BỘ KẾ HOẠCH VÀ ĐẦU TƯ

HỌC VIỆN CHÍNH SÁCH VÀ PHÁT TRIỂN

Trí tuệ và Phát triển

ĐỀ TÀI

BÁO CÁO PROJECT TELECOM

Nhóm 2
Lớp học phần: Quản trị DLL với Apache Spark 01
GVHD: Vũ Văn Hiệu

Năm học: 2023-2024

BẢNG THÔNG TIN THÀNH VIÊN NHÓM

STT	HỌ VÀ TÊN	MSV	
1	Nguyễn Trần Diệu Linh	7123112100	Trưởng nhóm
2	Nguyễn Thị Quỳnh Chi	7123112076	Thành viên
3	Nguyễn Phi Hùng	7123112093	Thành viên
4	Nguyễn Quốc Khánh	7123112098	Thành viên
5	Nguyễn Thành Vinh	7123112129	Thành viên
6	Ngô Quang Khải	7123112097	Thành viên

LỜI NÓI ĐẦU

Trong thời đại số hóa hiện nay, ngành viễn thông đang trở thành một trong những lĩnh vực quan trọng nhất, đóng vai trò không thể thay thế trong việc kết nối và giao tiếp giữa con người. Cùng với sự phát triển của công nghệ, dữ liệu trong lĩnh vực viễn thông cũng ngày càng tăng lên một cách đáng kể, tạo ra một cơ hội lớn cho việc áp dụng các công nghệ phân tích dữ liệu để cải thiện hiệu suất hoạt động và cung cấp dịch vụ tốt hơn cho người dùng.

Trong bối cảnh này, Apache Spark đã trở thành một công cụ mạnh mẽ và linh hoạt trong việc xử lý và phân tích dữ liệu lớn. Với khả năng xử lý dữ liệu song song và mở rộng linh hoạt, Apache Spark đã trở thành lựa chọn hàng đầu cho các ứng dụng trong nhiều lĩnh vực khác nhau, bao gồm cả viễn thông.

Bài tập lớn này nhằm mục đích áp dụng Apache Spark vào lĩnh vực viễn thông, từ việc phân tích dữ liệu từ các cuộc gọi điện thoại đến dự đoán nguy cơ chuyển mạng của khách hàng. Chúng em sẽ tập trung vào việc áp dụng các kỹ thuật xử lý dữ liệu và học máy để tạo ra các giải pháp hiệu quả, giúp các nhà cung cấp dịch vụ viễn thông cải thiện trải nghiệm của khách hàng và tối ưu hóa hoạt động kinh doanh của mình.

Thông qua bài tập này, chúng em hy vọng sẽ mang lại cái nhìn tổng quan về cách áp dụng Apache Spark trong ngành viễn thông và khám phá những tiềm năng mà công nghệ này mang lại. Chúng em cũng hy vọng rằng bài tập sẽ giúp bạn hiểu rõ hơn về quy trình phân tích dữ liệu và xây dựng mô hình trong một môi trường thực tế.

Xin chân thành cảm ơn sự quan tâm và hỗ trợ của các bạn trong suốt quá trình thực hiện bài tập này. Mong rằng bài tập sẽ mang lại những kiến thức hữu ích và trải nghiệm thú vị cho mọi người.

LỜI CẢM ƠN

Xin gửi lời biết ơn sâu sắc đến thầy đã dành thời gian và công sức để hướng dẫn nhóm trong suốt quá trình thực hiện bài tập lớn về Apache Spark trong lĩnh vực viễn thông.

Sự tận tâm và kiến thức sâu rộng của thầy không chỉ đã giúp chúng em hiểu rõ hơn về công nghệ và phương pháp phân tích dữ liệu, mà còn đã khuyến khích chúng em khám phá và áp dụng những kiến thức đó vào thực tế.

Chúng em rất biết ơn sự hỗ trợ và động viên từ thầy, những điều này đã giúp chúng em vượt qua những thách thức và phát triển kỹ năng của mình một cách toàn diện hơn.

Một lần nữa, chúng em xin chân thành cảm ơn thầy đã luôn bên cạnh và hướng dẫn chúng em trong suốt quá trình này. Sự hiểu biết và kinh nghiệm mà thầy chia sẻ sẽ luôn là nguồn động viên và cảm hứng lớn lao cho chúng em trong hành trình tương lai.

Xin chân thành cảm ơn!

MỤC LỤC

I) Giới thiệu bài toán

Trong ngành viễn thông, dữ liệu từ các cuộc gọi điện thoại, tin nhắn, dữ liệu internet và các dịch vụ khác đang trở thành một nguồn thông tin quan trọng, cung cấp thông tin sâu rộng về hành vi của người dùng và hiệu suất của mạng lưới. Đối với các nhà cung cấp dịch vụ viễn thông, việc phân tích và trích xuất insights từ dữ liệu này không chỉ giúp họ hiểu rõ hơn về nhu cầu của khách hàng mà còn giúp họ tối ưu hóa các chiến lược kinh doanh và cải thiện chất lượng dịch vụ.

Mô tả bài toán:

Dữ liệu đầu vào: Bao gồm thông tin về các cuộc gọi điện thoại như số điện thoại gọi, số điện thoại nhận, thời gian gọi, địa điểm và các thông tin kỹ thuật khác như mức độ tín hiệu, loại cuộc gọi, và thông tin về gói cước.
Mục tiêu:

- Phân tích hành vi của khách hàng: Dự đoán xu hướng gọi điện thoại, phân tích thói quen gọi và nhận cuộc gọi, xác định các mô hình gọi điện thoại phổ biến và không phổ biến.
- Đo lường hiệu suất mạng lưới: Theo dõi tỷ lệ gọi thất bại, thời gian kết nối trung bình, và sự phân phối của cuộc gọi theo vùng địa lý để phát hiện và giải quyết vấn đề kỹ thuật.
- Phát hiện gian lận và spam: Tìm kiếm các mô hình dự đoán và phát hiện các cuộc gọi lừa đảo hoặc gọi spam.

