 'GeographicField6B', 'GeographicField7A', 'Geographi
cField7B', 'GeographicField8A', 'GeographicField8B', 'GeographicField9A', 'Ge
ographicField9B', 'GeographicField10A', 'GeographicField10B', 'GeographicFiel
d11A', 'GeographicField11B', 'GeographicField12A', 'GeographicField12B', 'Geo
graphicField13A', 'GeographicField13B', 'GeographicField14A', 'GeographicFiel
d14B', 'GeographicField15A', 'GeographicField15B', 'GeographicField16A', 'Geo
graphicField16B', 'GeographicField17A', 'GeographicField17B', 'GeographicFiel
d18A', 'GeographicField18B', 'GeographicField19A', 'GeographicField19B', 'Geo
graphicField20A', 'GeographicField20B', 'GeographicField21A', 'GeographicFiel
d21B', 'GeographicField22A', 'GeographicField22B', 'GeographicField23A', 'Geo
graphicField23B', 'GeographicField24A', 'GeographicField24B', 'GeographicFiel
d25A', 'GeographicField25B', 'GeographicField26A', 'GeographicField26B', 'Geo
```

graphicField27A', 'GeographicField27B', 'GeographicField28A', 'GeographicFiel
d28B', 'GeographicField29A', 'GeographicField29B', 'GeographicField30A', 'Geo
graphicField30B', 'GeographicField31A', 'GeographicField31B', 'GeographicFiel
d32A', 'GeographicField32B', 'GeographicField33A', 'GeographicField33B', 'Geo
graphicField34A', 'GeographicField34B', 'GeographicField35A', 'GeographicFiel
d35B', 'GeographicField36A', 'GeographicField36B', 'GeographicField37A', 'Geo
graphicField37B', 'GeographicField38A', 'GeographicField38B', 'GeographicFiel
d39A', 'GeographicField39B', 'GeographicField40A', 'GeographicField40B', 'Geo
graphicField41A', 'GeographicField41B', 'GeographicField42A', 'GeographicFiel
d42B', 'GeographicField43A', 'GeographicField43B', 'GeographicField44A', 'Geo
graphicField44B', 'GeographicField45A', 'GeographicField45B', 'GeographicFiel
d46A', 'GeographicField46B', 'GeographicField47A', 'GeographicField47B', 'Geo
graphicField48A', 'GeographicField48B', 'GeographicField49A', 'GeographicFiel
d49B', 'GeographicField50A', 'GeographicField50B', 'GeographicField51A', 'Geo
graphicField51B', 'GeographicField52A', 'GeographicField52B', 'GeographicFiel
d53A', 'GeographicField53B', 'GeographicField54A', 'GeographicField54B', 'Geo
graphicField55A', 'GeographicField55B', 'GeographicField56A', 'GeographicFiel
d56B', 'GeographicField57A', 'GeographicField57B', 'GeographicField58A', 'Geo
graphicField58B', 'GeographicField59A', 'GeographicField59B', 'GeographicFiel
d60A', 'GeographicField60B', 'GeographicField61A', 'GeographicField61B', 'Geo
graphicField62A', 'GeographicField62B']

In [22]:
```
# Chọn cột 'Id' từ  num_cols
num_id=num_cols.pop(0)
print("num_id:", num_id)

# Lưu cột 'Id' trong num_id variable
num_id=[num_id]
print(num_id)

# Xóa cột 'label' trong numerical columns
# 'label' bị xóa vì mục tiêu validate mô hình
num_cols.remove('label')
print("num_cols:", num_cols)
```

```
num_id: Id
['Id']
num_cols: ['Field7', 'Field8', 'Field9', 'Field11', 'CoverageField1A', 'Cover
ageField1B', 'CoverageField2A', 'CoverageField2B', 'CoverageField3A', 'Covera
geField3B', 'CoverageField4A', 'CoverageField4B', 'CoverageField5A', 'Coverag
eField5B', 'CoverageField6A', 'CoverageField6B', 'CoverageField11A', 'Coverag
eField11B', 'SalesField1A', 'SalesField1B', 'SalesField2A', 'SalesField2B',
'SalesField3', 'SalesField4', 'SalesField5', 'SalesField6', 'SalesField8', 'S
alesField9', 'SalesField10', 'SalesField11', 'SalesField12', 'SalesField13',
'SalesField14', 'SalesField15', 'PersonalField1', 'PersonalField2', 'Personal
Field4A', 'PersonalField4B', 'PersonalField5', 'PersonalField6', 'PersonalFie
ld8', 'PersonalField9', 'PersonalField10A', 'PersonalField10B', 'PersonalFiel
d11', 'PersonalField12', 'PersonalField13', 'PersonalField14', 'PersonalField
15', 'PersonalField22', 'PersonalField23', 'PersonalField24', 'PersonalField2
5', 'PersonalField26', 'PersonalField27', 'PersonalField28', 'PersonalField2
9', 'PersonalField30', 'PersonalField31', 'PersonalField32', 'PersonalField3
3', 'PersonalField34', 'PersonalField35', 'PersonalField36', 'PersonalField3
7', 'PersonalField38', 'PersonalField39', 'PersonalField40', 'PersonalField4
1', 'PersonalField42', 'PersonalField43', 'PersonalField44', 'PersonalField4
5', 'PersonalField46', 'PersonalField47', 'PersonalField48', 'PersonalField4
9', 'PersonalField50', 'PersonalField51', 'PersonalField52', 'PersonalField5
3', 'PersonalField54', 'PersonalField55', 'PersonalField56', 'PersonalField5
7', 'PersonalField58', 'PersonalField59', 'PersonalField60', 'PersonalField6
1', 'PersonalField62', 'PersonalField63', 'PersonalField64', 'PersonalField6
5', 'PersonalField66', 'PersonalField67', 'PersonalField68', 'PersonalField6
9', 'PersonalField70', 'PersonalField71', 'PersonalField72', 'PersonalField7
3', 'PersonalField74', 'PersonalField75', 'PersonalField76', 'PersonalField7
7', 'PersonalField78', 'PersonalField79', 'PersonalField80', 'PersonalField8
1', 'PersonalField82', 'PersonalField83', 'PersonalField84', 'PropertyField1
A', 'PropertyField1B', 'PropertyField2A', 'PropertyField2B', 'PropertyField
6', 'PropertyField8', 'PropertyField9', 'PropertyField10', 'PropertyField11
A', 'PropertyField11B', 'PropertyField12', 'PropertyField13', 'PropertyField1
5', 'PropertyField16A', 'PropertyField16B', 'PropertyField17', 'PropertyField
18', 'PropertyField19', 'PropertyField20', 'PropertyField21A', 'PropertyField
21B', 'PropertyField22', 'PropertyField23', 'PropertyField24A', 'PropertyFiel
d24B', 'PropertyField25', 'PropertyField26A', 'PropertyField26B', 'PropertyFi
eld27', 'PropertyField29', 'PropertyField35', 'PropertyField39A', 'PropertyFi
eld39B', 'GeographicField1A', 'GeographicField1B', 'GeographicField2A', 'Geog
raphicField2B', 'GeographicField3A', 'GeographicField3B', 'GeographicField4
A', 'GeographicField4B', 'GeographicField5A', 'GeographicField5B', 'Geographi
cField6A', 'GeographicField6B', 'GeographicField7A', 'GeographicField7B', 'Ge
ographicField8A', 'GeographicField8B', 'GeographicField9A', 'GeographicField9
B', 'GeographicField10A', 'GeographicField10B', 'GeographicField11A', 'Geogra
phicField11B', 'GeographicField12A', 'GeographicField12B', 'GeographicField13
A', 'GeographicField13B', 'GeographicField14A', 'GeographicField14B', 'Geogra
phicField15A', 'GeographicField15B', 'GeographicField16A', 'GeographicField16
B', 'GeographicField17A', 'GeographicField17B', 'GeographicField18A', 'Geogra
phicField18B', 'GeographicField19A', 'GeographicField19B', 'GeographicField20
A', 'GeographicField20B', 'GeographicField21A', 'GeographicField21B', 'Geogra
phicField22A', 'GeographicField22B', 'GeographicField23A', 'GeographicField23
B', 'GeographicField24A', 'GeographicField24B', 'GeographicField25A', 'Geogra
phicField25B', 'GeographicField26A', 'GeographicField26B', 'GeographicField27
A', 'GeographicField27B', 'GeographicField28A', 'GeographicField28B', 'Geogra
phicField29A', 'GeographicField29B', 'GeographicField30A', 'GeographicField30
B', 'GeographicField31A', 'GeographicField31B', 'GeographicField32A', 'Geogra
phicField32B', 'GeographicField33A', 'GeographicField33B', 'GeographicField34
A', 'GeographicField34B', 'GeographicField35A', 'GeographicField35B', 'Geogra
```

```
phicField36A', 'GeographicField36B', 'GeographicField37A', 'GeographicField37
B', 'GeographicField38A', 'GeographicField38B', 'GeographicField39A', 'Geogra
phicField39B', 'GeographicField40A', 'GeographicField40B', 'GeographicField41
A', 'GeographicField41B', 'GeographicField42A', 'GeographicField42B', 'Geogra
phicField43A', 'GeographicField43B', 'GeographicField44A', 'GeographicField44
B', 'GeographicField45A', 'GeographicField45B', 'GeographicField46A', 'Geogra
phicField46B', 'GeographicField47A', 'GeographicField47B', 'GeographicField48
A', 'GeographicField48B', 'GeographicField49A', 'GeographicField49B', 'Geogra
phicField50A', 'GeographicField50B', 'GeographicField51A', 'GeographicField51
B', 'GeographicField52A', 'GeographicField52B', 'GeographicField53A', 'Geogra
phicField53B', 'GeographicField54A', 'GeographicField54B', 'GeographicField55
A', 'GeographicField55B', 'GeographicField56A', 'GeographicField56B', 'Geogra
phicField57A', 'GeographicField57B', 'GeographicField58A', 'GeographicField58
B', 'GeographicField59A', 'GeographicField59B', 'GeographicField60A', 'Geogra
phicField60B', 'GeographicField61A', 'GeographicField61B', 'GeographicField62
A', 'GeographicField62B']
```

Kiểm tra lại số dòng trên tập train

```
In [23]: df_final.count()
```

```
Out[23]: 260753
```

Xác định các categorical variable (biến phân loại) và nummerical variable (biến số) trong tập test

In [24]:
```
# Xác định Categorical variable
cat_cols_test = [item[0] for item in test_data.dtypes if item[1].startswith('s
tring')]
print("cat_cols_test:", cat_cols_test)

# Xác định Numerical variable
num_cols_test = [item[0] for item in test_data.dtypes if item[1].startswith('i
nt') | item[1].startswith('double')]
print("num_cols_test:", num_cols_test)
```

