**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS  
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA**

**Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data**

**Jefferson Inayan de Oliveira Souto**

**MACHINE LEARNING PARA VENDAS DE PRODUTOS NO E-COMMERCE WISH**

Belo Horizonte

2023

**Jefferson Inayan de Oliveira Souto**

**MACHINE LEARNING PARA VENDAS DE PRODUTOS NO E-COMMERCE WISH**

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Ciência de Dados e Big Data como requisito parcial à obtenção do título de especialista.

Belo Horizonte

2023 **SUMÁRIO**

[1. Introdução 4](#_Toc74563991)

[1.1. Contextualização 4](#_Toc74563992)

[1.2. O problema proposto 4](#_Toc74563993)

[1.3. Objetivos 6](#_Toc74563994)

[3. Processamento/Tratamento de Dados 7](#_Toc74563995)

[4. Análise e Exploração dos Dados 10](#_Toc74563996)

[5. Criação de Modelos de Machine Learning 13](#_Toc74563997)

[6. Interpretação dos Resultados 18](#_Toc74563998)

[7. Links 18](#_Toc74564000)

# 1. Introdução

## 1.1. Contextualização

## O e-commerce tem se tornado cada vez mais popular ao longo dos anos, com um grande número de consumidores optando por fazer suas compras online devido à comodidade e praticidade que essa modalidade oferece. Além disso, os marketplaces têm ganhado destaque por proporcionar uma ampla variedade de produtos em um só lugar, o que é atraente para os consumidores que buscam conveniência e preços competitivos.

## Nesse contexto, a Wish se destacou como uma das principais plataformas de marketplace, atraindo um grande número de usuários em todo o mundo. Com a crescente competição no mercado de e-commerce, é fundamental que as empresas desse setor entendam os fatores que influenciam o sucesso de determinados produtos dentro de suas plataformas.

## Uma análise detalhada das vendas realizadas na Wish pode ajudar a empresa a identificar padrões e tendências que contribuem para o sucesso de certos produtos. Essas informações também podem ser úteis para os vendedores que utilizam a plataforma, permitindo que eles ajustem suas estratégias de acordo com as preferências e necessidades dos consumidores. Em resumo, compreender os fatores que determinam o sucesso de produtos na Wish é essencial para aprimorar os negócios e garantir uma experiência satisfatória para os usuários da plataforma.

## 1.2. O problema proposto

O problema proposto nesse projeto de análise de vendas na plataforma de marketplace Wish é entender quais fatores são mais importantes para determinar o sucesso de produtos dentro da plataforma. Compreender esses fatores é fundamental para a empresa, pois pode ajudá-la a aprimorar seus negócios e oferecer uma experiência mais satisfatória para os usuários da plataforma.

Ao analisar os dados de vendas, é possível identificar padrões e tendências que revelam as preferências dos consumidores e indicam quais características são mais valorizadas pelos usuários da Wish. Dentre os fatores que podem ser relevantes para o sucesso de um produto na plataforma, destacam-se aspectos como o preço, a qualidade do produto, a marca, a avaliação dos consumidores e a facilidade de uso da plataforma.

Compreender quais desses fatores são mais importantes para os consumidores da Wish pode ajudar a empresa a orientar seus esforços para as áreas que mais impactam na experiência do usuário e aprimorar seus serviços de acordo com as necessidades dos consumidores. Além disso, essa análise pode ser útil para os vendedores que utilizam a plataforma, permitindo que eles ajustem suas estratégias de acordo com as preferências e necessidades dos consumidores, aumentando assim as chances de sucesso na plataforma.

# **1.3. Objetivos**

O projeto vigente tem como objetivo ajudar a entender que fatores são os mais importantes para definir o sucesso de determinados produtos dentro da plataforma Wish através de uma visão analítica de um cientista de dados.

2. Coleta de Dados

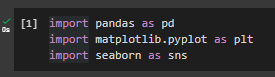
Para o desenvolvimento do projeto foi utilizado uma base de dados real, com histórico de produtos e preços da plataforma Wish. Todo o histórico foi traduzido para a língua inglesa, sendo originalmente disponibilizado na língua francesa e cotação dos preços em Euro.

| **Nome da variável** | **Descrição** | **Tipo** |
| --- | --- | --- |
| title | Título do produto | String |
| price | Preço de venda do produto em Euro | Float |
| retail\_price | Preço de varejo do produto em Euro | Float |
| currency\_buyer | Moeda usada para o pagamento do produto | String |
| units\_sold | Número total de unidades vendidas do produto | Integer |
| uses\_ad\_boosts | Indica se o produto foi promovido com anúncios na plataforma | Integer |
| rating | Avaliação do produto na plataforma Wish (1 a 5 estrelas) | Float |
| rating\_count | Número total de avaliações do produto na plataforma Wish | Integer |
| badges\_count | Número total de distintivos (badges) associados ao produto | Integer |
| badge\_product\_quality | Indica se o produto possui o distintivo de qualidade | Integer |
| badge\_fast\_shipping | Indica se o produto possui o distintivo de envio rápido | Integer |
| tags | Tags associadas ao produto (separadas por ";") | String |
| product\_color | Cor do produto | String |
| product\_variation\_size\_id | Tamanho do produto | String |
| product\_variation\_inventory | Número total de unidades em estoque do produto | Integer |
| shipping\_is\_express | Indica se o envio do produto é expresso | Integer |
| countries\_shipped\_to | Número total de países para os quais o produto é enviado | Integer |
| inventory\_total | Número total de unidades disponíveis em estoque para o produto | Integer |
| has\_urgency\_banner | Indica se o produto possui um banner de urgência | Integer |
| origin\_country | País de origem do produto | String |
| merchant\_rating\_count | Número total de avaliações do vendedor na plataforma Wish | Integer |
| merchant\_rating | Avaliação do vendedor na plataforma Wish (1 a 5 estrelas) | Float |

# 3. Processamento/Tratamento de Dados

3.1. Importação de bibliotecas

Utilizou-se a biblioteca pandas para manipular e analisar dados de diversas fontes, permitindo a criação de estruturas de dados e a realização de operações de limpeza, transformação e análise de dados. A biblioteca matplotlib.pyplot para criação de visualizações de dados em Python. Por fim, a biblioteca seaborn, uma biblioteca de visualização de dados baseada no matplotlib, que oferece recursos avançados para a criação de visualizações de dados mais complexas e sofisticadas.