Ý nghĩa và ứng dụng:

- Hiểu rõ hơn về hành vi và nhu cầu của khách hàng: Phân tích dữ liệu cuộc gọi có thể giúp nhà cung cấp dịch vụ viễn thông hiểu rõ hơn về thói quen gọi điện thoại của khách hàng, từ đó tạo ra các dịch vụ và gói cước phù hợp.
- Cải thiện chất lượng dịch vụ: Bằng cách theo dõi và đánh giá hiệu suất mạng lưới từ dữ liệu cuộc gọi, các nhà cung cấp có thể phát hiện và giải quyết các vấn đề kỹ thuật một cách nhanh chóng, tăng cường trải nghiệm người dùng.
- Bảo vệ an ninh mạng: Phân tích dữ liệu cuộc gọi có thể giúp phát hiện và ngăn chặn các cuộc gọi lừa đảo, spam và các hoạt động đe dọa bảo mật thông tin cá nhân của người dùng.

Phạm vi dự án:

- Thu thập, tiền xử lý và khám phá dữ liệu cuộc gọi từ hệ thống viễn thông.
- Áp dụng các kỹ thuật xử lý dữ liệu lớn và học máy để phân tích và trích xuất insights từ dữ liệu.
- Xây dựng các mô hình dự đoán và phát hiện gian lận từ dữ liệu cuộc gọi.
- Tạo ra các báo cáo và đồ thị minh họa để trình bày kết quả phân tích và đánh giá hiệu suất mô hình.

Bài toán phân tích dữ liệu từ hệ thống viễn thông không chỉ giúp chúng ta hiểu rõ hơn về hành vi của khách hàng mà còn đóng vai trò quan trọng trong việc cải thiện dịch vụ và quản lý mạng lưới. Chúng ta hy vọng rằng việc áp dụng Apache Spark trong bài toán này sẽ mang lại những insights giá trị và đóng góp tích cực vào ngành viễn thông.

II) Nghiên cứu liên quan

Trong lĩnh vực viễn thông, có nhiều nghiên cứu quan trọng đã được thực hiện với mục đích nghiên cứu và phát triển các phương pháp phân tích dữ liệu từ các nguồn dữ liệu viễn thông như cuộc gọi điện thoại, tin nhắn và dữ liệu internet. Dưới đây là một số nghiên cứu liên quan mà chúng ta có thể tham khảo:

"Big data analytics for user mobility and call activities prediction in telecom: A survey" (2020) của Hou et al.: Nghiên cứu này cung cấp một tổng quan về các phương pháp và kỹ thuật phân tích dữ liệu lớn để dự đoán di chuyển người dùng và hoạt động gọi trong ngành viễn thông. Tác giả đã phân tích các phương pháp học máy và mô hình dự đoán để hiểu và dự đoán hành vi của khách hàng.

"Telecommunications data for the humanitarian community: A systematic review" (2021) của Avsigol et al.: Nghiên cứu này tập trung vào việc khám phá tiềm năng của dữ liệu viễn thông trong việc hỗ trợ các hoạt động nhân đạo. Tác giả đã đánh giá các ứng dụng của dữ liệu viễn thông trong việc dự đoán và phản ứng đối với các tình huống khẩn cấp và tình trạng khẩn cấp, như phản ứng cứu trợ sau thiên tai.

"Predictive analytics for customer churn in the telecommunications industry: A systematic literature review" (2021) của Zeng et al.: Nghiên cứu này tập trung vào việc áp dụng các phương pháp dự đoán để phòng ngừa và giảm thiểu sự rời bỏ của khách hàng trong ngành viễn thông. Tác giả đã tổng hợp các phương pháp và mô hình dự đoán khách hàng rời bỏ từ các nghiên cứu trước đây để đưa ra cái nhìn tổng quan về lĩnh vực này.

Những nghiên cứu trên cung cấp các góc nhìn đa dạng về ứng dụng của dữ liệu viễn thông và các phương pháp phân tích dữ liệu trong lĩnh vực này. Chúng ta có thể tận dụng những kiến thức và kinh nghiệm từ những nghiên cứu này để áp dụng vào bài toán của chúng ta và tạo ra những giải pháp hiệu quả.

III) Phương pháp

3.1 Các kỹ thuật nền tảng

Trong quá trình phân tích dữ liệu viễn thông, chúng ta cần sử dụng một loạt các kỹ thuật và công nghệ nền tảng để tiền xử lý dữ liệu, xây dựng mô hình và trích xuất insights. Dưới đây là một số kỹ thuật nền tảng cụ thể:

- Tiền xử lý dữ liệu:

Xử lý dữ liệu thiếu: Trong quá trình thu thập dữ liệu từ các nguồn khác nhau, có thể xuất hiện các dòng dữ liệu thiếu hoặc không hợp lệ. Chúng ta cần áp dụng các kỹ thuật như điền giá trị trung bình, median hoặc loại bỏ dòng dữ liệu để xử lý vấn đề này.

Loại bỏ nhiễu: Dữ liệu viễn thông thường chứa nhiều nhiễu và dữ liệu không chính xác. Chúng ta cần sử dụng các kỹ thuật như lọc nhiễu, làm sạch dữ liệu để loại bỏ nhiễu và cải thiện chất lượng dữ liệu.

Chuyển đổi dữ liệu: Dữ liệu viễn thông có thể được thu thập từ nhiều nguồn khác nhau và có định dạng khác nhau. Chúng ta cần chuyển đổi dữ liệu thành định dạng thích hợp để thuận tiện cho việc phân tích và xử lý.

- Phân tích dữ liệu lớn:

Apache Spark: Apache Spark là một framework phổ biến cho phân tích dữ liệu lớn. Với khả năng xử lý dữ liệu song song và phân tán, Spark sẽ giúp chúng ta xử lý dữ liệu viễn thông một cách hiệu quả và nhanh chóng.

Hadoop: Hadoop là một hệ sinh thái công nghệ sử dụng để lưu trữ và xử lý dữ liệu lớn. Chúng ta có thể sử dụng Hadoop cùng với Spark để lưu trữ và xử lý dữ liệu viễn thông trên cụm máy tính phân tán.