```
cat_cols_test: ['Field6', 'Field10', 'Field12', 'CoverageField8', 'CoverageFi
eld9', 'SalesField7', 'PersonalField7', 'PersonalField16', 'PersonalField17',
'PersonalField18', 'PersonalField19', 'PropertyField3', 'PropertyField4', 'Pr
opertyField5', 'PropertyField7', 'PropertyField14', 'PropertyField28', 'Prope
rtyField30', 'PropertyField31', 'PropertyField32', 'PropertyField33', 'Proper
tyField34', 'PropertyField36', 'PropertyField37', 'PropertyField38', 'Geograp
hicField63', 'GeographicField64']
num_cols_test: ['Id', 'Field7', 'Field8', 'Field9', 'Field11', 'CoverageField
1A', 'CoverageField1B', 'CoverageField2A', 'CoverageField2B', 'CoverageField3
A', 'CoverageField3B', 'CoverageField4A', 'CoverageField4B', 'CoverageField5
A', 'CoverageField5B', 'CoverageField6A', 'CoverageField6B', 'CoverageField11
A', 'CoverageField11B', 'SalesField1A', 'SalesField1B', 'SalesField2A', 'Sale
sField2B', 'SalesField3', 'SalesField4', 'SalesField5', 'SalesField6', 'Sales
Field8', 'SalesField9', 'SalesField10', 'SalesField11', 'SalesField12', 'Sale
sField13', 'SalesField14', 'SalesField15', 'PersonalField1', 'PersonalField
2', 'PersonalField4A', 'PersonalField4B', 'PersonalField5', 'PersonalField6',
'PersonalField8', 'PersonalField9', 'PersonalField10A', 'PersonalField10B',
'PersonalField11', 'PersonalField12', 'PersonalField13', 'PersonalField14',
'PersonalField15', 'PersonalField22', 'PersonalField23', 'PersonalField24',
'PersonalField25', 'PersonalField26', 'PersonalField27', 'PersonalField28',
'PersonalField29', 'PersonalField30', 'PersonalField31', 'PersonalField32',
'PersonalField33', 'PersonalField34', 'PersonalField35', 'PersonalField36',
'PersonalField37', 'PersonalField38', 'PersonalField39', 'PersonalField40',
'PersonalField41', 'PersonalField42', 'PersonalField43', 'PersonalField44',
'PersonalField45', 'PersonalField46', 'PersonalField47', 'PersonalField48',
'PersonalField49', 'PersonalField50', 'PersonalField51', 'PersonalField52',
'PersonalField53', 'PersonalField54', 'PersonalField55', 'PersonalField56',
'PersonalField57', 'PersonalField58', 'PersonalField59', 'PersonalField60',
'PersonalField61', 'PersonalField62', 'PersonalField63', 'PersonalField64',
'PersonalField65', 'PersonalField66', 'PersonalField67', 'PersonalField68',
'PersonalField69', 'PersonalField70', 'PersonalField71', 'PersonalField72',
'PersonalField73', 'PersonalField74', 'PersonalField75', 'PersonalField76',
'PersonalField77', 'PersonalField78', 'PersonalField79', 'PersonalField80',
'PersonalField81', 'PersonalField82', 'PersonalField83', 'PersonalField84',
'PropertyField1A', 'PropertyField1B', 'PropertyField2A', 'PropertyField2B',
'PropertyField6', 'PropertyField8', 'PropertyField9', 'PropertyField10', 'Pro
pertyField11A', 'PropertyField11B', 'PropertyField12', 'PropertyField13', 'Pr
opertyField15', 'PropertyField16A', 'PropertyField16B', 'PropertyField17', 'P
ropertyField18', 'PropertyField19', 'PropertyField20', 'PropertyField21A', 'P
ropertyField21B', 'PropertyField22', 'PropertyField23', 'PropertyField24A',
'PropertyField24B', 'PropertyField25', 'PropertyField26A', 'PropertyField26
B', 'PropertyField27', 'PropertyField29', 'PropertyField35', 'PropertyField39
A', 'PropertyField39B', 'GeographicField1A', 'GeographicField1B', 'Geographic
Field2A', 'GeographicField2B', 'GeographicField3A', 'GeographicField3B', 'Geo
graphicField4A', 'GeographicField4B', 'GeographicField5A', 'GeographicField5
B', 'GeographicField6A', 'GeographicField6B', 'GeographicField7A', 'Geographi
cField7B', 'GeographicField8A', 'GeographicField8B', 'GeographicField9A', 'Ge
ographicField9B', 'GeographicField10A', 'GeographicField10B', 'GeographicFiel
d11A', 'GeographicField11B', 'GeographicField12A', 'GeographicField12B', 'Geo
graphicField13A', 'GeographicField13B', 'GeographicField14A', 'GeographicFiel
d14B', 'GeographicField15A', 'GeographicField15B', 'GeographicField16A', 'Geo
graphicField16B', 'GeographicField17A', 'GeographicField17B', 'GeographicFiel
d18A', 'GeographicField18B', 'GeographicField19A', 'GeographicField19B', 'Geo
graphicField20A', 'GeographicField20B', 'GeographicField21A', 'GeographicFiel
d21B', 'GeographicField22A', 'GeographicField22B', 'GeographicField23A', 'Geo
graphicField23B', 'GeographicField24A', 'GeographicField24B', 'GeographicFiel
d25A', 'GeographicField25B', 'GeographicField26A', 'GeographicField26B', 'Geo
```

graphicField27A', 'GeographicField27B', 'GeographicField28A', 'GeographicFiel
d28B', 'GeographicField29A', 'GeographicField29B', 'GeographicField30A', 'Geo
graphicField30B', 'GeographicField31A', 'GeographicField31B', 'GeographicFiel
d32A', 'GeographicField32B', 'GeographicField33A', 'GeographicField33B', 'Geo
graphicField34A', 'GeographicField34B', 'GeographicField35A', 'GeographicFiel
d35B', 'GeographicField36A', 'GeographicField36B', 'GeographicField37A', 'Geo
graphicField37B', 'GeographicField38A', 'GeographicField38B', 'GeographicFiel
d39A', 'GeographicField39B', 'GeographicField40A', 'GeographicField40B', 'Geo
graphicField41A', 'GeographicField41B', 'GeographicField42A', 'GeographicFiel
d42B', 'GeographicField43A', 'GeographicField43B', 'GeographicField44A', 'Geo
graphicField44B', 'GeographicField45A', 'GeographicField45B', 'GeographicFiel
d46A', 'GeographicField46B', 'GeographicField47A', 'GeographicField47B', 'Geo
graphicField48A', 'GeographicField48B', 'GeographicField49A', 'GeographicFiel
d49B', 'GeographicField50A', 'GeographicField50B', 'GeographicField51A', 'Geo
graphicField51B', 'GeographicField52A', 'GeographicField52B', 'GeographicFiel
d53A', 'GeographicField53B', 'GeographicField54A', 'GeographicField54B', 'Geo
graphicField55A', 'GeographicField55B', 'GeographicField56A', 'GeographicFiel
d56B', 'GeographicField57A', 'GeographicField57B', 'GeographicField58A', 'Geo
graphicField58B', 'GeographicField59A', 'GeographicField59B', 'GeographicFiel
d60A', 'GeographicField60B', 'GeographicField61A', 'GeographicField61B', 'Geo
graphicField62A', 'GeographicField62B']

In [25]:
```
# Chọn cột 'Id' từ num_cols_test và lưu nó vào trong'num_id_test'
num_id_test=num_cols_test.pop(0)
print("num_id_test:", num_id_test)

# Lưu nó vào trong'num_id_test'
num_id_test=[num_id_test]
print(num_id_test)
print(num_cols_test)
```

```
num_id_test: Id
['Id']
['Field7', 'Field8', 'Field9', 'Field11', 'CoverageField1A', 'CoverageField1
B', 'CoverageField2A', 'CoverageField2B', 'CoverageField3A', 'CoverageField3
B', 'CoverageField4A', 'CoverageField4B', 'CoverageField5A', 'CoverageField5
B', 'CoverageField6A', 'CoverageField6B', 'CoverageField11A', 'CoverageField1
1B', 'SalesField1A', 'SalesField1B', 'SalesField2A', 'SalesField2B', 'SalesFi
eld3', 'SalesField4', 'SalesField5', 'SalesField6', 'SalesField8', 'SalesFiel
d9', 'SalesField10', 'SalesField11', 'SalesField12', 'SalesField13', 'SalesFi
eld14', 'SalesField15', 'PersonalField1', 'PersonalField2', 'PersonalField4
A', 'PersonalField4B', 'PersonalField5', 'PersonalField6', 'PersonalField8',
'PersonalField9', 'PersonalField10A', 'PersonalField10B', 'PersonalField11',
'PersonalField12', 'PersonalField13', 'PersonalField14', 'PersonalField15',
'PersonalField22', 'PersonalField23', 'PersonalField24', 'PersonalField25',
'PersonalField26', 'PersonalField27', 'PersonalField28', 'PersonalField29',
'PersonalField30', 'PersonalField31', 'PersonalField32', 'PersonalField33',
'PersonalField34', 'PersonalField35', 'PersonalField36', 'PersonalField37',
'PersonalField38', 'PersonalField39', 'PersonalField40', 'PersonalField41',
'PersonalField42', 'PersonalField43', 'PersonalField44', 'PersonalField45',
'PersonalField46', 'PersonalField47', 'PersonalField48', 'PersonalField49',
'PersonalField50', 'PersonalField51', 'PersonalField52', 'PersonalField53',
'PersonalField54', 'PersonalField55', 'PersonalField56', 'PersonalField57',
'PersonalField58', 'PersonalField59', 'PersonalField60', 'PersonalField61',
'PersonalField62', 'PersonalField63', 'PersonalField64', 'PersonalField65',
'PersonalField66', 'PersonalField67', 'PersonalField68', 'PersonalField69',
'PersonalField70', 'PersonalField71', 'PersonalField72', 'PersonalField73',
'PersonalField74', 'PersonalField75', 'PersonalField76', 'PersonalField77',
'PersonalField78', 'PersonalField79', 'PersonalField80', 'PersonalField81',
'PersonalField82', 'PersonalField83', 'PersonalField84', 'PropertyField1A',
'PropertyField1B', 'PropertyField2A', 'PropertyField2B', 'PropertyField6', 'P
ropertyField8', 'PropertyField9', 'PropertyField10', 'PropertyField11A', 'Pro
pertyField11B', 'PropertyField12', 'PropertyField13', 'PropertyField15', 'Pro
pertyField16A', 'PropertyField16B', 'PropertyField17', 'PropertyField18', 'Pr
opertyField19', 'PropertyField20', 'PropertyField21A', 'PropertyField21B', 'P
ropertyField22', 'PropertyField23', 'PropertyField24A', 'PropertyField24B',
'PropertyField25', 'PropertyField26A', 'PropertyField26B', 'PropertyField27',
'PropertyField29', 'PropertyField35', 'PropertyField39A', 'PropertyField39B',
'GeographicField1A', 'GeographicField1B', 'GeographicField2A', 'GeographicFie
ld2B', 'GeographicField3A', 'GeographicField3B', 'GeographicField4A', 'Geogra
phicField4B', 'GeographicField5A', 'GeographicField5B', 'GeographicField6A',
'GeographicField6B', 'GeographicField7A', 'GeographicField7B', 'GeographicFie
ld8A', 'GeographicField8B', 'GeographicField9A', 'GeographicField9B', 'Geogra
phicField10A', 'GeographicField10B', 'GeographicField11A', 'GeographicField11
B', 'GeographicField12A', 'GeographicField12B', 'GeographicField13A', 'Geogra
phicField13B', 'GeographicField14A', 'GeographicField14B', 'GeographicField15
A', 'GeographicField15B', 'GeographicField16A', 'GeographicField16B', 'Geogra
phicField17A', 'GeographicField17B', 'GeographicField18A', 'GeographicField18
B', 'GeographicField19A', 'GeographicField19B', 'GeographicField20A', 'Geogra
phicField20B', 'GeographicField21A', 'GeographicField21B', 'GeographicField22
A', 'GeographicField22B', 'GeographicField23A', 'GeographicField23B', 'Geogra
phicField24A', 'GeographicField24B', 'GeographicField25A', 'GeographicField25
B', 'GeographicField26A', 'GeographicField26B', 'GeographicField27A', 'Geogra
phicField27B', 'GeographicField28A', 'GeographicField28B', 'GeographicField29
A', 'GeographicField29B', 'GeographicField30A', 'GeographicField30B', 'Geogra
phicField31A', 'GeographicField31B', 'GeographicField32A', 'GeographicField32
B', 'GeographicField33A', 'GeographicField33B', 'GeographicField34A', 'Geogra
phicField34B', 'GeographicField35A', 'GeographicField35B', 'GeographicField36
```