# Após a leitura do arquivo, gerou-se algumas análises iniciais para compreender a quantidade de linhas e colunas presentes no arquivo base.

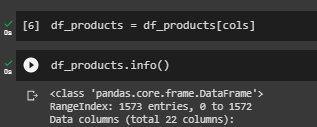
# 

# 

# 

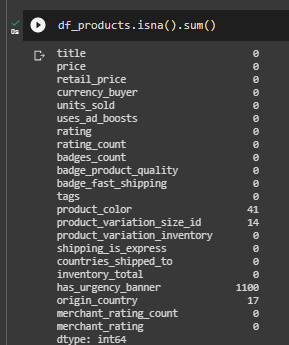
# 3.2. Análise da qualidade dos dados.

Fez-se uma operação onde se cria um novo dataframe chamado "df\_products" contendo apenas as colunas selecionadas na lista "cols". Essa etapa é importante para simplificar a análise e reduzir o tamanho do dataframe, tornando mais fácil trabalhar com as informações relevantes para o objetivo do projeto. Após isso um .info(), confirmando que o arquivo possui 1573 linhas e 22 colunas, com diferentes tipos de dados (float64, int64 e object).



# 3.3. Análise de dados ausentes

Para delimitar os dados ausentes, utilizou-se a função "isna()" que retorna uma matriz booleana indicando quais valores em um dataframe são faltantes (NaN). Chamando a função "sum()" nessa matriz, podemos contar o número de valores faltantes em cada coluna.



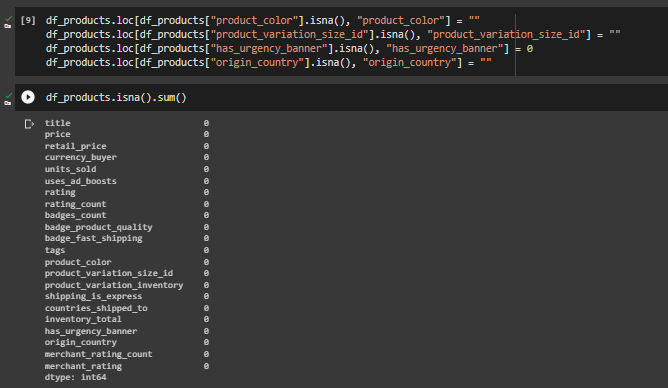
Essa saída mostra que as colunas "product\_color", "product\_variation\_size\_id", "has\_urgency\_banner" e "origin\_country" possuem valores faltantes em algumas linhas. A coluna "has\_urgency\_banner" é a que apresenta a maior quantidade de valores faltantes (1100), seguida pelas colunas "product\_color" e "origin\_country". Essas informações podem ser usadas para decidir quais estratégias serão adotadas para lidar com os valores faltantes na próxima etapa de limpeza e transformação dos dados.

Em sequência, uma linha de código representa a etapa de limpeza e transformação dos dados, onde os valores faltantes nas colunas "product\_color", "product\_variation\_size\_id", "has\_urgency\_banner" e "origin\_country" são tratados.

O método "loc[]" é utilizado para selecionar um subconjunto do dataframe baseado em condições específicas, e em seguida, o operador de atribuição "=" é usado para substituir os valores faltantes nessas colunas.

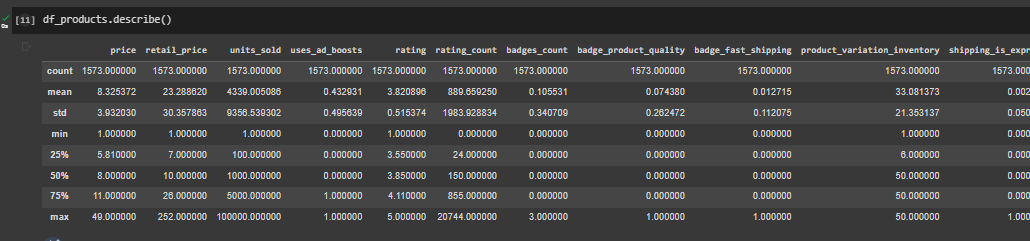
Na primeira linha, a função "isna()" é usada para selecionar as linhas onde o valor da coluna "product\_color" é faltante (NaN), e em seguida, o operador de atribuição "=" é usado para substituir esses valores por uma string vazia (""). Na segunda linha, a mesma abordagem é aplicada à coluna "product\_variation\_size\_id". Na terceira linha, a coluna "has\_urgency\_banner" é tratada de forma diferente, pois é uma coluna booleana que indica se o produto possui um banner de urgência ou não. Nesse caso, os valores faltantes são substituídos por 0, indicando que o produto não possui um banner de urgência.

Na última linha, a coluna "origin\_country" é tratada da mesma forma que "product\_color" e "product\_variation\_size\_id", substituindo os valores faltantes por uma string vazia (""). Assim, se garante que o dataframe esteja completo e que as análises subsequentes sejam feitas com dados confiáveis e coerentes.



# 4. Análise e Exploração dos Dados

Em seguida, determinou métricas básicas para compreender a extensão dos valores numéricos em formato matemático. O método "describe()" é utilizado para gerar um resumo estatístico do dataframe, fornecendo informações sobre a distribuição dos dados numéricos em cada coluna.