❖ Học máy và khai phá dữ liệu:

Phân loại và dự đoán: Sử dụng các thuật toán học máy như Random Forest, Gradient Boosting, hoặc Neural Networks để phân loại và dự đoán các biến số quan trọng như hành vi của khách hàng, khả năng churn, và phát hiện gian lận.
Phân tích cụm: Áp dụng kỹ thuật phân tích cụm (clustering) như K-means để nhận biết các nhóm khách hàng có hành vi tương tự, từ đó tạo ra các chiến lược phục vụ và tiếp thị phù hợp.

❖ Visual Analytics:

Sử dụng các công cụ như Matplotlib, Seaborn hoặc Plotly để tạo ra các biểu đồ và đồ thị trực quan hóa dữ liệu. Việc này giúp chúng ta hiểu rõ hơn về các mẫu và xu hướng trong dữ liệu viễn thông.

=> Bằng cách sử dụng kết hợp các kỹ thuật và công nghệ nền tảng này, chúng ta có thể xử lý và phân tích dữ liệu viễn thông một cách hiệu quả, từ đó tạo ra những insights giá trị để hỗ trợ quyết định kinh doanh và cải thiện dịch vụ cho khách hàng.

3.2 Đề xuất phương pháp

Dựa trên nền tảng của các kỹ thuật đã được đề cập ở phần trước, chúng em đề xuất một phương pháp cụ thể để phân tích dữ liệu viễn thông trong bài toán này:

- *Thu thập dữ liệu:* Thu thập dữ liệu từ các nguồn khác nhau như hệ thống cuộc gọi điện thoại, tin nhắn, dữ liệu internet và dịch vụ khác trong ngành viễn thông. Dữ liệu này có thể bao gồm thông tin về số điện thoại, thời gian gọi, địa điểm, loại cuộc gọi và các thuộc tính khác liên quan.
- *Tiền xử lý dữ liệu:* Tiến hành tiền xử lý dữ liệu bao gồm xử lý dữ liệu thiếu, loại bỏ nhiễu, và chuẩn hóa dữ liệu để chuẩn bị cho quá trình phân tích.
- *Phân tích hành vi khách hàng:* Sử dụng các kỹ thuật phân tích mẫu dữ liệu để khám phá các mẫu và xu hướng trong hành vi của khách hàng, như thói quen gọi điện thoại, vị trí phổ biến, và thời gian gọi.

Xây dựng các mô hình dự đoán để dự đoán các hành vi tiêu biểu của khách hàng, như thời điểm gọi điện thoại phổ biến nhất hoặc xác suất gọi spam.

- *Đánh giá hiệu suất mạng lưới:* Phân tích các chỉ số hiệu suất mạng lưới như tỷ lệ gọi thất bại, thời gian kết nối trung bình, và sự phân phối của cuộc gọi theo vùng địa lý.

Sử dụng các kỹ thuật phân tích dữ liệu để phát hiện và giải quyết các vấn đề kỹ thuật, từ đó cải thiện chất lượng dịch vụ cho người dùng.

- *Phát hiện gian lận và spam:* Xây dựng các mô hình phân loại để phát hiện các cuộc gọi lừa đảo hoặc gọi spam dựa trên các đặc điểm và mẫu trong dữ liệu.

Tối ưu hóa mô hình và đánh giá hiệu suất để đảm bảo khả năng phát hiện chính xác và giảm thiểu số lượng dương tính giả.

- Trình bày kết quả: Tạo ra các báo cáo và đồ thị minh họa để trình bày kết quả phân tích và đánh giá hiệu suất, từ đó cung cấp insights giá trị cho các quyết định kinh doanh và quản lý mạng lưới.

Bằng cách áp dụng phương pháp này, hy vọng sẽ có thể tận dụng tối đa dữ liệu viễn thông và tạo ra những insights quan trọng để cải thiện dịch vụ và quản lý mạng lưới viễn thông một cách hiệu quả.

IV) Thực nghiệm

4.1. Miêu tả dữ liệu

CustomerID: Mã số của khách hàng.
Churn: Trạng thái churn của khách hàng, có thể là "Yes" (rời bỏ) hoặc "No" (không rời bỏ).
MonthlyRevenue: Doanh thu hàng tháng của khách hàng.
MonthlyMinutes: Số phút sử dụng hàng tháng của khách hàng.
TotalRecurringCharge: Tổng lệ phí định kỳ hàng tháng.
DirectorAssistedCalls: Số cuộc gọi được hỗ trợ bởi giám đốc.
OverageMinutes: Số phút vượt giới hạn hàng tháng.
RoamingCalls: Số cuộc gọi lưu động.
PercChangeMinutes: Tỷ lệ thay đổi phút sử dụng so với tháng trước.
PercChangeRevenues: Tỷ lệ thay đổi doanh thu so với tháng trước.
DroppedCalls: Số cuộc gọi bị từ chối hoặc kết thúc đột ngột.
BlockedCalls: Số cuộc gọi bị chặn.
UnansweredCalls: Số cuộc gọi không được trả lời.
CustomerCareCalls: Số cuộc gọi tới dịch vụ chăm sóc khách hàng.
ThreewayCalls: Số cuộc gọi ba chiều.
ReceivedCalls: Số cuộc gọi đến.
OutboundCalls: Số cuộc gọi đi.
InboundCalls: Số cuộc gọi đến.
PeakCallsInOut: Số cuộc gọi đỉnh hàng ngày (đến và đi).
OffPeakCallsInOut: Số cuộc gọi ngoại giờ hàng ngày (đến và đi).
DroppedBlockedCalls: Số cuộc gọi bị từ chối hoặc chặn.
CallForwardingCalls: Số cuộc gọi chuyển tiếp.
CallWaitingCalls: Số cuộc gọi đang chờ.
MonthsInService: Số tháng sử dụng dịch vụ.
UniqueSubs: Số lượng thuê bao duy nhất.
ActiveSubs: Số lượng thuê bao hoạt động.
ServiceArea: Khu vực dịch vụ.
và các thuộc tính khác như Handsets, IncomeGroup, CreditRating, Occupation, MaritalStatus, vv.