```
A', 'GeographicField36B', 'GeographicField37A', 'GeographicField37B', 'Geogra
phicField38A', 'GeographicField38B', 'GeographicField39A', 'GeographicField39
B', 'GeographicField40A', 'GeographicField40B', 'GeographicField41A', 'Geogra
phicField41B', 'GeographicField42A', 'GeographicField42B', 'GeographicField43
A', 'GeographicField43B', 'GeographicField44A', 'GeographicField44B', 'Geogra
phicField45A', 'GeographicField45B', 'GeographicField46A', 'GeographicField46
B', 'GeographicField47A', 'GeographicField47B', 'GeographicField48A', 'Geogra
phicField48B', 'GeographicField49A', 'GeographicField49B', 'GeographicField50
A', 'GeographicField50B', 'GeographicField51A', 'GeographicField51B', 'Geogra
phicField52A', 'GeographicField52B', 'GeographicField53A', 'GeographicField53
B', 'GeographicField54A', 'GeographicField54B', 'GeographicField55A', 'Geogra
phicField55B', 'GeographicField56A', 'GeographicField56B', 'GeographicField57
A', 'GeographicField57B', 'GeographicField58A', 'GeographicField58B', 'Geogra
phicField59A', 'GeographicField59B', 'GeographicField60A', 'GeographicField60
B', 'GeographicField61A', 'GeographicField61B', 'GeographicField62A', 'Geogra
phicField62B']
```

Kiểm tra lại số dòng trên tập test.

```
In [26]: test_data.count()
```

```
Out[26]: 173836
```

Tổng số thuộc tính của hai biến categorical variable và nummerical variable trên tập test.

```
In [27]: len(num_cols_test), len(cat_cols_test)
```

```
Out[27]: (269, 27)
```

Kiểm tra và xử lý các giá trị bị thiếu trong tập train

Kiểm tra bị thiếu trong tập train

Chúng tôi kiểm tra các giá trị bị thiếu với bằng funtion count_nulls.

In [28]:
```
# Kiểm tra các giá trị bị thiếu  với Pyspark Dataframe
def count_nulls(c):
    """Input pyspark dataframe and return list of columns with missing value a
nd it's total value"""
    null_counts = []          # tạo một list rỗng để lưu kết quả
    for col in c.dtypes:
        cname = col[0]
        ctype = col[1]
        nulls = c.where( c[cname].isNull()).count() # điếm các giá trị null tr
ên mỗi cột
        result = tuple([cname, nulls])
        null_counts.append(result)
    null_counts=[(x,y) for (x,y) in null_counts if y!=0]  # xem các cột có giá
trị bị thiếu
    return null_counts
```

In [29]:
```
help(count_nulls)
```

```
Help on function count_nulls in module __main__:

count_nulls(c)
    Input pyspark dataframe and return list of columns with missing value and
it's total value
```

Tập dữ liệu có 298 cột và có 9 cột có giá trị bị thiếu (null).

In [30]:
```
# Sử dụng hàm count_nulls và dùng nó cho tập train (df_final)
%time null_counts = count_nulls(df_final)
null_counts
```

```
Wall time: 4min 57s
```

Out[30]:
```
[('PersonalField7', 113),
 ('PersonalField84', 124208),
 ('PropertyField3', 81),
 ('PropertyField4', 63),
 ('PropertyField29', 200685),
 ('PropertyField32', 70),
 ('PropertyField34', 70),
 ('PropertyField36', 113),
 ('PropertyField38', 1220)]
```

Từ hàm count_nulls chúng tôi nhận được thông tin tất cả các cột có giá trị bị thiếu, nó là một số và lưu vào danh sách "null_counts".

Từ null_counts, chúng tôi chỉ lấy thông tin của tên cột và lưu vào danh sách "list_cols_miss", giống như trong tập lệnh bên dưới:

In [31]:
```
list_cols_miss=[x[0] for x in null_counts]
list_cols_miss
```

Out[31]:
```
['PersonalField7',
 'PersonalField84',
 'PropertyField3',
 'PropertyField4',
 'PropertyField29',
 'PropertyField32',
 'PropertyField34',
 'PropertyField36',
 'PropertyField38']
```

Từ list_cols_miss tạo một dataframe có tên là "df_miss". Mục đích là xác định các giá trị bị thiếu trong cột categorical và numerical.

In [32]:
```
# Tạo dataframe chỉ có list_cols_miss
df_miss= df_final.select(*list_cols_miss)
```

In [33]:
```
df_miss.dtypes
```

Out[33]:
```
[('PersonalField7', 'string'),
 ('PersonalField84', 'int'),
 ('PropertyField3', 'string'),
 ('PropertyField4', 'string'),
 ('PropertyField29', 'int'),
 ('PropertyField32', 'string'),
 ('PropertyField34', 'string'),
 ('PropertyField36', 'string'),
 ('PropertyField38', 'string')]
```

Sau khi tạo "df_miss", chúng tôi chia thành hai cột categorical và numerical bị thiếu. Đối với cột categorical sẽ có tên là "catcolums_miss" và cột numerical sẽ có tên là "numcolumns_miss".

In [34]:
```
# Xác định cột categorical và cột numerical bị thiếu giá trị
# cột categorical
catcolums_miss=[item[0] for item in df_miss.dtypes if item[1].startswith('stri
ng')]  #will select name of column with string data type
print("catcolums_miss:", catcolums_miss)

# cột numerical
numcolumns_miss = [item[0] for item in df_miss.dtypes if item[1].startswith('i
nt') | item[1].startswith('double')] #will select name of column with integer
 or double data type
print("numcolumns_miss:", numcolumns_miss)
```

```
catcolums_miss: ['PersonalField7', 'PropertyField3', 'PropertyField4', 'Prope
rtyField32', 'PropertyField34', 'PropertyField36', 'PropertyField38']
numcolumns_miss: ['PersonalField84', 'PropertyField29']
```

Xử lý các giá trị bị thiếu trong tập train

Đối với các cột categorical chúng tôi sẽ điền các giá trị thường xuyên xuất hiện. Do đó, chúng tôi phải đếm các giá trị trong mỗi cột bằng cách đếm và sắp xếp giảm dần từng cột trong dataframe mà không có giá trị nào bị thiếu. Do đó, chúng tôi loại bỏ các giá trị bị thiếu và lưu trong dataframe mới được gọi là "df_Nomiss".

Đối với các cột numerical chúng tôi điền các giá trị còn thiếu với giá trị trung bình trong cột đó.

Chúng tôi tiến hành xóa các giá trị null trong cột.

In [35]:
```
df_Nomiss=df_final.na.drop()
```

Chúng tôi điền giá trị còn thiếu vào categorical variable với giá trị có tần suất xuất hiện nhiều nhất. Ví dụ như cột "PersonalField7" chúng tôi điền giá trị "N".

In [36]:
```
# Điền giá trị còn thiếu vào categorical variable  với giá trị có tần suất xuất hiện nhiều nhất
%time
for x in catcolums_miss:
    mode=df_Nomiss.groupBy(x).count().sort(col("count").desc()).collect()[0][0]
    print(x, mode)
    df_final = df_final.na.fill({x:mode})
```

```
Wall time: 0 ns
PersonalField7 N
PropertyField3 N
PropertyField4 N
PropertyField32 Y
PropertyField34 N
PropertyField36 N
PropertyField38 N
```

Chúng tôi điền giá trị còn thiếu vào numerical variable với giá trị trung bình của tất cả các giá trị trong cột. Ví dụ như cột "PersonalField84" chúng tôi điền giá trị "2.0".

In [37]:
```
# Điền giá trị còn thiếu vào biến số với giá trị trung bình
for i in numcolumns_miss:
    meanvalue = df_final.select(round(mean(i))).collect()[0][0]
    print(i, meanvalue)
    df_final=df_final.na.fill({i:meanvalue})
```

```
PersonalField84 2.0
PropertyField29 0.0
```

Chúng tôi kiểm tra xem các cột trong tập train còn giá trị nào bị thiếu nữa hay không.

In [38]:
```
# Kiểm tra giá trị còn thiếu sau khi điền
%time null_counts = count_nulls(df_final)
null_counts
```

```
Wall time: 6min 6s
```

Out[38]:
```
[]
```

Bây giờ, null_counts đầu ra là null, có nghĩa là không còn cột nào bị thiếu giá trị.

Kiểm tra và xử lý các giá trị bị thiếu ở tập test

Kiểm tra các giá trị bị thiếu ở tập test

Dùng Pyspark Dataframe

In [39]:
```
# Sử dụng hàm count_nulls và dùng nó cho tập test (test_data)
%time null_test= count_nulls(test_data)
null_test
```