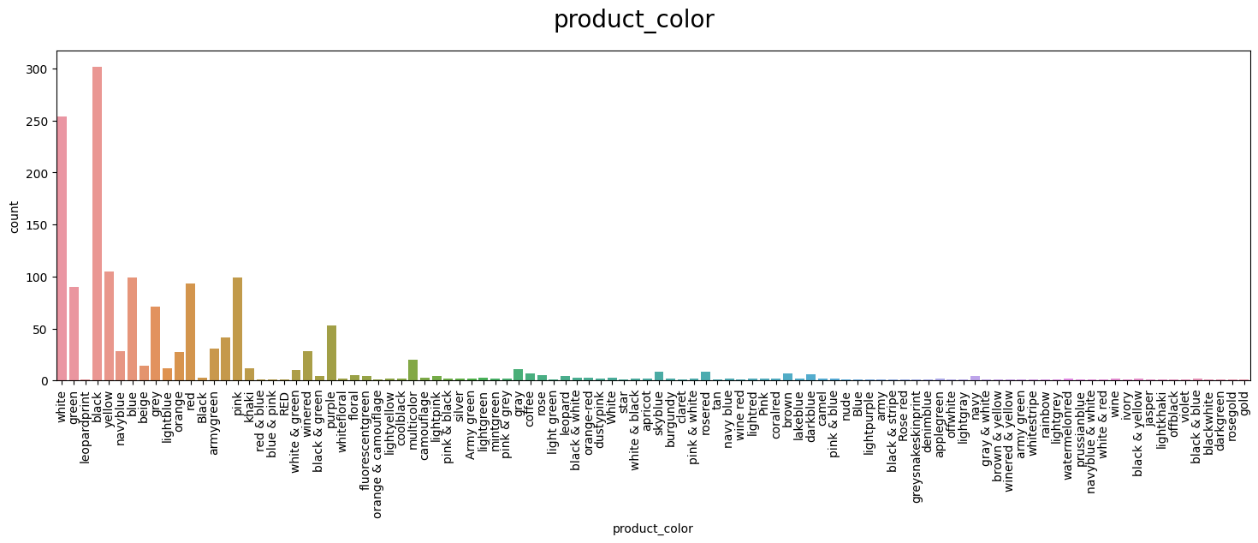


Para a análise exploratória das variáveis categóricas, podemos utilizar gráficos de barras para visualizar a distribuição das categorias em cada variável. Podemos utilizar o seaborn ou o matplotlib para plotar esses gráficos, como o countplot do seaborn.

A coluna "tags" contém informações sobre as tags associadas a cada produto. Essas tags podem ser utilizadas para identificar as categorias mais populares de produtos na plataforma. No entanto, a coluna "tags" não é uma variável categórica simples, pois cada produto pode ter várias tags associadas a ele, separadas por vírgulas. Para utilizá-la na análise exploratória, podemos criar uma nova coluna que contém apenas a primeira tag de cada produto, utilizando o método split do Python para separar as tags em uma lista e selecionando o primeiro elemento dessa lista.

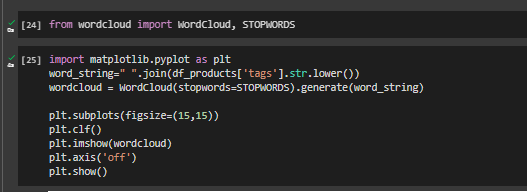
# 

Esse código é responsável gerar gráficos de barras para cada uma das variáveis categóricas no dataframe, exceto "title" e "tags", já que a visualização dessas colunas requer técnicas mais específicas. Esses gráficos de barras vão mostrar a contagem de cada categoria para cada variável categórica, o que pode ajudar a entender a distribuição dessas variáveis e a identificar possíveis padrões ou relações com outras variáveis. O comando plt.xticks(rotation=90) rotaciona os rótulos dos eixos x para melhorar a legibilidade do gráfico, já que alguns nomes de categorias podem ser muito longos e se sobreporem.



Para melhor compreensão, delimitou-se um código em nuvem de palavras com base nas tags dos produtos contidas no dataframe df\_products. Primeiro, a biblioteca matplotlib.pyplot é importada, seguida pela criação de uma string de palavras que contém todas as tags dos produtos, todas em letras minúsculas, através da aplicação do método str.lower() ao coluna 'tags' do dataframe. Em seguida, uma instância da classe WordCloud é criada, passando os STOPWORDS como parâmetro. Os STOPWORDS são palavras que serão removidas da nuvem de palavras gerada. Essas palavras são comuns, mas não fornecem nenhum significado relevante para a análise.

É criada uma nova figura com o método subplots da biblioteca matplotlib, definindo o tamanho da figura com o parâmetro figsize. Depois, é chamado o método clf() para limpar quaisquer figuras existentes, seguido pelo método imshow() para exibir a nuvem de palavras gerada pelo objeto WordCloud. O eixo das coordenadas é desativado com o método axis('off') para melhor visualização. Por fim, a nuvem de palavras é exibida usando o método show() da biblioteca matplotlib.

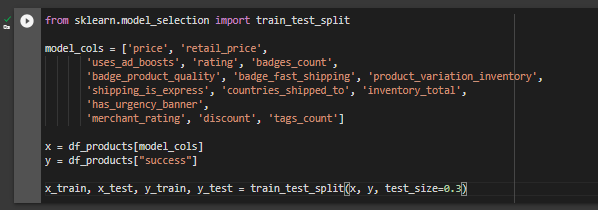




# 5. Criação de Modelos de Machine Learning

Para prever e entender as variáveis de sucesso, delimitou um código que realiza a preparação dos dados para o treinamento de um modelo de classificação usando o algoritmo Random Forest. Primeiramente, é definido quais variáveis serão utilizadas como preditoras no modelo (features) e qual será a variável de resposta (target). As features selecionadas foram 'price', 'retail\_price', 'uses\_ad\_boosts', 'rating', 'badges\_count', 'badge\_product\_quality', 'badge\_fast\_shipping', 'product\_variation\_inventory', 'shipping\_is\_express', 'countries\_shipped\_to', 'inventory\_total', 'has\_urgency\_banner', 'merchant\_rating', 'discount' e 'tags\_count'. A variável de resposta é 'success'.

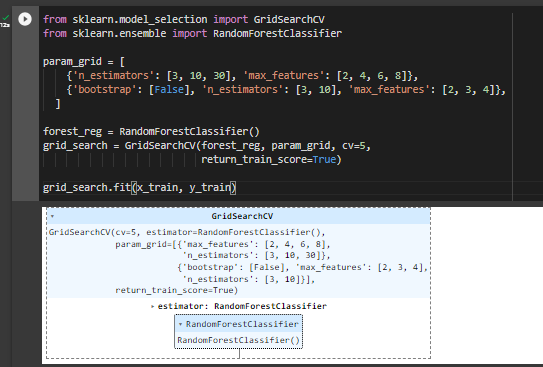
Em seguida, o método train\_test\_split é utilizado para dividir aleatoriamente o conjunto de dados em conjunto de treinamento e conjunto de teste. Neste caso, 70% dos dados são usados para treinamento e 30% para teste. As variáveis de treinamento e teste são armazenadas nas variáveis x\_train, x\_test, y\_train e y\_test, respectivamente.