4.2 Tiền xử lý dữ liệu

Trước khi lập mô hình và trực quan hóa, trước hết chúng ta cần tuân theo quy trình kỹ thuật dữ liệu, cụ thể là quy trình ETL, viết tắt của Trích xuất, chuyển đổi và tải các biến từ các mục dữ liệu đã xác định.

```
# Delete the rows with 'NA' values
for column in df.columns:
    if df.schema[column].dataType == pyspark.sql.types.StringType():
        df = df.filter(df[column] != "NA")
```

```
# Check the description of Churn column
df.select("Churn").describe().show()


# Going through all columns required to be transfer into numerical values
columns_string = [
    column
    for column in df.columns
    if df.schema[column].dataType == pyspark.sql.types.StringType()
]
columns_string


# create the view of df under spark
df.createOrReplaceTempView("df")


# check the distribution of picked columns
spark.sql(
    "SELECT \
            Churn, \
            CAST (avg(MonthlyRevenue) as decimal(8,2)) as avg_MonthlyRevenue, \
            CAST (avg(MonthlyMinutes) as decimal(8,2)) as avg_MonthlyMinutes, \
            CAST (avg(CurrentEquipmentDays) as decimal(8,2)) as
avg_CurrentEquipmentDays, \
            CAST (avg(TotalRecurringCharge) as decimal(8,2)) as
avg_TotalRecurringCharge, \
            CAST (avg(OverageMinutes) as decimal(8,2)) as avg_OverageMinutes, \
            CAST (avg(RoamingCalls) as decimal(8,2)) as avg_RoamingCalls \
            FROM df GROUP BY Churn"
).show()


# Check the values of string columns
df.select([column for column in df.columns if column in columns_string]).show(1)


# Loop the list and use StringIndexer encodes the string columns of labels(Yes or
No) to columns of label indices(1 or 0)
for column in columns_for_indexer:
    indexer = StringIndexer(inputCol=column, outputCol=column + "Index")
    df = indexer.fit(df).transform(df)
# Drop the original string columns of labels
df = df.select([column for column in df.columns if column not in
columns_for_indexer])
```

```
# Check the Churn column again to confirm the result
df.select("ChurnIndex").describe().show()


# Replace unknown values to 0 for Price Column
df = df.replace(to_replace={"Unknown": "0"}, subset=["HandsetPrice"])


# Delete the rows with 'NA' values
for column in df.columns:
    if df.schema[column].dataType == pyspark.sql.types.StringType():
        df = df.filter(df[column] != "Unknown")


# Another Iteration for feature engineering
columns_string = [
    column
    for column in df.columns
    if df.schema[column].dataType == pyspark.sql.types.StringType()
]
columns_string


# Mapping string values into integer based on its values
mapping_PrizmCode = {"Other": "0", "Suburban": "1", "Town": "2", "Rural": "3"}
df = df.replace(to_replace=mapping_PrizmCode, subset=["PrizmCode"])


# Prepare the list of columns to be transformed into float
columns_to_float = [
    "MonthlyRevenue",
    "MonthlyMinutes",
    "TotalRecurringCharge",
    "DirectorAssistedCalls",
    "OverageMinutes",
    "RoamingCalls",
    "PercChangeMinutes",
    "PercChangeRevenues",
]


# Transform ServiceArea and CreditRating columns only to keep the int value
df = df.withColumn("ServiceArea", df["ServiceArea"].substr(-3, 3))
df = df.withColumn("CreditRating", df["CreditRating"].substr(1, 1))
```

```
# Transform the type of columns to float
for column in columns_to_float:
    df = df.withColumn(column, df[column].cast("float"))


# Create a Pandas DataFrame for data visualization
df_pd = df.toPandas()


# Before transform the type of Occupation column, let's check it's distribution first
graph = df_pd[["Occupation", "MonthlyRevenue"]]
ax = sns.barplot(x="Occupation", y="MonthlyRevenue", data=graph)


# Transform string values to numbers using mapping
temp = (
    df_pd.loc[:, ["Occupation", "MonthlyRevenue"]]
    .groupby("Occupation")
    .mean()
    .sort_values(["MonthlyRevenue"], ascending=[0])
)


mapping_Occupation = dict([temp.index[i], str(i)] for i in range(len(temp)))
print(mapping_Occupation)

df = df.replace(to_replace=mapping_Occupation, subset=["Occupation"])


# Prepare the list of columns to be transformed into integer
columns_to_int = [
    "Handsets",
    "HandsetModels",
    "CurrentEquipmentDays",
    "AgeHH1",
    "AgeHH2",
    "HandsetPrice",
    "ServiceArea",
    "CreditRating",
    "PrizmCode",
    "Occupation",
    "ChurnIndex",
    "ChildrenInHHIndex",
    "HandsetRefurbishedIndex",
    "HandsetWebCapableIndex",
    "TruckOwnerIndex",
    "RVOwnerIndex",
    "HomeownershipIndex",
```

```
    "BuysViaMailOrderIndex",
    "RespondsToMailOffersIndex",
    "OptOutMailingsIndex",
    "NonUSTravelIndex",
    "OwnsComputerIndex",
    "HasCreditCardIndex",
    "NewCellphoneUserIndex",
    "NotNewCellphoneUserIndex",
    "OwnsMotorcycleIndex",
    "MadeCallToRetentionTeamIndex",
    "MaritalStatusIndex",
]


# Transform the type of columns to integer
for column in columns_to_int:
    df = df.withColumn(column, df[column].cast("int"))
```

4.3 Các độ đo đánh giá hiệu năng

- ***Accuracy (Độ chính xác):***

Độ chính xác là tỷ lệ giữa số lượng các dự đoán đúng và tổng số mẫu trong tập dữ liệu. Mặc dù độ chính xác có thể là một độ đo hữu ích, nhưng nó không phản ánh chính xác hiệu suất của mô hình khi các lớp trong dữ liệu không cân bằng.