```
Wall time: 3min 14s
```

Out[39]:
```
[('PersonalField7', 69),
 ('PersonalField84', 82812),
 ('PropertyField3', 69),
 ('PropertyField4', 52),
 ('PropertyField5', 1),
 ('PropertyField29', 133945),
 ('PropertyField30', 1),
 ('PropertyField32', 41),
 ('PropertyField34', 41),
 ('PropertyField36', 67),
 ('PropertyField38', 846)]
```

In [40]:
```
list_miss_test=[x[0] for x in null_test]
list_miss_test
```

Out[40]:
```
['PersonalField7',
 'PersonalField84',
 'PropertyField3',
 'PropertyField4',
 'PropertyField5',
 'PropertyField29',
 'PropertyField30',
 'PropertyField32',
 'PropertyField34',
 'PropertyField36',
 'PropertyField38']
```

In [41]:
```
test_miss= test_data.select(*list_miss_test)
# xem data types(kiểu dữ liệu của cột)
test_miss.dtypes
```

Out[41]:
```
[('PersonalField7', 'string'),
 ('PersonalField84', 'int'),
 ('PropertyField3', 'string'),
 ('PropertyField4', 'string'),
 ('PropertyField5', 'string'),
 ('PropertyField29', 'int'),
 ('PropertyField30', 'string'),
 ('PropertyField32', 'string'),
 ('PropertyField34', 'string'),
 ('PropertyField36', 'string'),
 ('PropertyField38', 'string')]
```

In [42]:
```
# Xác định cột categorical và cột numerical bị thiếu giá trị
# cột categorical
catcolums_miss_test=[item[0] for item in test_miss.dtypes if item[1].startswit
h('string')]  #will select name of column with string data type
print("catcolums_miss_test:", catcolums_miss_test)

# cột numerical
numcolumns_miss_test = [item[0] for item in test_miss.dtypes if item[1].starts
with('int') | item[1].startswith('double')] #will select name of column with i
nteger or double data type
print("numcolumns_miss_test:", numcolumns_miss_test)
```

```
catcolums_miss_test: ['PersonalField7', 'PropertyField3', 'PropertyField4',
'PropertyField5', 'PropertyField30', 'PropertyField32', 'PropertyField34', 'P
ropertyField36', 'PropertyField38']
numcolumns_miss_test: ['PersonalField84', 'PropertyField29']
```

Xử lý các giá trị bị thiếu trong tập test

Chúng tôi tiến hành xóa các giá trị null trong các cột của tập test.

In [43]:
```
test_Nomiss=test_data.na.drop()
```

Chúng tôi điền giá trị còn thiếu vào categorical variable với giá trị có tần suất xuất hiện nhiều nhất. Ví dụ như cột "PersonalField5" chúng tôi điền giá trị "Y".

In [44]:
```
# Điền giá trị còn thiếu vào categorical variable  với giá trị có tần suất xuất hiện nhiều nhất
for x in catcolums_miss_test:
    mode=test_Nomiss.groupBy(x).count().sort(col("count").desc()).collect()[0][0]
    print(x, mode)
    test_data = test_data.na.fill({x:mode})
```

```
PersonalField7 N
PropertyField3 N
PropertyField4 N
PropertyField5 Y
PropertyField30 N
PropertyField32 Y
PropertyField34 N
PropertyField36 N
PropertyField38 N
```

Tương tự, chúng tôi điền giá trị còn thiếu vào numerical variable với giá trị trung bình của tất cả các giá trị trong cột.

In [45]:
```
# Điền giá trị còn thiếu vào biến số với giá trị trung bình
for i in numcolumns_miss_test:
    meanvalue_test = test_data.select(round(mean(i))).collect()[0][0]
    print(i, meanvalue_test)
    test_data=test_data.na.fill({i:meanvalue_test})
```

```
PersonalField84 2.0
PropertyField29 0.0
```

Chúng tôi kiểm tra xem các cột trong tập test còn giá trị nào bị thiếu nữa hay không.

In [46]:
```
# Kiểm tra giá trị còn thiếu sau khi điền
%time null_test = count_nulls(test_data)
null_test
```

```
Wall time: 3min 29s
```

Out[46]:
```
[]
```

Bây giờ, null_test đầu ra là null, có nghĩa là không còn cột nào bị thiếu giá trị.

Đối chiếu các giá trị trong cột phân loại (categorical columns) ở tập train (df_final) và tập test (test_data)

Trong bước này, chúng tôi kiểm tra xem các giá trị trong cột phân loại (categorical columns) ở tập train và tập test có khớp với nhau hay không. Nếu không, các giá trị trong cột ở tập test sẽ được equated (làm bằng nhau) với tập train. Bước này là cần thiết để tránh sai sót trong feature engineering, nếu có sự khác nhau giữa các giá trị trong cột ở tập train và tập test, lỗi sẽ xuất hiện tại quy trình feature engineering trong tập test. Mô hình được đào tạo ở tập train sẽ không áp dụng được ở tập test.

In [47]:
```
def check_category(a1,a2,y):
    """đầu vào là hai dataframe bạn muốn đối chiếu và tên cột phân loại"""
    print('column:',y)
    var1=a1.select([y]).distinct()
    var2=a2.select([y]).distinct()
    diff2=var2.subtract(var1).collect()
    diff2=[r[y] for r in diff2]
    diff1=var1.subtract(var2).collect()
    diff1=[r[y] for r in diff1]
    if diff1 == diff2:  # giá trị hai cột ở tập train và test khớp với nhau
        print('diff2:', diff2)
        print('diff1:', diff1)
        print('Columns match!!')
    else:
        if len(diff1)!=0 and len(diff2)==len(diff1):
            print('diff2:', diff2)
            print('diff1:', diff1)
            # thay thế các giá trị trong cột của dataframe2 bằng các giá trị t
rong cột của dataframe1
            a2=a2.replace(diff2, diff1, y)
            print('Columns match now!!')
        else:
            if len(diff2)!=len(diff1) and len(diff2)!=0:
                print('diff2:', diff2)
                print('diff1:', diff1)
                # giá trị xuất hiền nhiều nhất trong một cột
                dominant1=a1.groupBy(y).count().sort(col("count").desc()).coll
ect()[0][0]
                dominant2=a2.groupBy(y).count().sort(col("count").desc()).coll
ect()[0][0]
                print('dominant2:', dominant2)
                print('dominant1:', dominant1)
                # thay thế các giá trị trong cột của dataframe2 bằng giá trị x
uất hiền nhiều nhất (dominant1)
                a2=a2.replace(diff2, dominant1, y)
                print('Columns match now!!')
            else:
                print('diff1:', diff1)
                print('diff2:', diff2)
    return a2
```

Chúng tôi tiến hành gọi hàm check_category2 để kiểm tra các giá trị trong cột phân loại (categorical columns) ở tập train (df_final) và tập test (test_data) có khớp với nhau không. Ví dụ như ở cột PersonalField16 các giá trị ở tập train ['XU', 'XN', 'XY', 'ZO', 'ZQ', 'ZV', 'ZL', 'ZS', 'ZP'] và ở tập test là ['ZM', 'XG', 'YG'], các giá trị sẽ được thay thế bằng 'ZA'.

In [48]:
```
for y in cat_cols_test:
   test_data=check_category(df_final,test_data,y)
```

```
column: Field6
diff2: []
diff1: []
Columns match!!
column: Field10
diff2: []
diff1: []
Columns match!!
column: Field12
diff2: []
diff1: []
Columns match!!
column: CoverageField8
diff2: []
diff1: []
Columns match!!
column: CoverageField9
diff2: []
diff1: []
Columns match!!
column: SalesField7
diff2: []
diff1: []
Columns match!!
column: PersonalField7
diff2: []
diff1: []
Columns match!!
column: PersonalField16
diff2: ['ZM', 'XG', 'YG']
diff1: ['XU', 'XN', 'XY', 'ZO', 'ZQ', 'ZV', 'ZL', 'ZS', 'ZP']
dominant2: ZA
dominant1: ZA
Columns match now!!
column: PersonalField17
diff2: ['XZ', 'XF', 'YO', 'ZJ']
diff1: ['XP', 'XY', 'ZR', 'XJ', 'YP', 'XO']
dominant2: ZE
dominant1: ZE
Columns match now!!
column: PersonalField18
diff2: ['XB']
diff1: ['ZG', 'XV', 'ZI', 'XH']
dominant2: XR
dominant1: XR
Columns match now!!
column: PersonalField19
diff2: ['ZS']
diff1: ['XL', 'YI', 'ZC']
dominant2: XD
dominant1: XD
Columns match now!!
column: PropertyField3
diff2: []
diff1: []
Columns match!!
column: PropertyField4
```

```
diff2: []
diff1: []
Columns match!!
column: PropertyField5
diff2: []
diff1: []
Columns match!!
column: PropertyField7
diff2: ['T']
diff1: ['B']
Columns match now!!
column: PropertyField14
diff2: []
diff1: []
Columns match!!
column: PropertyField28
diff2: []
diff1: []
Columns match!!
column: PropertyField30
diff2: []
diff1: []
Columns match!!
column: PropertyField31
diff2: []
diff1: []
Columns match!!
column: PropertyField32
diff2: []
diff1: []
Columns match!!
column: PropertyField33
diff2: []
diff1: []
Columns match!!
column: PropertyField34
diff2: []
diff1: []
Columns match!!
column: PropertyField36
diff2: []
diff1: []
Columns match!!
column: PropertyField37
diff2: [' ']
diff1: []
dominant2: N
dominant1: N
Columns match now!!
column: PropertyField38
diff2: []
diff1: []
Columns match!!
column: GeographicField63
diff2: []
diff1: []
Columns match!!
```