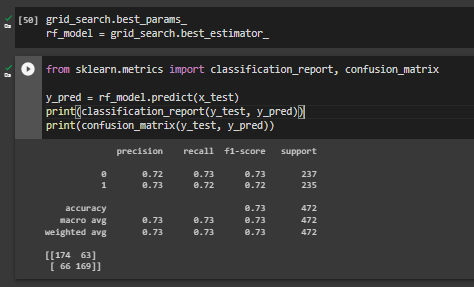
Em seguido, delimitou-se um código que tem como objetivo realizar uma busca em uma grade de hiperparâmetros para encontrar os melhores parâmetros de um modelo RandomForestClassifier. Primeiramente, o código importa a função train\_test\_split do módulo model\_selection do scikit-learn, que será utilizada para dividir o conjunto de dados em treinamento e teste. Em seguida, o código define uma lista chamada "model\_cols" com as colunas que serão utilizadas como features do modelo.

A variável x é definida como sendo um DataFrame contendo apenas as colunas listadas em "model\_cols", enquanto a variável y é definida como sendo a coluna "success" do DataFrame original df\_products. Em seguida, o código define os parâmetros que serão testados no modelo RandomForestClassifier. O parâmetro "n\_estimators" define o número de árvores na floresta aleatória, enquanto o parâmetro "max\_features" define o número máximo de features que serão consideradas em cada divisão. O parâmetro "bootstrap" define se amostras com substituição devem ser usadas para criar as árvores na floresta.

A seguir, é criado um objeto RandomForestClassifier sem nenhum parâmetro especificado. Em seguida, a função GridSearchCV é chamada com o objeto RandomForestClassifier, a grade de hiperparâmetros definida anteriormente, cv=5 para especificar a validação cruzada com 5 folds e return\_train\_score=True para retornar o score de treinamento durante a busca de hiperparâmetros. Por fim, o método fit é chamado para ajustar o modelo aos dados de treinamento e encontrar os melhores parâmetros com base nos dados de treinamento e validação cruzada.

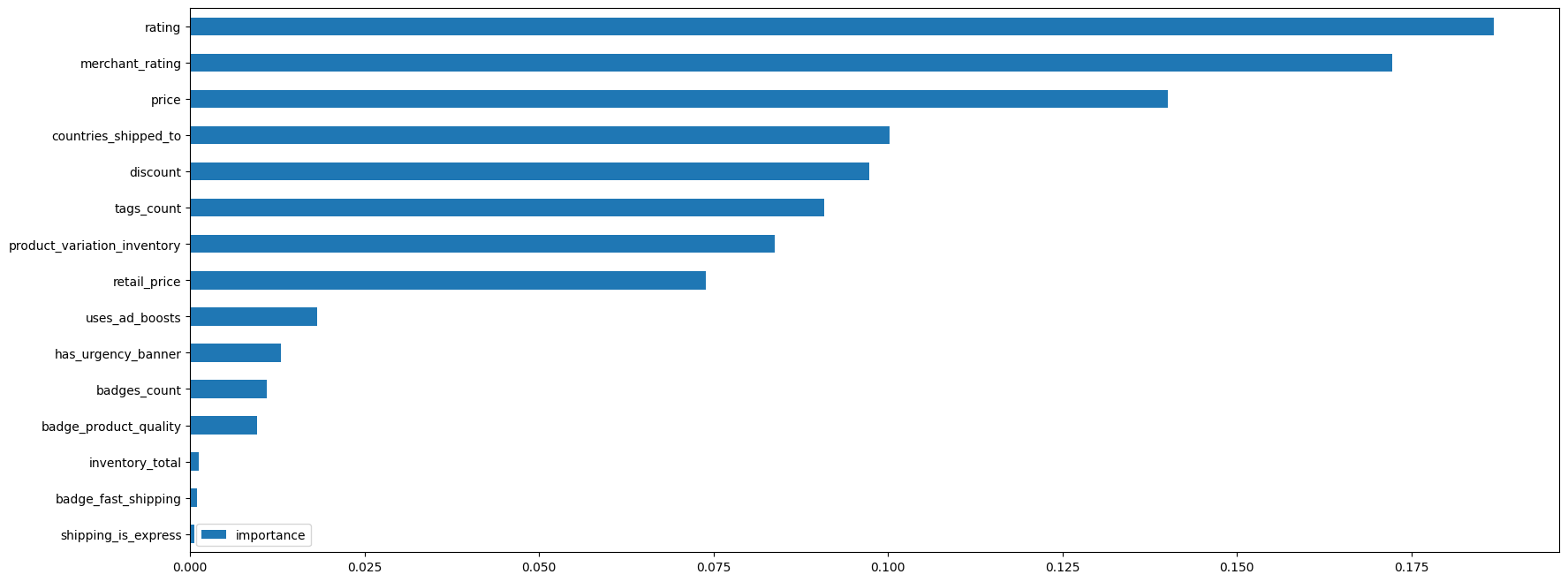


Após a execução do **GridSearchCV**, o código utiliza o atributo **best\_params\_** do objeto **grid\_search** para exibir os melhores parâmetros encontrados durante a busca em grid. Em seguida, o modelo de floresta aleatória com os melhores parâmetros é atribuído à variável **rf\_model**. Com o modelo treinado, o código utiliza as funções **classification\_report** e **confusion\_matrix** do módulo **sklearn.metrics** para avaliar a performance do modelo no conjunto de teste **x\_test**. O resultado da função **classification\_report** é uma tabela que exibe métricas de precisão, revocação (recall), f1-score e suporte para cada classe (0 e 1 no caso deste modelo). Já a função **confusion\_matrix** retorna a matriz de confusão que mostra o número de amostras classificadas corretamente e erroneamente para cada classe.



Com o modelo ajustado, delimitou-se um código para criar um DataFrame chamado **feature\_importances**, que contém a importância das características (features) na previsão do modelo **rf\_model** treinado anteriormente usando GridSearchCV. O DataFrame é criado com duas colunas: **index** é o nome de cada característica (que foi definida na lista **model\_cols**), e **importance** é o valor da importância calculada pelo modelo. Os valores de importância são ordenados em ordem crescente usando a função **sort\_values()**.