- ***Precision (Độ chính xác của dương tính):***

Precision đo lường tỷ lệ giữa số lượng các dự đoán dương tính đúng và tổng số dự đoán dương tính. Nó là một độ đo của khả năng của mô hình trong việc tránh các dự đoán sai tích cực (false positive). Precision càng cao thì tỷ lệ dự đoán đúng càng cao.

- ***Recall (Tỷ lệ bao phủ):***

Tỷ lệ bao phủ đo lường tỷ lệ giữa số lượng các dự đoán dương tính đúng và tổng số mẫu dương tính trong tập dữ liệu. Nó là một độ đo của khả năng của mô hình trong việc phát hiện tất cả các trường hợp tích cực có sẵn (true positive). Recall càng cao thì tỷ lệ bao phủ càng tốt.

- ***F1-score:***

F1-score là trung bình điều hòa giữa precision và recall. Nó cung cấp một đánh giá tổng thể về hiệu suất của mô hình, đặc biệt là khi có sự không cân bằng giữa các lớp trong tập dữ liệu. F1-score càng cao thì mô hình càng tốt.

- ***ROC Curve (Receiver Operating Characteristic Curve) và AUC (Area Under the Curve):***

Đường cong ROC là biểu đồ của tỷ lệ True Positive Rate (TPR) so với tỷ lệ False Positive Rate (FPR) ở các ngưỡng quyết định khác nhau. AUC là diện tích dưới đường cong ROC và là một độ đo tổng thể của hiệu suất của mô hình. AUC càng cao, mô hình càng tốt trong việc phân loại các mẫu.

- Các độ đo này cung cấp thông tin chi tiết và toàn diện về hiệu suất của mô hình phân loại, giúp ta đánh giá và so sánh giữa các mô hình khác nhau. Trong thư viện scikit-learn, các hàm như classification_report, roc_curve và auc được cung cấp để tính toán và hiển thị các độ đo này.

4.4 Các tham số và môi trường cài đặt

```
# Validation for hyper-parameter tuning.
# Randomly splits the input dataset into train and validation sets,
# and uses evaluation metric on the validation set to select the best model.
rf = RandomForestClassifier(
   featuresCol="features", labelCol="label", predictionCol="prediction",
maxBins=16
)

# Use a ParamGridBuilder to construct a grid of parameters to search over.
# TrainValidationSplit will try all combinations of values and determine best model
using the evaluator.
paramGrid = (
   ParamGridBuilder()
   .addGrid(rf.maxDepth, [8, 10, 12])
   .addGrid(rf.minInstancesPerNode, [1, 3, 5, 10])
   .addGrid(rf.numTrees, [50, 100])
   .build()
)

# In this case the estimator is BinaryClassificationEvaluator
# A TrainValidationSplit requires an Estimator, a set of Estimator ParamMaps, and
an Evaluator.
tvs = TrainValidationSplit(
   estimator=rf,
   estimatorParamMaps=paramGrid,
   evaluator=BinaryClassificationEvaluator(),
   trainRatio=0.8,
)  # 80% of the data will be used for training, 20% for validation.

# Run TrainValidationSplit, and choose the best set of parameters.
model = tvs.fit(train)
```

Trong đoạn mã trên, chúng ta đang thực hiện quá trình tinh chỉnh siêu tham số (hyper-parameter tuning) cho mô hình Random Forest Classifier bằng cách sử dụng kỹ thuật Train-Validation Split. Dưới đây là giải thích chi tiết về các tham số được sử dụng trong quá trình tinh chỉnh:

- RandomForestClassifier: Là một mô hình phân loại dựa trên cây quyết định, trong đó các cây quyết định được sử dụng để xây dựng một tập hợp các cây ngẫu nhiên, và kết quả cuối cùng là sự bỏ phiếu của các cây đó.

- featuresCol: Là tên cột chứa các đặc trưng (features) trong dữ liệu.

- labelCol: Là tên cột chứa nhãn (labels) trong dữ liệu, đây là nhãn dự đoán.

- predictionCol: Là tên cột trong DataFrame kết quả được dự đoán.

- maxBins: Là số lượng tối đa của các bin khi chia các đặc trưng liên tục thành các bin. Đây là một siêu tham số của Random Forest.

- ParamGridBuilder: Là một builder để tạo ra một lưới các tham số để tinh chỉnh. Trong đoạn mã này, chúng ta tạo một lưới các giá trị cho các tham số của mô hình Random Forest như maxDepth, minInstancesPerNode, và numTrees.

- maxDepth: Là độ sâu tối đa của các cây quyết định trong mô hình Random Forest.

- minInstancesPerNode: Số lượng mẫu tối thiểu cần có trong một nút lá của cây quyết định.

- numTrees: Số lượng cây quyết định trong mô hình Random Forest.

- TrainValidationSplit: Là một kỹ thuật tinh chỉnh siêu tham số trong đó tập dữ liệu được chia thành tập huấn luyện và tập validation. Các tham số mô hình được đánh giá dựa trên hiệu suất trên tập validation.

- estimator: Mô hình cần được tinh chỉnh, trong trường hợp này là mô hình Random Forest Classifier.

- estimatorParamMaps: Là một lưới các tham số cần được đánh giá.

- evaluator: Bộ đánh giá sẽ được sử dụng để đánh giá hiệu suất của mô hình trên tập validation. Trong đoạn mã này, chúng ta sử dụng BinaryClassificationEvaluator để đánh giá mô hình dựa trên đánh giá về hiệu suất phân loại nhị phân.

- trainRatio: Tỷ lệ dữ liệu sẽ được sử dụng cho việc huấn luyện, phần còn lại sẽ được sử dụng cho việc validation. Trong đoạn mã này, chúng ta sử dụng tỷ lệ 80-20, tức là 80% dữ liệu sẽ được sử dụng cho huấn luyện và 20% còn lại sẽ được sử dụng cho validation.

- fit(train): Thực hiện quá trình tinh chỉnh siêu tham số bằng cách sử dụng Train-Validation Split trên tập dữ liệu huấn luyện (train). Kết quả là một mô hình tốt nhất được chọn dựa trên hiệu suất trên tập validation.