```
column: GeographicField64
diff2: []
diff1: []
Columns match!!
```

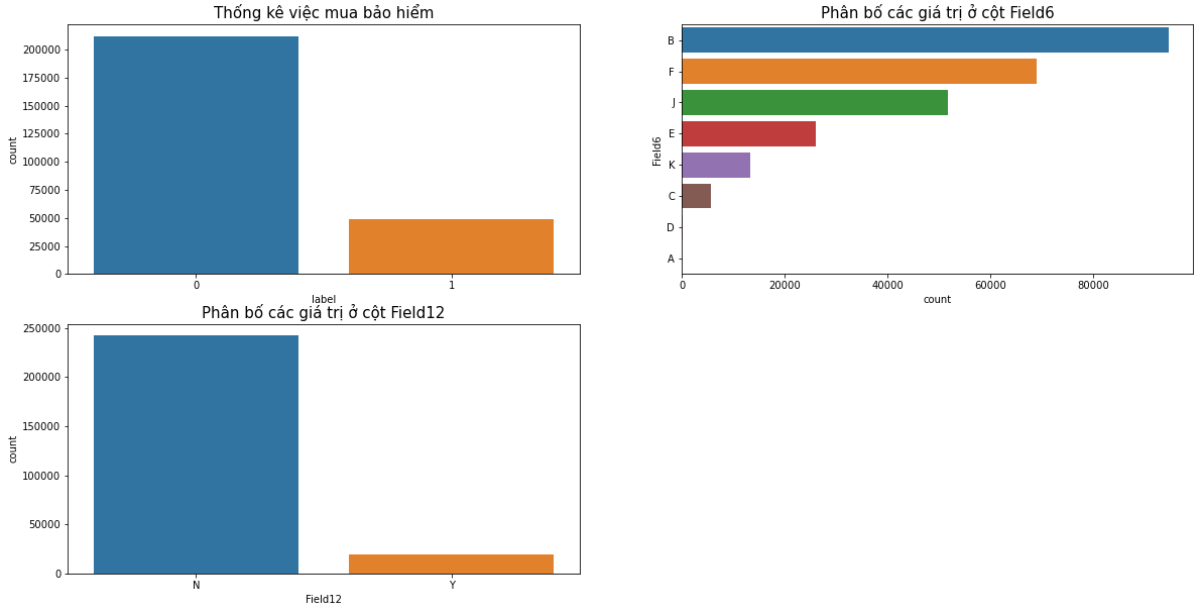
Exploratory data analysis (EDA)

Biểu đồ cột phân phối giá trị của một vài categorical variable

Ở biểu đồ "Thống kê việc mua bảo hiểm" cho thấy được tập train có đến 81.2% khách hàng là mua bảo hiểm và 18.8% khách hàng là không mua bảo hiểm.

Ở biểu đồ "Phân bố các giá trị ở cột Field6" và "Phân bố các giá trị ở cột Field12" cho thấy sự phân bố của các giá trị trong cột.

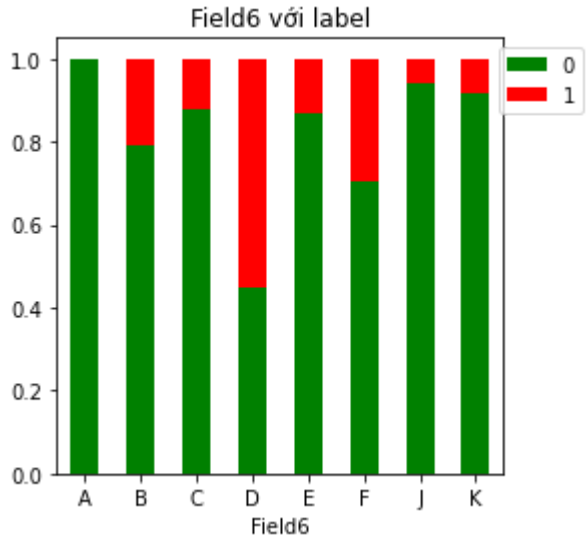
In [49]:
```
df_pd=df_final.toPandas()
plt.figure(figsize=(20,10))
plt.subplot(221)
sns.countplot(x='label', data=df_pd, order=df_pd['label'].value_counts().index
)
plt.title('Thống kê việc mua bảo hiểm', fontsize=15)
plt.subplot(222)
sns.countplot(y='Field6', data=df_pd, order=df_pd['Field6'].value_counts().ind
ex)
plt.title('Phân bố các giá trị ở cột Field6', fontsize=15)
plt.subplot(223)
sns.countplot(x='Field12', data=df_pd, order=df_pd['Field12'].value_counts().i
ndex)
plt.title('Phân bố các giá trị ở cột Field12', fontsize=15)
plt.show()
```

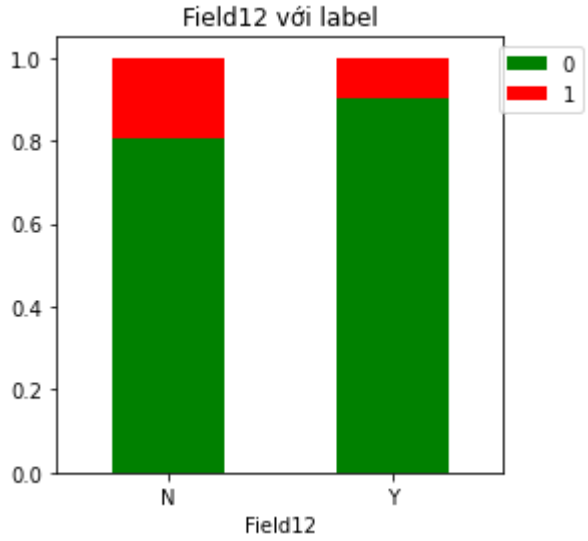


Chúng tôi tiến hành visualization một vài categorical variable (biến phân loại) với label (nhãn). Ví dụ đối với những khách hàng có giá trị "A" ở cột Field6 thì 100% là họ sẽ mua bảo hiểm (label= 0).

In [50]:
```
pd.crosstab(df_pd['Field6'], df_pd['label'], normalize='index').plot.bar(rot=0
, stacked=True,
            color=['green', 'red'], figsize=(4,4), title="Field6 với label")
plt.legend(loc='upper right', bbox_to_anchor=(1.2, 1))


pd.crosstab(df_pd['Field12'], df_pd['label'], normalize='index').plot.bar(rot=
0, stacked=True,
            color=['green', 'red'], figsize=(4,4), title="Field12 với label")
plt.legend(loc='upper right', bbox_to_anchor=(1.2, 1))
plt.show()
```

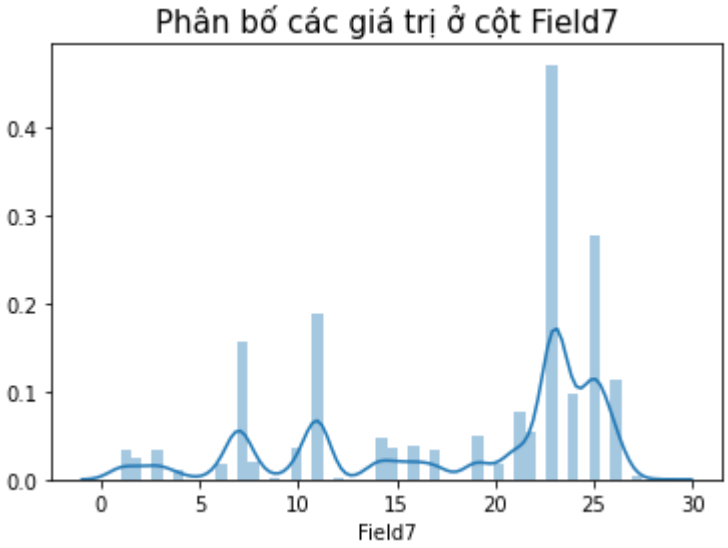




Biểu đồ cột phân phối giá trị của một numerical variables (biến số)

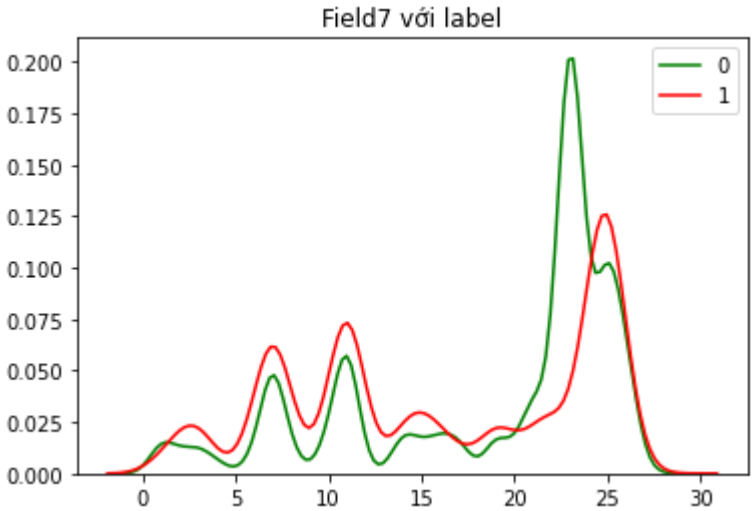
Chúng tôi tiến hành visualization sự phân bố giá trị ở cột Field7.

In [51]:
```
sns.distplot(df_pd['Field7'])
plt.title('Phân bố các giá trị ở cột Field7', fontsize=15)
plt.show()
```



Chúng tôi tiến hành visualization một numerical variables (biến số) với label (nhãn). Ví dụ một lượng lớn khách hàng mua bảo hiểm (label=0) thì giá trị trong cột Field7 của họ nằm ở khoảng 23-24.

In [52]:
```
sns.kdeplot(df_pd[df_pd["label"]==0]["Field7"], label="0", color="green")
sns.kdeplot(df_pd[df_pd["label"]==1]["Field7"], label="1", color="red")
plt.title("Field7 với label")
plt.show()
```



Thay thế (replace) các giá trị ít xuất hiện trong tập train và tập test

Chúng ta có thể thấy, có nhiều giá trị trong các cột với xuất hiện với số lượng ít, để không ảnh hưởng đến hiệu suất của mô hình chúng tôi tiến hành thay thế các giá trị này bằng các giá trị xuất hiện số lượng nhiều nhất trong mỗi cột. Chúng tôi chọn các giá trị xuất hiện dưới 0.7% trong tổng số 260753 thì sẽ bị thay thế.

In [53]:
```
threshold=98
threshold2=0.7
```

Đầu vào là một dataframe và categorical variables (biến phân loại), thay thế các giá trị xuất hiện không đáng kể bằng giá trị có phần trăm xuất hiện lớn nhất và đầu ra là một dataframe mới.

In [54]:
```
def replace_cat2(f,cols):
    df_percent=f.groupBy(cols).count().sort(col("count").desc())\
                .withColumn('total',sum(col('count')).over(window))\
                .withColumn('Percent',col('count')*100/col('total'))
    dominant_cat=df_percent.select(df_percent['Percent']).collect()[0][0]
    count_dist=f.select([cols]).distinct().count()
    if count_dist > 2 and dominant_cat <= threshold :
        print('column:', cols)
        cols_names.append(cols)
        replacement=f.groupBy(cols).count().sort(col("count").desc()).collect
()[0][0]
        print("replacement:",replacement)
        replacing.append(replacement)
        insign_cat=df_percent.filter(df_percent['Percent']< threshold2).select
(df_percent[cols]).collect()
        insign_cat=[r[cols] for r in insign_cat]
        category.append(insign_cat)
        print("insign_cat:",insign_cat)
        f=f.replace(insign_cat,replacement, cols)
    return f
```

Ví dụ chúng tôi tiến hành thống kê phần trăm xuất hiện của các giá trị trong cột Field6, cho thấy giá trị 'B' có phần trăm xuất hiện nhiều nhất (36.3%). Do đó các giá trị có phần trăm xuất hiện nhỏ hơn 0.7% như 'A' (0.01%) và D (0.09%) sẽ được thay thế bằng giá trị xuất hiện nhiều nhất 'B'.

In [55]:
```
df_percent=df_final.groupBy('Field6').count().sort(col("count").desc())\
                   .withColumn('total',sum(col('count')).over(window))\
                   .withColumn('Percent',col('count')*100/col('total'))
df_percent.show()
```