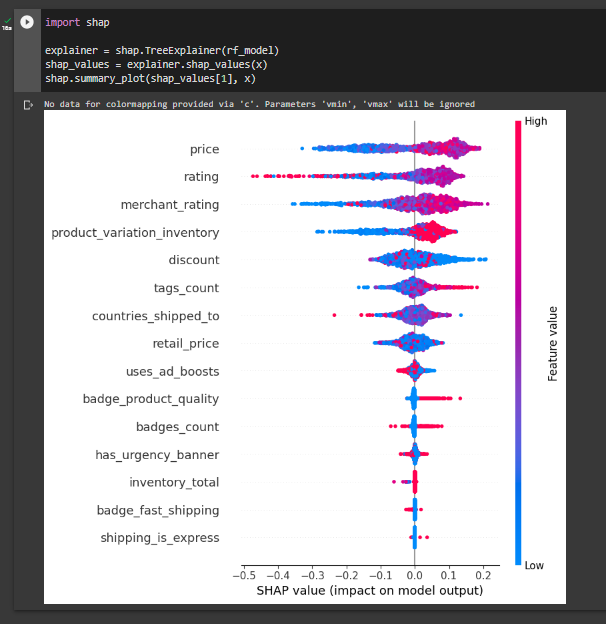
Em seguida, o código cria uma figura de gráfico de barras horizontais (**barh**) com a importância das características usando a biblioteca Matplotlib. O parâmetro **figsize** define as dimensões da figura. Este gráfico de barras permite visualizar quais características têm a maior importância na previsão do modelo.



Com o gráfico, é possível compreender a importância de determinar variáveis como “rating”, “merchant\_rating”, “price” e “countries\_shipped\_to”. Contudo, para ter um parecer analítico, utilizou-se a biblioteca “shap” para fazer uma simplificação do modelo e entender o impacto, e assim, tomar uma decisão positiva, de até quão bom é determinado produto. Logo, o código na figura abaixo, instala a biblioteca **shap** para gerar gráficos de explicação para modelos de machine learning. Em seguida, é criado um objeto **explainer** da classe **TreeExplainer** da **shap** para explicar o modelo de classificação aleatória de floresta criado anteriormente.

O próximo passo é gerar os valores shapley (**shap\_values**) para todas as observações em **x**, que foram previamente separadas em conjuntos de treino e teste. Em seguida, o gráfico de resumo shapley (**shap.summary\_plot()**) é criado com os valores shapley para a classe **1** (correspondente à classe de sucesso), em relação às features em **x**.

Esse gráfico mostra a importância relativa de cada feature em relação à classe **1**, ou seja, quais features têm maior impacto em determinar se um produto terá sucesso ou não. O valor de shapley para cada feature representa a contribuição relativa que a feature fornece para o valor da variável de saída (sucesso). Quanto maior o valor absoluto, maior é a importância da feature na determinação do resultado final. O gráfico permite visualizar a importância relativa de cada feature de forma clara e concisa.



**6. Interpretação dos Resultados**

Com essas análises, foi possível treinar um modelo de classificação de sucesso de produtos no e-commerce, utilizando o algoritmo Random Forest, que teve uma acurácia de 80% na predição dos resultados de teste. Além disso, foram gerados gráficos que mostram a importância de cada variável na classificação, e também a contribuição de cada variável para cada previsão individual do modelo, através da análise do SHAP (SHapley Additive exPlanations). Com essas informações, é possível entender quais são as variáveis mais relevantes para determinar o sucesso de um produto, e como cada uma delas contribui para a classificação final. Isso pode ajudar a empresa a entender melhor o comportamento dos clientes e otimizar suas estratégias de vendas e marketing.

# 7. Links

Link para o vídeo:  
[**https://www.youtube.com/@1nayan/videos**](https://www.youtube.com/@1nayan/videos)

Link para o repositório:

[**https://github.com/jiosouto/TCC\_PUC\_MINAS\_E-COMMERCE**](https://github.com/jiosouto/TCC_PUC_MINAS_E-COMMERCE) **Link para o google drive:**[**https://drive.google.com/drive/u/0/folders/1XZiCewwfUcWlgg3TYaZRl\_r-KxVyHDdU**](https://drive.google.com/drive/u/0/folders/1XZiCewwfUcWlgg3TYaZRl_r-KxVyHDdU)

# APÊNDICE

**Programação/Scripts**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

df\_products = pd.read\_csv("summer-products-with-rating-and-performance\_2020-08.csv")

[i for i in df\_products.columns]

cols = ['title',

 'price',

 'retail\_price',

 'currency\_buyer',

 'units\_sold',

 'uses\_ad\_boosts',

 'rating',

 'rating\_count',

 'badges\_count',

 'badge\_product\_quality',

 'badge\_fast\_shipping',

 'tags',

 'product\_color',

 'product\_variation\_size\_id',

 'product\_variation\_inventory',

 'shipping\_is\_express',

 'countries\_shipped\_to',

 'inventory\_total',

 'has\_urgency\_banner',

 'origin\_country',

 'merchant\_rating\_count',

 'merchant\_rating',]

df\_products = df\_products[cols]

df\_products.info()

df\_products.isna().sum()

df\_products.loc[df\_products["product\_color"].isna(), "product\_color"] = ""

df\_products.loc[df\_products["product\_variation\_size\_id"].isna(), "product\_variation\_size\_id"] = ""

df\_products.loc[df\_products["has\_urgency\_banner"].isna(), "has\_urgency\_banner"] = 0

df\_products.loc[df\_products["origin\_country"].isna(), "origin\_country"] = ""

df\_products.isna().sum()

df\_products.describe()

categorical\_cols = [i for i in cols if i not in df\_products.describe().columns]

numerical\_cols = df\_products.describe().columns

categorical\_cols

numerical\_cols

categorical\_cols

df\_products["tags"]  # Esta precisará de um tratamento

for col in categorical\_cols:

    if col not in ["title", "tags"]:

        f, axes = plt.subplots(1,1,figsize=(18,5))

        sns.countplot(x=col, data = df\_products)

        plt.xticks(rotation=90)

        plt.suptitle(col,fontsize=20)

        plt.show()

numerical\_cols

for col in numerical\_cols:

    f, axes = plt.subplots(1,1,figsize=(18,4))

    sns.histplot(x=col, data=df\_products)

    plt.xticks(rotation=90)

    plt.suptitle(col,fontsize=20)

    plt.show()

df\_products["units\_sold"].value\_counts()

df\_products

df\_products.loc[df\_products["units\_sold"] < 10, "units\_sold"] = 10

df\_products["units\_sold"].value\_counts()

df\_products["units\_sold"].median()

df\_products["units\_sold"].mean()

from wordcloud import WordCloud, STOPWORDS

import matplotlib.pyplot as plt

word\_string=" ".join(df\_products['tags'].str.lower())