4.5 Các phương pháp cơ sở

- ***Random Forest Classifier (RFC):***

Random Forest là một phương pháp học máy dựa trên việc xây dựng nhiều cây quyết định trong quá trình huấn luyện và kết hợp kết quả từ tất cả các cây con để đưa ra dự đoán. Mỗi cây quyết định được xây dựng dựa trên một tập dữ liệu con được lấy mẫu từ dữ liệu gốc và sử dụng một phần của các đặc trưng. Sau đó, kết quả từ tất cả các cây con được kết hợp thông qua việc bỏ phiếu để đưa ra dự đoán cuối cùng. Random Forest thường cho kết quả tốt trong nhiều tình huống khác nhau và ít bị ảnh hưởng bởi overfitting.

- ***Gradient Boosted Tree (GBT):***

Gradient Boosted Tree là một phương pháp khác trong học máy, nơi các cây quyết định được tạo ra tuần tự. Mỗi cây trong GBT cố gắng cải thiện lỗi của các cây trước đó bằng cách tập trung vào các mẫu bị sai lệch. GBT thường cho hiệu suất dự đoán tốt hơn so với Random Forest nhưng đồng thời cũng đòi hỏi thời gian huấn luyện lâu hơn và cần nhiều tinh chỉnh tham số hơn.

- ***KMeans Clustering:***

KMeans là một thuật toán phân cụm không giám sát được sử dụng để phân nhóm dữ liệu thành các nhóm khác nhau dựa trên các đặc trưng của chúng. Phương pháp này cố gắng tìm ra các trung tâm cụm sao cho tổng bình phương khoảng cách từ các điểm dữ liệu đến trung tâm cụm là nhỏ nhất có thể. KMeans thường được sử dụng để phân loại khách hàng hoặc nhóm khách hàng dựa trên hành vi hoặc các yếu tố khác trong dữ liệu telecom.

- Các phương pháp cơ sở này đều có thể cung cấp kết quả tốt khi được áp dụng đúng cách và tinh chỉnh tham số phù hợp. Trong đoạn mã, chúng ta đã sử dụng các module và công cụ từ thư viện PySpark và scikit-learn để triển khai và đánh giá các phương pháp này.

4.6 Phân tích so sánh các kết quả

Trong phần này, chúng ta sẽ thực hiện phân tích và so sánh các kết quả thu được từ các phương pháp cơ sở đã được áp dụng. Cụ thể, chúng ta sẽ xem xét các phương pháp Random Forest Classifier (RFC), Gradient Boosted Tree (GBT), và KMeans Clustering. Dưới đây là một số bước mà chúng ta có thể thực hiện trong phân tích này:

- ***Đánh giá hiệu suất của mô hình phân loại (RFC và GBT):***

Sử dụng các độ đo như Accuracy, Precision, Recall, F1-score để đánh giá hiệu suất của mô hình phân loại RFC và GBT trên tập dữ liệu kiểm tra.
Vẽ biểu đồ ROC và tính diện tích dưới đường cong ROC (AUC) để so sánh khả năng phân loại của hai mô hình.

- ***Đánh giá hiệu suất của mô hình phân cụm (KMeans):***

Sử dụng các phương pháp đánh giá như Silhouette Score để đánh giá chất lượng của các cụm được tạo ra bởi KMeans.
So sánh kết quả của KMeans với nhãn thực tế (nếu có) để xác định mức độ phân loại của thuật toán.

- ***So sánh kết quả và lựa chọn mô hình tốt nhất:***

So sánh kết quả hiệu suất của RFC và GBT để xác định mô hình nào hoạt động tốt hơn trên tập dữ liệu đã cho.

Đánh giá tính khả thi của việc sử dụng KMeans để phân cụm khách hàng và xem xét khả năng sử dụng thông tin này trong các chiến lược kinh doanh.