```
+------+-----+------+--------------------+
|Field6|count| total|             Percent|
+------+-----+------+--------------------+
|     B|94694|260753|  36.315593684444664|
|     F|69053|260753|  26.482149773923982|
|     J|51782|260753|   19.85864016904887|
|     E|26063|260753|   9.995282892239015|
|     K|13226|260753|   5.072233109494426|
|     C| 5660|260753|   2.170636579444915|
|     D|  240|260753| 0.09204112704360065|
|     A|   35|260753|0.013422664360525095|
+------+-----+------+--------------------+
```

Thay thế (replace) các giá trị ít xuất hiện trong tập train

In [56]:
```
replacing=[]
cols_names=[]
category=[]
for cols in cat_cols:
    df_final=replace_cat2(df_final,cols)
```

```
column: Field6
replacement: B
insign_cat: ['D', 'A']
column: Field10
replacement: 935
insign_cat: []
column: CoverageField8
replacement: Y
insign_cat: ['Z', 'U', 'V', 'W']
column: CoverageField9
replacement: E
insign_cat: ['B', 'I', 'H', 'C', 'L']
column: SalesField7
replacement: K
insign_cat: []
column: PersonalField16
replacement: ZA
insign_cat: ['XC', 'ZK', 'XI', 'ZD', 'ZE', 'XZ', 'ZJ', 'XL', 'XP', 'ZU', 'Y
I', 'XK', 'XF', 'ZB', 'XN', 'XT', 'ZI', 'XU', 'XY', 'ZO', 'ZQ', 'ZV', 'XV',
'ZL', 'ZS', 'ZP']
column: PersonalField17
replacement: ZE
insign_cat: ['YU', 'YS', 'YL', 'XH', 'ZL', 'ZW', 'XB', 'XW', 'XL', 'ZS', 'Y
K', 'YX', 'ZN', 'ZO', 'ZG', 'ZC', 'ZP', 'YI', 'XT', 'XN', 'ZU', 'XC', 'YM',
'XI', 'XQ', 'XX', 'YT', 'ZM', 'YE', 'ZT', 'YZ', 'ZA', 'YQ', 'ZD', 'YW', 'YG',
'XD', 'ZR', 'YP', 'XM', 'XP', 'XY', 'XJ', 'ZB', 'XO']
column: PersonalField18
replacement: XR
insign_cat: ['XK', 'XW', 'ZD', 'XO', 'ZL', 'XP', 'ZJ', 'ZF', 'YK', 'XN', 'Z
V', 'ZQ', 'ZT', 'XY', 'XJ', 'XS', 'YL', 'XC', 'ZK', 'ZE', 'ZA', 'XT', 'ZU',
'ZR', 'XL', 'XI', 'ZO', 'XM', 'YH', 'XE', 'YR', 'ZC', 'YO', 'ZS', 'XG', 'ZM',
'YJ', 'YM', 'XD', 'XX', 'XV', 'ZB', 'ZG', 'ZI', 'XH']
column: PersonalField19
replacement: XD
insign_cat: ['ZO', 'XT', 'ZA', 'YG', 'XZ', 'XY', 'ZJ', 'ZB', 'XF', 'YK', 'Z
W', 'YE', 'XP', 'XN', 'XW', 'ZI', 'XI', 'YL', 'XM', 'YN', 'ZL', 'ZE', 'ZH',
'ZK', 'XJ', 'ZG', 'XV', 'ZF', 'XR', 'ZP', 'XK', 'YM', 'XB', 'ZM', 'ZR', 'ZU',
'XE', 'ZD', 'XO', 'XQ', 'XG', 'XH', 'YI', 'XL', 'ZC', 'XS']
column: PropertyField7
replacement: O
insign_cat: ['K', 'H', 'E', 'L', 'F', 'G', 'M', 'P', 'C', 'B']
column: PropertyField14
replacement: C
insign_cat: ['D']
column: PropertyField28
replacement: B
insign_cat: ['C']
column: PropertyField31
replacement: O
insign_cat: []
column: PropertyField33
replacement: H
insign_cat: []
column: GeographicField63
replacement: N
insign_cat: [' ']
column: GeographicField64
```

```
replacement: CA
insign_cat: []
```

Từ hàm **replace_cat2** chúng tôi có ba list (danh sách) như sau:

• Danh sách các columns (cols_names)

• Danh sách các giá trị thay thế (replacing)

• Danh sách các giá trị bị thay thế (category)

Ba list (danh sách) này sẽ được tạo thành một dataframe gọi là **g**.

In [57]:
```
# Tạo dataframe
g=spark.createDataFrame(list(zip(cols_names, replacing, category)),['cols_name
s', 'replacing', 'category'])
g.show(9)
```

```
+---------------+---------+--------------------+
|     cols_names|replacing|            category|
+---------------+---------+--------------------+
|         Field6|        B|              [D, A]|
|        Field10|      935|                  []|
| CoverageField8|        Y|        [Z, U, V, W]|
| CoverageField9|        E|     [B, I, H, C, L]|
|    SalesField7|        K|                  []|
|PersonalField16|       ZA|[XC, ZK, XI, ZD, ...|
|PersonalField17|       ZE|[YU, YS, YL, XH, ...|
|PersonalField18|       XR|[XK, XW, ZD, XO, ...|
|PersonalField19|       XD|[ZO, XT, ZA, YG, ...|
+---------------+---------+--------------------+
only showing top 9 rows
```

In [58]:
```
g.dtypes
```

Out[58]:
```
[('cols_names', 'string'),
 ('replacing', 'string'),
 ('category', 'array<string>')]
```

In [59]:
```
g.printSchema()
```

```
root
 |-- cols_names: string (nullable = true)
 |-- replacing: string (nullable = true)
 |-- category: array (nullable = true)
 |    |-- element: string (containsNull = true)
```

Sau khi tiến hành thay thế các giá trị ít xuất hiện trong tập train, chúng tôi tiến hành kiểm tra lại phần trăm xuất hiện của giá trị trong cột. Bên dưới là ví dụ về phần trăm xuất hiện của các giá trị trong cột CoverageField8.

In [60]:
```
f_percent=df_final.groupBy('CoverageField8').count().sort(col("count").desc())
\
                        .withColumn('total',sum(col('count')).over(window))\
                        .withColumn('Percent',col('count')*100/col('total'))
```

In [61]:
```
f_percent.show()
```

```
+--------------+------+------+------------------+
|CoverageField8| count| total|           Percent|
+--------------+------+------+------------------+
|             Y|145350|260753|55.742407565780645|
|             T| 93778|260753|35.964303382894926|
|             X| 21625|260753| 8.293289051324434|
+--------------+------+------+------------------+
```

Thay thế (replace) các giá trị ít xuất hiện trong tập test

Chúng tôi đã có một dataframe g chứa thông tin về các cột, các giá trị thay thế và bị thay thế trong quá trình thay thế các các giá trị ít xuất hiện ở tập train. Dựa vào những thông tin trên, các giá trị ít xuất hiện ở các cột trong tập test sẽ bị thay thế tương tự như ở tập train.

In [62]:
```
#xem dataframe
g.show(15)
```

```
+-----------------+---------+--------------------+
|       cols_names|replacing|            category|
+-----------------+---------+--------------------+
|           Field6|        B|              [D, A]|
|          Field10|      935|                  []|
|   CoverageField8|        Y|        [Z, U, V, W]|
|   CoverageField9|        E|     [B, I, H, C, L]|
|      SalesField7|        K|                  []|
|  PersonalField16|       ZA|[XC, ZK, XI, ZD, ...|
|  PersonalField17|       ZE|[YU, YS, YL, XH, ...|
|  PersonalField18|       XR|[XK, XW, ZD, XO, ...|
|  PersonalField19|       XD|[ZO, XT, ZA, YG, ...|
|   PropertyField7|        O|[K, H, E, L, F, G...|
|  PropertyField14|        C|                 [D]|
|  PropertyField28|        B|                 [C]|
|  PropertyField31|        O|                  []|
|  PropertyField33|        H|                  []|
|GeographicField63|        N|                [ ]|
+-----------------+---------+--------------------+
only showing top 15 rows
```

Chúng tôi tiến hành thay thế các giá trị ít xuất hiện trong các cột ở tập test.

In [63]:
```
cols_names_list=g.select('cols_names').collect() # chọn lấy cols_names từ data
frame g
cols_names_list=[r['cols_names'] for r in cols_names_list]

# Hàm thay thế các giá trị ít xuất hiện trong tập test
for z in cols_names_list:
    print('cols_names:',z)
    replacement_cat=g.filter(g['cols_names']== z).select(g['replacing']).colle
ct()[0][0]
    print('replacement_cat:', replacement_cat)
    insignificant_cat=g.filter(g['cols_names']== z).select(g['category']).coll
ect()[0][0]
    print('insignificant_cat:',insignificant_cat)
    test_data=test_data.replace(insignificant_cat,replacement_cat, z)
```

```
cols_names: Field6
replacement_cat: B
insignificant_cat: ['D', 'A']
cols_names: Field10
replacement_cat: 935
insignificant_cat: []
cols_names: CoverageField8
replacement_cat: Y
insignificant_cat: ['Z', 'U', 'V', 'W']
cols_names: CoverageField9
replacement_cat: E
insignificant_cat: ['B', 'I', 'H', 'C', 'L']
cols_names: SalesField7
replacement_cat: K
insignificant_cat: []
cols_names: PersonalField16
replacement_cat: ZA
insignificant_cat: ['XC', 'ZK', 'XI', 'ZD', 'ZE', 'XZ', 'ZJ', 'XL', 'XP', 'Z
U', 'YI', 'XK', 'XF', 'ZB', 'XN', 'XT', 'ZI', 'XU', 'XY', 'ZO', 'ZQ', 'ZV',
'XV', 'ZL', 'ZS', 'ZP']
cols_names: PersonalField17
replacement_cat: ZE
insignificant_cat: ['YU', 'YS', 'YL', 'XH', 'ZL', 'ZW', 'XB', 'XW', 'XL', 'Z
S', 'YK', 'YX', 'ZN', 'ZO', 'ZG', 'ZC', 'ZP', 'YI', 'XT', 'XN', 'ZU', 'XC',
'YM', 'XI', 'XQ', 'XX', 'YT', 'ZM', 'YE', 'ZT', 'YZ', 'ZA', 'YQ', 'ZD', 'YW',
'YG', 'XD', 'ZR', 'YP', 'XM', 'XP', 'XY', 'XJ', 'ZB', 'XO']
cols_names: PersonalField18
replacement_cat: XR
insignificant_cat: ['XK', 'XW', 'ZD', 'XO', 'ZL', 'XP', 'ZJ', 'ZF', 'YK', 'X
N', 'ZV', 'ZQ', 'ZT', 'XY', 'XJ', 'XS', 'YL', 'XC', 'ZK', 'ZE', 'ZA', 'XT',
'ZU', 'ZR', 'XL', 'XI', 'ZO', 'XM', 'YH', 'XE', 'YR', 'ZC', 'YO', 'ZS', 'XG',
'ZM', 'YJ', 'YM', 'XD', 'XX', 'XV', 'ZB', 'ZG', 'ZI', 'XH']
cols_names: PersonalField19
replacement_cat: XD
insignificant_cat: ['ZO', 'XT', 'ZA', 'YG', 'XZ', 'XY', 'ZJ', 'ZB', 'XF', 'Y
K', 'ZW', 'YE', 'XP', 'XN', 'XW', 'ZI', 'XI', 'YL', 'XM', 'YN', 'ZL', 'ZE',
'ZH', 'ZK', 'XJ', 'ZG', 'XV', 'ZF', 'XR', 'ZP', 'XK', 'YM', 'XB', 'ZM', 'ZR',
'ZU', 'XE', 'ZD', 'XO', 'XQ', 'XG', 'XH', 'YI', 'XL', 'ZC', 'XS']
cols_names: PropertyField7
replacement_cat: O
insignificant_cat: ['K', 'H', 'E', 'L', 'F', 'G', 'M', 'P', 'C', 'B']
cols_names: PropertyField14
replacement_cat: C
insignificant_cat: ['D']
cols_names: PropertyField28
replacement_cat: B
insignificant_cat: ['C']
cols_names: PropertyField31
replacement_cat: O
insignificant_cat: []
cols_names: PropertyField33
replacement_cat: H
insignificant_cat: []
cols_names: GeographicField63
replacement_cat: N
insignificant_cat: [' ']
cols_names: GeographicField64
```