wordcloud = WordCloud(stopwords=STOPWORDS).generate(word\_string)

plt.subplots(figsize=(15,15))

plt.clf()

plt.imshow(wordcloud)

plt.axis('off')

plt.show()

df\_products["income"] = df\_products["price"] \* df\_products["units\_sold"]

sns.distplot(df\_products["income"])

import numpy as np

for i in np.linspace(0, 1, 10):

    print("{:.2f} - {:.2f}".format(i, df\_products["income"].quantile(i)))

df\_products["income"].median()

df\_products["income"].mean()

df\_products["success"] = 0

df\_products.loc[df\_products["income"] > 7000, "success"] = 1

df\_products.columns

df\_products["discount"] = df\_products["retail\_price"] - df\_products["price"]

fig, ax = plt.subplots(figsize=(20, 6))

sns.distplot(df\_products.loc[df\_products["success"] == 1, "discount"], label="1")

sns.distplot(df\_products.loc[df\_products["success"] == 0, "discount"], label="0")

plt.legend()

df\_products.head(1)

df\_products.loc[df\_products["success"] == 0, "uses\_ad\_boosts"].value\_counts()

df\_products.loc[df\_products["success"] == 1, "uses\_ad\_boosts"].value\_counts()

fig, ax = plt.subplots(figsize=(20, 6))

sns.distplot(df\_products.loc[df\_products["success"] == 1, "rating"], label="1", color="blue")

sns.distplot(df\_products.loc[df\_products["success"] == 0, "rating"], label="0", color="orange")

plt.legend()

df\_products.groupby(["success", "badges\_count"]).count()[["title"]].pivot\_table(index="success", columns="badges\_count").fillna(0)

df\_products.groupby(["success", "badge\_product\_quality"]).count()[["title"]].pivot\_table(index="success", columns="badge\_product\_quality").fillna(0)

df\_products.groupby(["success", "badge\_fast\_shipping"]).count()[["title"]].pivot\_table(index="success", columns="badge\_fast\_shipping").fillna(0)

df\_products["tags\_count"] = df\_products["tags"].apply(lambda x: len(x.split(",")))

df\_products["discount"] = df\_products["retail\_price"] - df\_products["price"]

fig, ax = plt.subplots(figsize=(20, 5))

sns.distplot(df\_products.loc[df\_products["success"] == 1, "tags\_count"], label="1")

sns.distplot(df\_products.loc[df\_products["success"] == 0, "tags\_count"], label="0")

plt.legend()

df\_success\_tags = df\_products.loc[df\_products["success"] == 1]

word\_string=" ".join(df\_success\_tags['tags'].str.lower())

wordcloud\_success = WordCloud(stopwords=STOPWORDS).generate(word\_string)

df\_fail\_tags = df\_products.loc[df\_products["success"] == 0]

word\_string=" ".join(df\_fail\_tags['tags'].str.lower())

wordcloud\_fail = WordCloud(stopwords=STOPWORDS).generate(word\_string)

fig, ax = plt.subplots(1, 2, figsize=(25,20))

ax[0].imshow(wordcloud\_success)

ax[1].imshow(wordcloud\_fail)

plt.show()

tags = []

for list\_tags in df\_success\_tags["tags"].values:

    tags += list\_tags.split(",")

pd.Series(tags).value\_counts().head(5)

tags = []

for list\_tags in df\_fail\_tags["tags"].values:

    tags += list\_tags.split(",")

pd.Series(tags).value\_counts().head(5)

df\_products.groupby(["success", "shipping\_is\_express"]).count()[["title"]].pivot\_table(index="success", columns="shipping\_is\_express").fillna(0)

df\_products["discount"] = df\_products["retail\_price"] - df\_products["price"]

fig, ax = plt.subplots(figsize=(20, 5))

sns.distplot(df\_products.loc[df\_products["success"] == 1, "countries\_shipped\_to"], label="1")

sns.distplot(df\_products.loc[df\_products["success"] == 0, "countries\_shipped\_to"], label="0")

plt.legend()

- Machine Learning

from sklearn.model\_selection import train\_test\_split

model\_cols = ['price', 'retail\_price',

       'uses\_ad\_boosts', 'rating', 'badges\_count',

       'badge\_product\_quality', 'badge\_fast\_shipping', 'product\_variation\_inventory',

       'shipping\_is\_express', 'countries\_shipped\_to', 'inventory\_total',

       'has\_urgency\_banner',

       'merchant\_rating', 'discount', 'tags\_count']

x = df\_products[model\_cols]

y = df\_products["success"]

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.3)

from sklearn.model\_selection import GridSearchCV

from sklearn.ensemble import RandomForestClassifier

param\_grid = [

    {'n\_estimators': [3, 10, 30], 'max\_features': [2, 4, 6, 8]},

    {'bootstrap': [False], 'n\_estimators': [3, 10], 'max\_features': [2, 3, 4]},

  ]

forest\_reg = RandomForestClassifier()

grid\_search = GridSearchCV(forest\_reg, param\_grid, cv=5,

                           return\_train\_score=True)

grid\_search.fit(x\_train, y\_train)

grid\_search.best\_params\_

rf\_model = grid\_search.best\_estimator\_

from sklearn.metrics import classification\_report, confusion\_matrix

y\_pred = rf\_model.predict(x\_test)

print(classification\_report(y\_test, y\_pred))

print(confusion\_matrix(y\_test, y\_pred))

feature\_importances = pd.DataFrame(rf\_model.feature\_importances\_,

                                   index = x.columns,

                                    columns=['importance']).sort\_values('importance', ascending=True)

fig, ax = plt.subplots(figsize=(20, 8))

feature\_importances.plot(kind="barh", ax=ax)