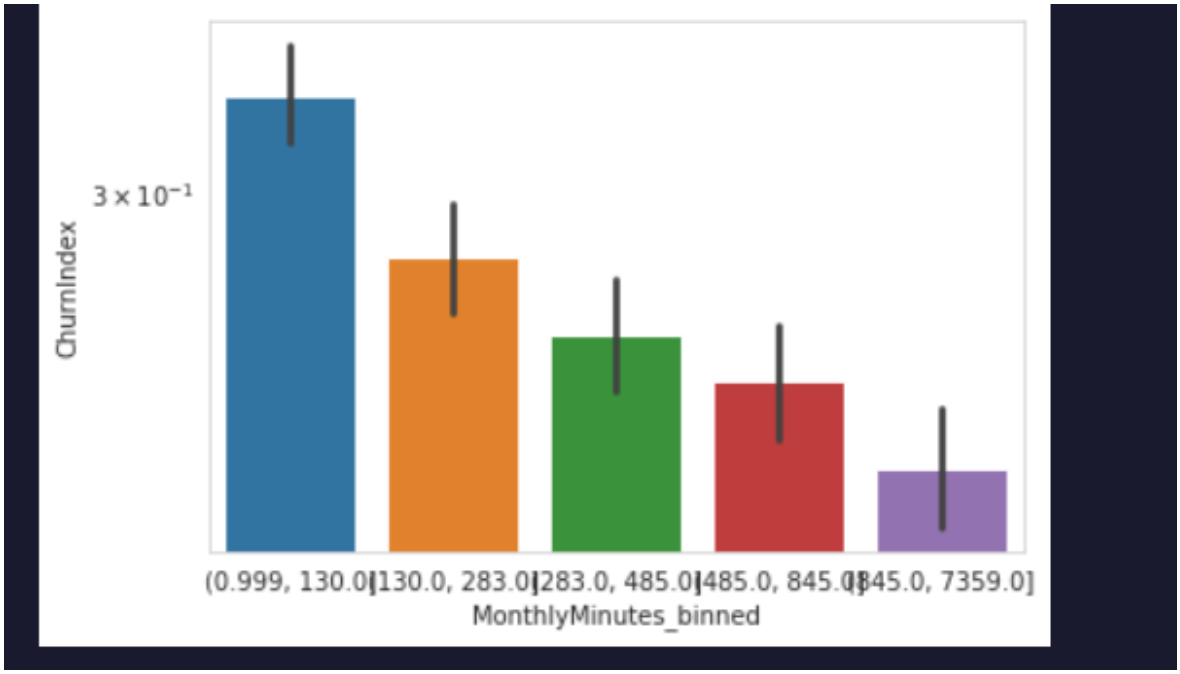
- ***Tinh chỉnh tham số (nếu cần):***

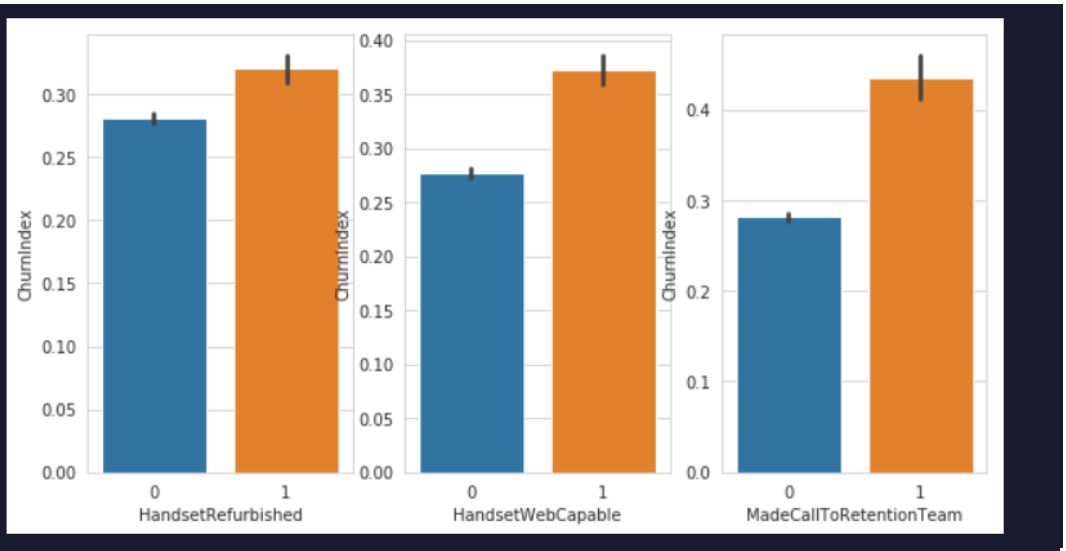
Nếu cần thiết, thực hiện tinh chỉnh tham số cho các mô hình để cải thiện hiệu suất của chúng trên tập dữ liệu kiểm tra.

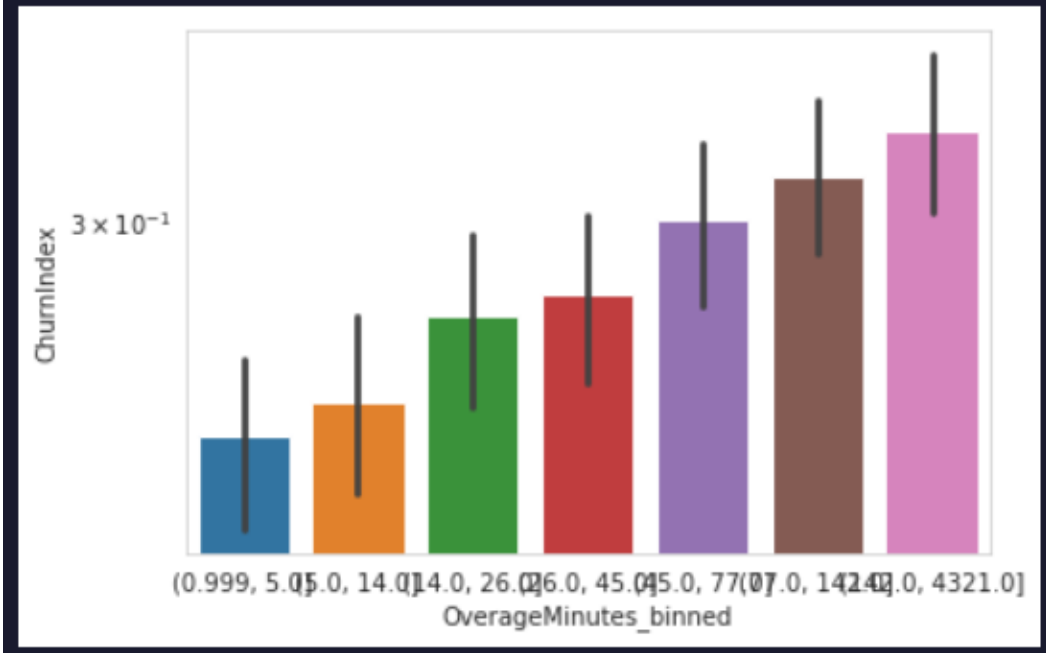
V) Kết luận

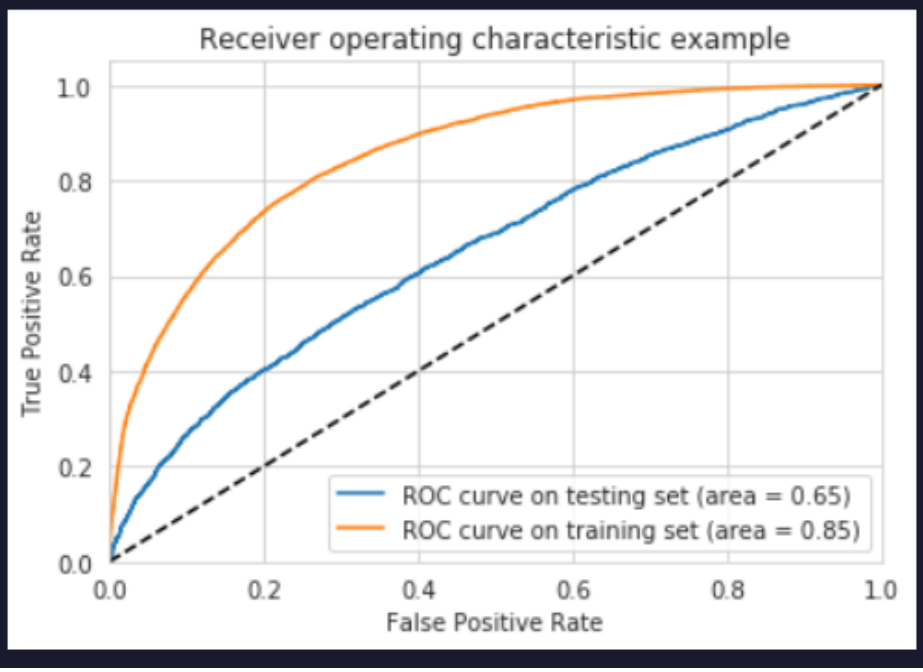
Tổng kết kết quả của phân tích và so sánh để xác định mô hình tốt nhất cho bài toán telecom cụ thể này.