```
replacement_cat: CA
insignificant_cat: []
```

Biến đổi tập train data và tập test theo định dạng của Spark

Trước khi đào tạo mô hình, chúng tôi chuyển đổi tập dữ liệu theo định dạng của spark, đưa tất cả các categorical variables (biến phân loại) về dạng số. Trong SparkML có một cách xử lý đối với các categorical variables (biến phân loại) gồm các bước sau:

• StringIndexer, mã hóa string label (các giá trị nhãn ở dạng chuỗi) thành index label (các giá trị nhãn ở dạng số) bằng cách sắp xếp theo thứ tự giảm dần tần số xuất hiện của string label và đưa ra chỉ số nhỏ nhất (giá trị 0) đối với string label có tần số xuất hiện nhiều nhất.

• One-hot Encoding, mapping (ánh xạ) string label (các giá trị nhãn ở dạng chuỗi) sang dạng nhị phân.

• Vector assembler, mapping (ánh xạ) các cột sang dạng vector (features).

Đầu vào là hai dataframe mà chúng ta muốn so sánh (compare) các categorical variables (biến phân loại) và đầu ra là tổng các distinct categoriest trong cả hai dataframe.

In [64]:
```
def check_distinct(a1,a2):
    total1=0
    total2=0
    for y in cat_cols:
        distinct1=a1.select([y]).distinct().count()
        distinct2=a2.select([y]).distinct().count()
        var1=a1.select([y]).distinct().collect()
        var1=[r[y] for r in var1]
        var2=a2.select([y]).distinct().collect()
        var2=[r[y] for r in var2]
        total1=total1+distinct1
        total2=total2+distinct2
    return total1, total2
```

In [65]:
```
def feature_engineering(a1):
    cat_columns_string_vec = []
    for c in cat_cols:
        cat_columns_string= c+"_vec"
        cat_columns_string_vec.append(cat_columns_string)
    stringIndexer = [StringIndexer(inputCol=x, outputCol=x+"_Index")
                     for x in cat_cols]
    #Dùng onehothncoding để mã hóa categorical variable sang dạng nhị phân
    encoder = [OneHotEncoder(inputCol=x+"_Index", outputCol=y)
           for x,y in zip(cat_cols, cat_columns_string_vec)]
    tmp = [[i,j] for i,j in zip(stringIndexer, encoder)]
    tmp = [i for sublist in tmp for i in sublist]
    cols_assember=num_id + num_cols + cat_columns_string_vec
    assembler=VectorAssembler(inputCols=cols_assember, outputCol='features')
    tmp += [assembler]
    pipeline=Pipeline(stages=tmp)
    df_final_feat=pipeline.fit(a1).transform(a1)
    return df_final_feat
```

Chúng tôi dùng hàm check_distinct để kiểm tra distinct categories giữa tập train và tập test.

• Nếu distinct categories giữa tập train và tập test không giống nhau: Chúng tôi tiến hành join (kết hợp) tập train và tập test lại với nhau và sau đó thực hiện feature engineering. Việc join này là cần thiết để tránh sai sót trong việc xây dựng mô hình do sự khác biệt về độ dài vector của tập train và độ dài vector của tập test.

• Nếu distinct categories giữa tập train và tập test giống nhau: Chúng tôi thực hiện feature engineering ở tập train và tập test riêng biệt nhau.

In [66]:
```
def Main_feature_engineering(df,df2):
    dist_total1, dist_total2=check_distinct(df,df2)
    if dist_total1!=dist_total2:
        Label_df=df.select('Id', 'label')
        df_final2=df.drop('label')
        all_df =df_final2.union(df2)
        all_df_feat=feature_engineering(all_df)
        id_train=df.select('Id').collect()
        id_train=[r['Id'] for r in id_train]
        id_test=df2.select('Id').collect()
        id_test=[r['Id'] for r in id_test]
        a=all_df_feat.filter(all_df['Id'].isin(id_train))
        b=all_df_feat.filter(all_df['Id'].isin(id_test))
        a=a.join(Label_df, 'Id')
    else:
        a=feature_engineering(df)
        b=feature_engineering(df2)
    return a,b
```

Gọi sử dụng hàm feature engineering

In [67]:
```
%time data2, test2=Main_feature_engineering(df_final, test_data)
```

```
Wall time: 4min 13s
```

Xem kết quả feature engineering ở tập train.

In [68]:
```
data2.select('Id', 'features').show(5)
```

```
+---+--------------------+
| Id|            features|
+---+--------------------+
|  1|(397,[0,1,2,3,4,5...|
|  2|(397,[0,1,2,3,4,5...|
|  4|(397,[0,1,2,3,4,5...|
|  6|(397,[0,1,2,3,4,5...|
|  8|(397,[0,1,2,3,4,5...|
+---+--------------------+
only showing top 5 rows
```

Xem kết quả feature engineering ở tập test.

In [69]:
```
test2.select('Id', 'features').show(5)
```

```
+---+--------------------+
| Id|            features|
+---+--------------------+
|  3|(397,[0,1,2,3,4,5...|
|  5|(397,[0,1,2,3,4,5...|
|  7|(397,[0,1,2,3,4,5...|
|  9|(397,[0,1,2,3,4,5...|
| 10|(397,[0,1,2,3,4,5...|
+---+--------------------+
only showing top 5 rows
```

XÂY DỰNG VÀ ĐÁNH GIÁ MÔ HÌNH

Chia tập train thành train and validation

Chia tập train thành hai tập train và validation trước khi xây dựng mô hình. Chúng tôi dùng randomSplit để chia tập dữ liệu một cách ngẫu nhiên:

• Tập train (data_train) : 70%

• Tập validation (data_val): 30%

In [70]:
```
# Chia df_final thành train and validation, train 70% và validation 30%.
data_train, data_val=data2.randomSplit([0.7,0.3], 24)
```

Mô hình Logistic Regression

Xây dựng mô hình

In [71]:
```
# Xây dựng mô hình logistic regression từ tập data train
lr=LogisticRegression(featuresCol='features', labelCol='label')
lr_model = lr.fit(data_train)

# Áp dụng mô hình ở tập data val để kiểm tra
lr_result = lr_model.transform(data_val)
```

Kết quả bên dưới ,mô tả các id khách hàng với các nhãn (label) thật sự 0 là đã mua bảo hiểm và 1 là không mua bảo hiểm. Prediction là nhãn mà mô hình đã dự đoán. Probability là xác suất dự đoán của mô hình.

Ví dụ, khách hàng có id là 14 đã mua bảo hiểm và mô hình dự doán xác suất mà khách hàng này mua bảo hiểm là 98%, khách hàng có id là 26 không mua bảo hiểm và mô hình dự đoán xác suất khách hàng này mua bảo hiểm là 2%.

In [72]:
```
# Xem id, label, prediction (dự đoán) và kết quả xác suất của mô hình
lr_result.select('Id', 'label', 'prediction', 'probability').show(5)
```

```
+---+-----+----------+--------------------+
| Id|label|prediction|         probability|
+---+-----+----------+--------------------+
| 14|    0|       0.0|[0.98532361927230...|
| 19|    0|       0.0|[0.89594049307716...|
| 26|    1|       1.0|[0.02192190745466...|
| 28|    0|       0.0|[0.51188083524367...|
| 35|    0|       0.0|[0.92060508074044...|
+---+-----+----------+--------------------+
only showing top 5 rows
```

Đánh giá mô hình

$Accuracy$ là tỉ lệ giữa số điểm được phân loại đúng và tổng số điểm. AUC (Area Under the Curve) là diện tích nằm dưới ROC curve. Giá trị này là một số dương nhỏ hơn hoặc bằng 1. Giá trị này càng lớn thì mô hình càng tốt.

Chúng tôi tiến hành tính giá trị của $Accuracy$ và AUC. Kết quả thu được là $Accuracy$ = 0.91 và AUC = 0.94

In [73]:
```
# Đánh giá mô hình bằng cách kiểm tra giá trị accuracy và AUC.
lr_eval = BinaryClassificationEvaluator(rawPredictionCol="probability", labelC
ol="label")
lr_eval2= MulticlassClassificationEvaluator(predictionCol="prediction", labelC
ol="label")
lr_AUC  = lr_eval.evaluate(lr_result)
lr_ACC  = lr_eval2.evaluate(lr_result, {lr_eval2.metricName:"accuracy"})

print("Độ đo đánh giá  mô hình Logistic Regression")
print("Accuracy = %0.2f" % lr_ACC)
print("AUC = %.2f" % lr_AUC)
```