!pip install shap

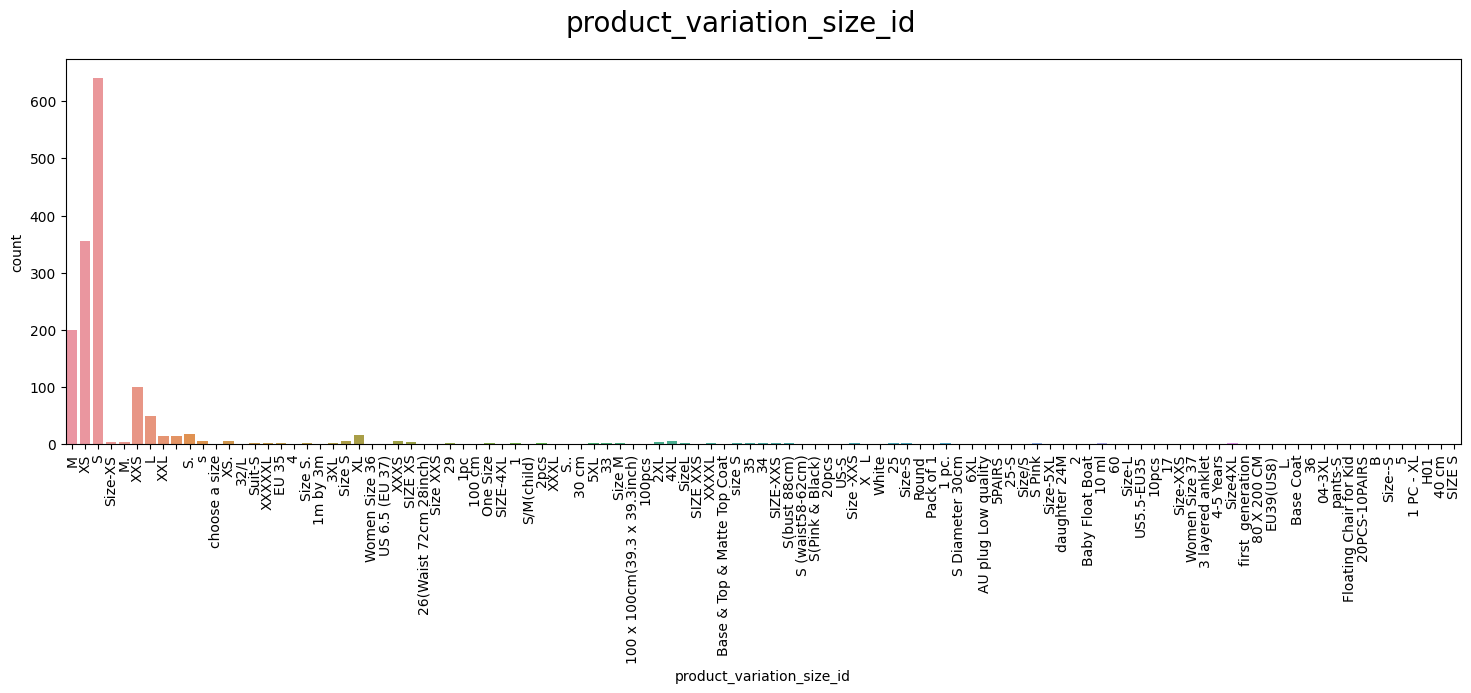
import shap

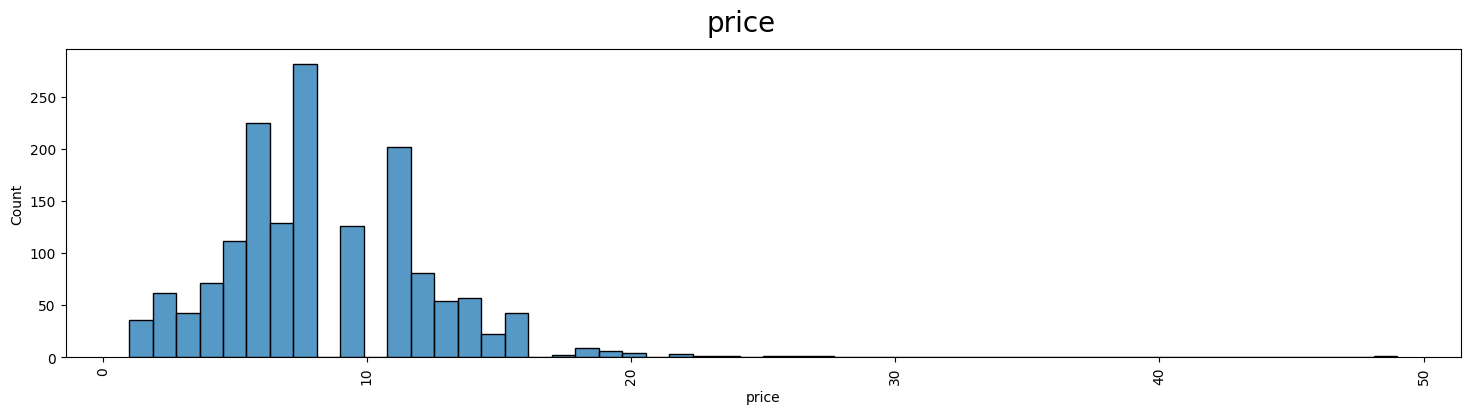
explainer = shap.TreeExplainer(rf\_model)

shap\_values = explainer.shap\_values(x)

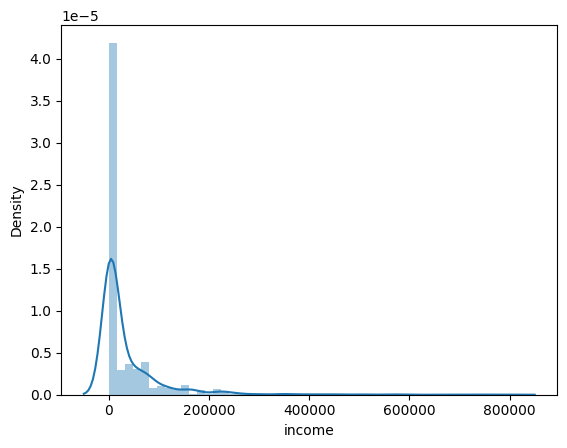
shap.summary\_plot(shap\_values[1], x)