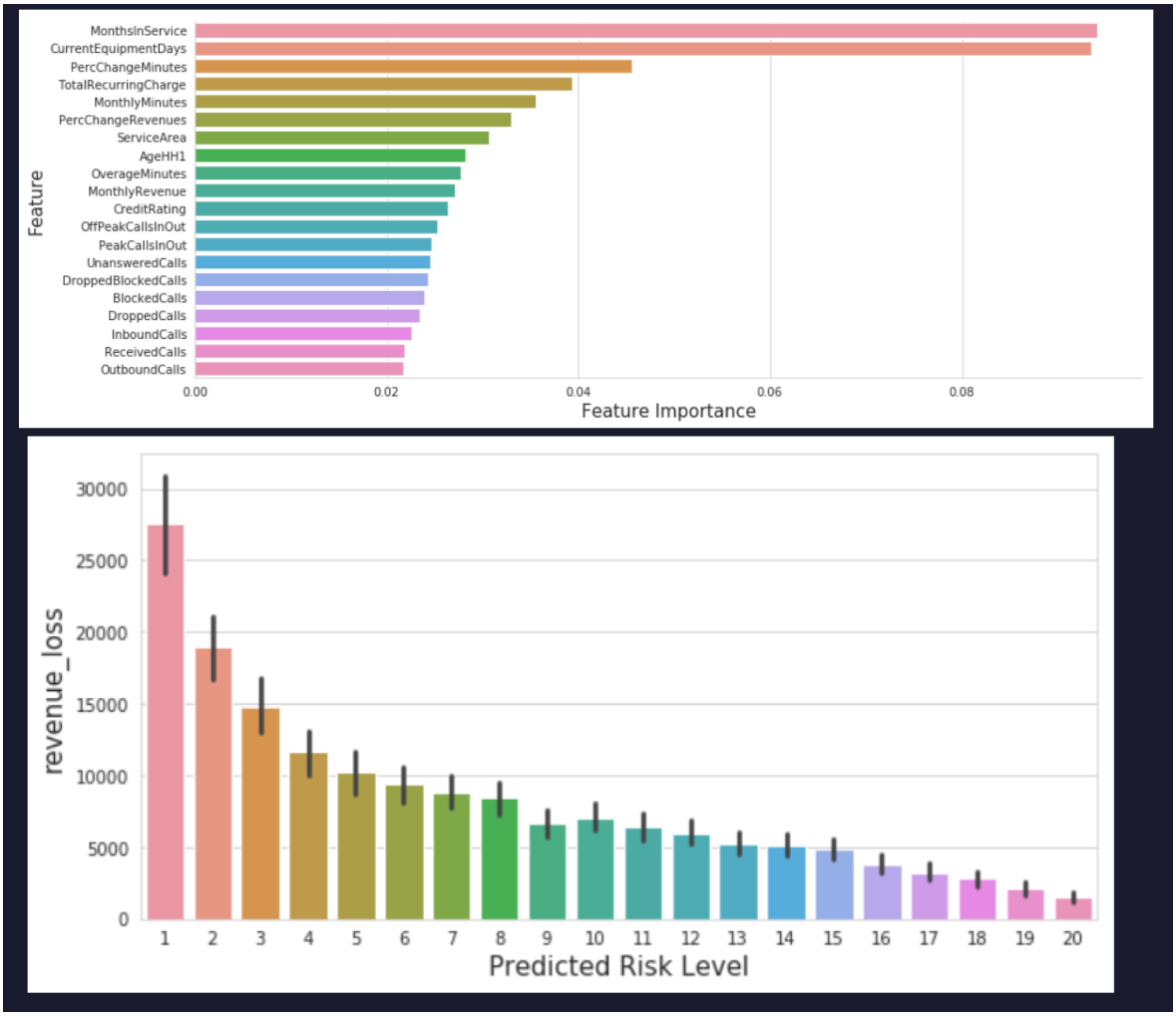
Bằng cách thực hiện các bước trên, chúng ta có thể đánh giá và so sánh hiệu suất của các phương pháp cơ sở khác nhau và lựa chọn mô hình phù hợp nhất cho bài toán.




ChurnIndex
HandsetRefurbished
HandsetWebCapable
MadeCallToRetentionTeam
0
1


ChurnIndex
3×10^{-1}
OverageMinutes_binned
(0.999, 5.0]
(14.0, 26.0]
(26.0, 45.0]
(45.0, 77.0]
(77.0, 142.0]
(142.0, 4321.0]


Receiver operating characteristic example
True Positive Rate
False Positive Rate
ROC curve on testing set (area = 0.65)
ROC curve on training set (area = 0.85)


MonthsInService
CurrentEquipmentDays
PercChangeMinutes
TotalRecurringCharge
MonthlyMinutes
PercChangeRevenues
ServiceArea
AgeHH1
OverageMinutes
MonthlyRevenue
CreditRating
OffPeakCallsInOut
PeakCallsInOut
UnansweredCalls
DroppedBlockedCalls
BlockedCalls
DroppedCalls
InboundCalls
ReceivedCalls
OutboundCalls
Feature
0.00
0.02
0.04
0.06
0.08
Feature Importance
revenue_loss
30000
25000
20000
15000
10000
5000
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Predicted Risk Level