```
Độ đo đánh giá  mô hình Logistic Regression
Accuracy = 0.91
AUC = 0.94
```

Xây dựng Confusion Matrix

Chúng tôi xây dựng $ConfusionMatrix$. Sau đó, chúng tôi tiến hành tính $Accuracy$, $Sensitivity$, $Specificity$, $Precision$

In [74]:
```
cm_lr_result = lr_result.crosstab("prediction", "label")
cm_lr_result = cm_lr_result.toPandas()
cm_lr_result
```

Out[74]:

prediction_label	0	1
0	1.0	1775	9483
1	0.0	61739	5204

In [75]:
```
# Tính Accuracy, Sensitivity, Specificity, Precision
TP = cm_lr_result["1"][0]
FP = cm_lr_result["0"][0]
TN = cm_lr_result["0"][1]
FN = cm_lr_result["1"][1]
Accuracy = (TP+TN)/(TP+FP+TN+FN)
Sensitivity = TP/(TP+FN)
Specificity = TN/(TN+FP)
Precision = TP/(TP+FP)

print ("Accuracy = %0.2f" %Accuracy )
print ("Sensitivity = %0.2f" %Sensitivity )
print ("Specificity = %0.2f" %Specificity )
print ("Precision = %0.2f" %Precision )
```

```
Accuracy = 0.91
Sensitivity = 0.65
Specificity = 0.97
Precision = 0.84
```

Tính hệ số Gini từ AUC

In [76]:
```
AUC = lr_AUC
Gini = (2 * AUC - 1)
print("AUC=%.2f" % AUC)
print("GINI ~=%.2f" % Gini)
```

```
AUC=0.94
GINI ~=0.88
```

Mô hình Decision Tree

Xây dựng mô hình

In [77]:
```
#  Xây dựng mô hình decision tree từ tập data train
dt=DecisionTreeClassifier(featuresCol = 'features', labelCol = 'label', maxDep
th = 3)
dt_model = dt.fit(data_train)

# Áp dụng mô hình ở tập data val để kiểm tra
dt_result = dt_model.transform(data_val)
```

In [78]:
```
# Xem id, label, prediction (dự đoán) và kết quả xác suất của mô hình
dt_result.select('Id', 'label', 'prediction', 'probability').show(5)
```

```
+---+-----+----------+--------------------+
| Id|label|prediction|         probability|
+---+-----+----------+--------------------+
| 14|    0|       0.0|[0.94966697089150...|
| 19|    0|       0.0|[0.69203875009104...|
| 26|    1|       1.0|[0.00475237618809...|
| 28|    0|       0.0|[0.69203875009104...|
| 35|    0|       0.0|[0.94966697089150...|
+---+-----+----------+--------------------+
only showing top 5 rows
```

Đánh giá mô hình

Chúng tôi tiến hành tính giá trị của $Accuracy$ và AUC. Kết quả thu được là $Acuracy$ = 0.88 và AUC = 0.85

In [79]:
```
# Đánh giá mô hình bằng cách kiểm tra giá trị accuracy và AUC.
dt_eval = BinaryClassificationEvaluator(rawPredictionCol="probability", labelC
ol="label")
dt_eval2= MulticlassClassificationEvaluator(predictionCol="prediction", labelC
ol="label")
dt_AUC  = dt_eval.evaluate(dt_result)
dt_ACC  = dt_eval2.evaluate(dt_result, {dt_eval2.metricName:"accuracy"})

print("Độ đo đánh giá  mô hình Decision Tree")
print("Accuracy = %0.2f" % dt_ACC)
print("AUC = %.2f" % dt_AUC)
```

```
Độ đo đánh giá  mô hình Decision Tree
Accuracy = 0.88
AUC = 0.85
```

Xây dựng Confusion Matrix

Chúng tôi xây dựng $ConfusionMatrix$. Sau đó, chúng tôi tiến hành tính $Accuracy$, $Sensitivity$, $Specificity$, $Precision$

In [80]:
```
cm_dt_result = dt_result.crosstab("prediction", "label")
cm_dt_result = cm_dt_result.toPandas()
cm_dt_result
```

Out[80]:

prediction_label	0	1
0	1.0	35	5234
1	0.0	63479	9453

In [81]:
```
# Tính Accuracy, Sensitivity, Specificity, Precision
TP = cm_dt_result["1"][0]
FP = cm_dt_result["0"][0]
TN = cm_dt_result["0"][1]
FN = cm_dt_result["1"][1]
Accuracy = (TP+TN)/(TP+FP+TN+FN)
Sensitivity = TP/(TP+FN)
Specificity = TN/(TN+FP)
Precision = TP/(TP+FP)

print ("Accuracy = %0.2f" %Accuracy )
print ("Sensitivity = %0.2f" %Sensitivity )
print ("Specificity = %0.2f" %Specificity )
print ("Precision = %0.2f" %Precision )
```

```
Accuracy = 0.88
Sensitivity = 0.36
Specificity = 1.00
Precision = 0.99
```

Tính hệ số Gini từ AUC

Chúng tôi tính hệ số $Gini$ cho mô hình và được giá trị $Gini$ là 0.69.

In [82]:
```
AUC = dt_AUC
Gini_dt = (2 * AUC - 1)
print("AUC=%.2f" % AUC)
print("GINI ~=%.2f" % Gini_dt)
```

```
AUC=0.85
GINI ~=0.69
```

Mô hình Random Forest

Xây dựng mô hình

In [83]:
```
#  Xây dựng mô hình Random Forest từ tập data train
rf = RandomForestClassifier(featuresCol='features', labelCol="label")
rf_model = rf.fit(data_train)

# Áp dụng mô hình ở tập data val để kiểm tra
rf_result = rf_model.transform(data_val)
```

In [84]:
```
# Xem id, label, prediction (dự đoán) và kết quả xác suất của mô hình
rf_result.select('Id', 'label', 'prediction', 'probability').show(5)
```

```
+---+-----+----------+--------------------+
| Id|label|prediction|         probability|
+---+-----+----------+--------------------+
| 14|    0|       0.0|[0.90056297479093...|
| 19|    0|       0.0|[0.65826847267150...|
| 26|    1|       0.0|[0.65374732939939...|
| 28|    0|       0.0|[0.67971159610546...|
| 35|    0|       0.0|[0.82276665653703...|
+---+-----+----------+--------------------+
only showing top 5 rows
```

Đánh giá mô hình

In [85]:
```
# Đánh giá mô hình bằng cách kiểm tra giá trị accuracy và AUC.
rf_eval = BinaryClassificationEvaluator(rawPredictionCol="probability", labelC
ol="label")
rf_eval2= MulticlassClassificationEvaluator(predictionCol="prediction", labelC
ol="label")
rf_AUC  = rf_eval.evaluate(rf_result)
rf_ACC  = rf_eval2.evaluate(rf_result, {rf_eval2.metricName:"accuracy"})

print("Độ đo đánh giá  mô hình Random Forest ")
print("Accuracy = %0.2f" % rf_ACC)
print("AUC = %.2f" % rf_AUC)
```

```
Độ đo đánh giá  mô hình Random Forest
Accuracy = 0.82
AUC = 0.87
```

Xây dựng Confusion Matrix

Chúng tôi xây dựng $ConfusionMatrix$. Sau đó, chúng tôi tiến hành tính $Accuracy$, $Sensitivity$, $Specificity$, $Precision$

In [86]:
```
cm_rf_result = rf_result.crosstab("prediction", "label")
cm_rf_result = cm_rf_result.toPandas()
cm_rf_result
```

Out[86]:

prediction_label	0	1
0 1.0	0	288
1 0.0	63514	14399

In [87]:
```
# Tính Accuracy, Sensitivity, Specificity, Precision
TP = cm_rf_result["1"][0]
FP = cm_rf_result["0"][0]
TN = cm_rf_result["0"][1]
FN = cm_rf_result["1"][1]
Accuracy = (TP+TN)/(TP+FP+TN+FN)
Sensitivity = TP/(TP+FN)
Specificity = TN/(TN+FP)
Precision = TP/(TP+FP)

print ("Accuracy = %0.2f" %Accuracy )
print ("Sensitivity = %0.2f" %Sensitivity )
print ("Specificity = %0.2f" %Specificity )
print ("Precision = %0.2f" %Precision )
```

```
Accuracy = 0.82
Sensitivity = 0.02
Specificity = 1.00
Precision = 1.00
```

Tính hệ số Gini từ AUC

In [88]:
```
AUC = rf_AUC
Gini_rf= (2 * AUC -1)

print("AUC=%.2f" % AUC)
print("GINI ~=%.2f" % Gini_rf)
```

```
AUC=0.87
GINI ~=0.74
```

Dự đoán trên tập test

Chúng tôi sử dụng hai mô hình tốt nhất để tiến hành dự đoán kết quả trên tập test và xuất file csv kết quả.

Dự đoán dùng mô hình Logistic Regression

In [89]:
```
# Áp dụng mô hình trên tập test
lr_predict = lr_model.transform(test2)
```

In [90]:
```
# Xem id, label, prediction (dự đoán) và kết quả xác suất của mô hình
lr_predict.select('Id', 'prediction', 'probability').show(5)
```

```
+---+----------+--------------------+
| Id|prediction|         probability|
+---+----------+--------------------+
|  3|       0.0|[0.98506840365227...|
|  5|       0.0|[0.91044259403466...|
|  7|       0.0|[0.96879151308842...|
|  9|       0.0|[0.97435600997568...|
| 10|       0.0|[0.71416584885882...|
+---+----------+--------------------+
only showing top 5 rows
```

In [91]:
```
# Chọn id và prediction từ kết quả của mô hình và lưu vào data frame có tên là
my_submission_lr
my_submission=lr_predict.select("Id","prediction")
```

In [92]:
```
# Chuyển sang Pandas dataframe
my_submission=my_submission.toPandas()
```

In [93]:
```
# lưu file ở dạng csv
my_submission.to_csv('C:/Users/PC/BA_ML/project_BA_ML/my_submission_lr.csv', i
ndex = False, header = True)
```

Dự đoán dùng mô hình Decision Tree

In [94]:
```
# Áp dụng mô hình trên tập test
dt_predict = dt_model.transform(test2)
```

In [95]:
```
# Xem id, label, prediction (dự đoán) và kết quả xác suất của mô hình
dt_predict.select('Id', 'prediction', 'probability').show(5)
```

```
+---+----------+--------------------+
| Id|prediction|         probability|
+---+----------+--------------------+
|  3|       0.0|[0.94966697089150...|
|  5|       0.0|[0.94966697089150...|
|  7|       0.0|[0.94966697089150...|
|  9|       0.0|[0.94966697089150...|
| 10|       0.0|[0.94966697089150...|
+---+----------+--------------------+
only showing top 5 rows
```

In [96]:
```
# Chọn id và prediction từ kết quả của mô hình và lưu vào data frame có tên là
my_submission_dt
my_submission2=dt_predict.select("Id","prediction")
```

In [97]:
```
# Chuyển sang Pandas dataframe
my_submission2=my_submission2.toPandas()
```

In [98]:
```
#save to csv
my_submission2.to_csv('C:/Users/PC/BA_ML/project_BA_ML/my_submission_dt.csv',
index = False, header = True)
```

Kết luận: Như vậy chúng tôi đã thực hiện thành công việc dự đoán việc mua bảo hiểm của các khách hàng ở tập test.