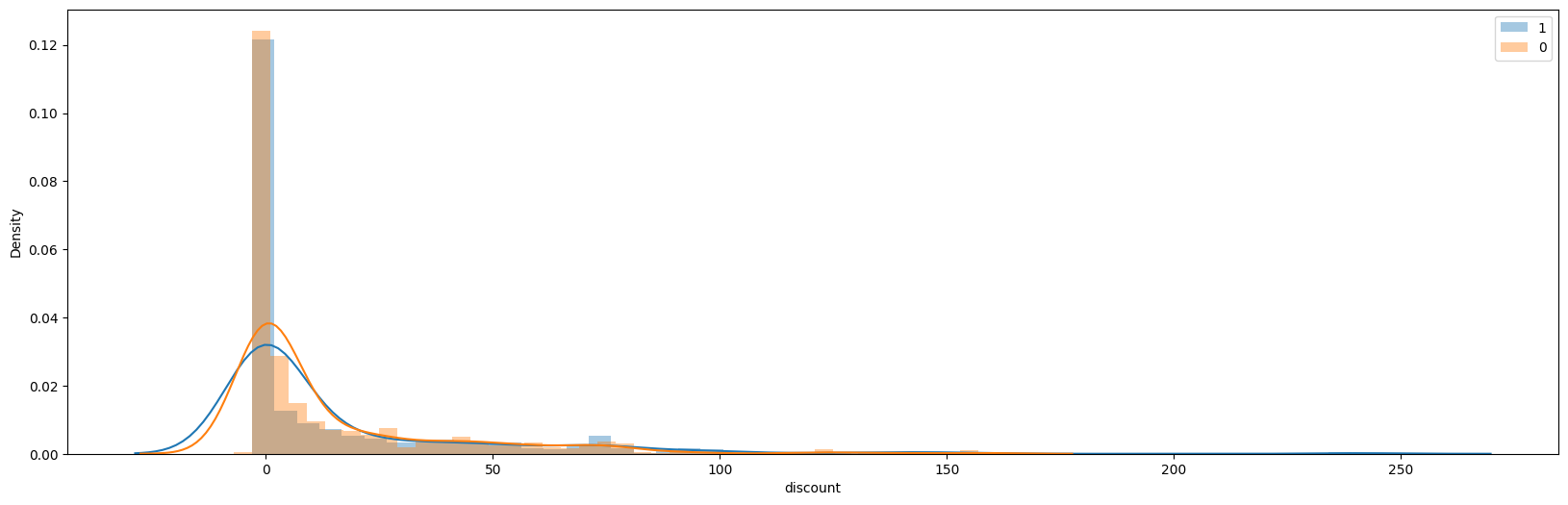
**Gráficos**

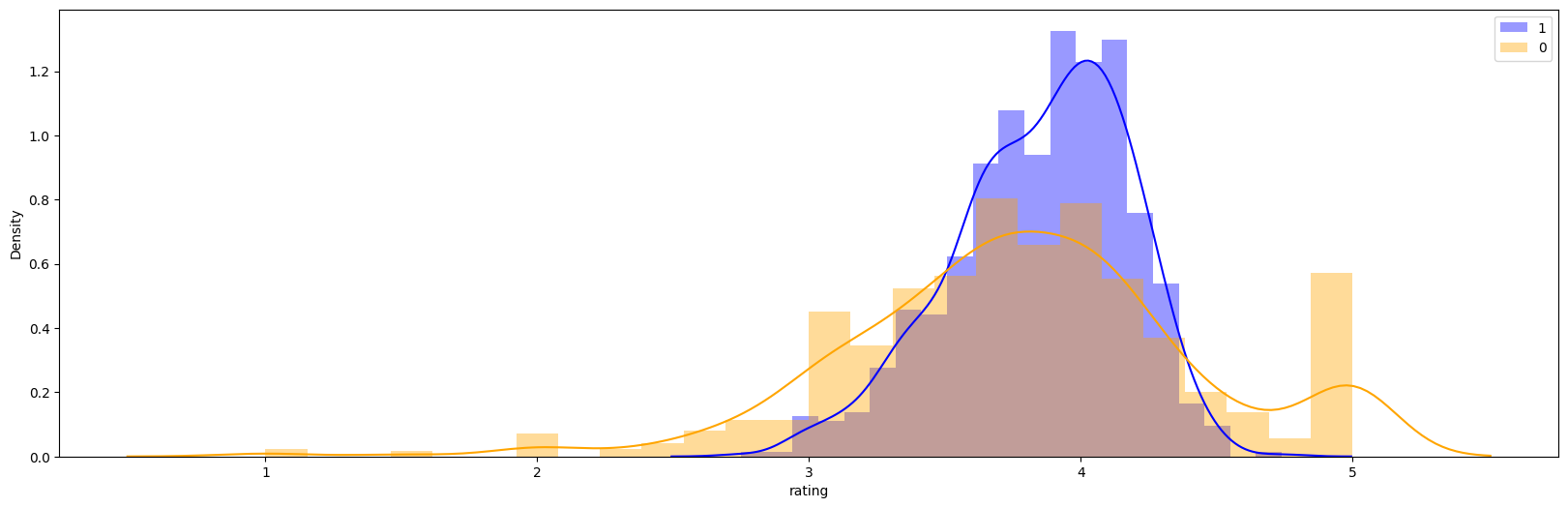
****

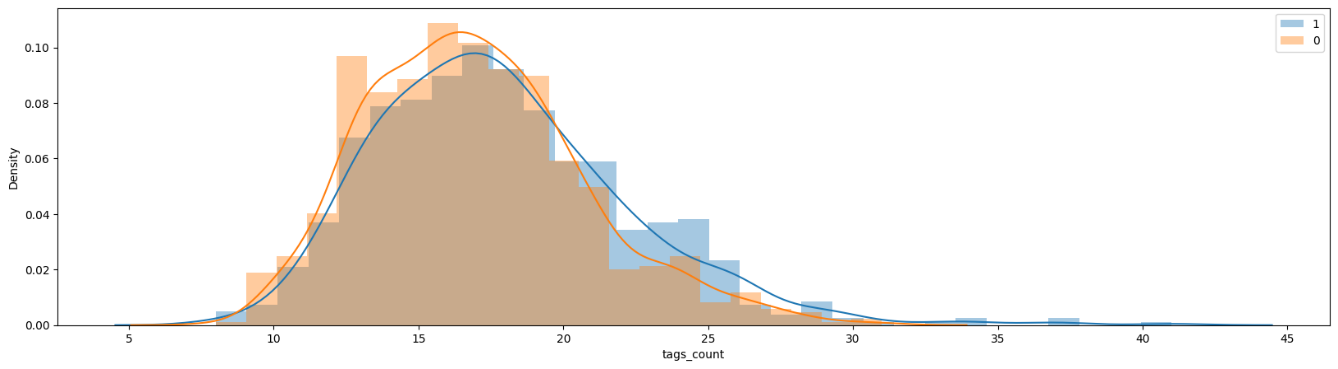
****

****

****

****

****

****

****

