

**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

<b>1. Introdução.....</b>	<b>4</b>
<b>1.1. Contextualização .....</b>	<b>4</b>
<b>1.2. O problema proposto.....</b>	<b>4</b>
<b>1.3. Objetivos .....</b>	<b>6</b>
<b>3. Processamento/Tratamento de Dados .....</b>	<b>7</b>
<b>4. Análise e Exploração dos Dados .....</b>	<b>10</b>
<b>5. Criação de Modelos de Machine Learning .....</b>	<b>11</b>
<b>6. Interpretação dos Resultados .....</b>	<b>18</b>
<b>7. Links.....</b>	<b>18</b>

## **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 atrativo 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.

```
[1] import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

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

```
#df_products.head()
df_products.tail()
```

	title	title_orig	price	retail_price	currency_buyer	units_sold	uses_ad_boosts	rating	rating_count	rating_five_count	...	merchant_rating_count	mercd
1568	Nouvelle Mode Femmes Bohème Pissenlit Imprimer...	New Fashion Women Bohemia Dandelion Print Tee ...	6.0	9	EUR	10000	1	4.08	1367	722.0	...	5316	

```
[i for i in df_products.columns]
```

```
['title',
'title_orig',
'price',
'retail_price',
'currency_buyer',
'units_sold',
'uses_ad_boosts',
'rating',
'rating_count',
'rating_five_count',
'rating_four_count',
'rating_three_count',
'rating_two_count',
'rating_one_count',
'badges_count',
'badge_local_product',
'badge_product_quality',
'badge_fast_shipping',
'tags',
'product_color',
'product_variation_size_id',
'product_variation_inventory']
```

#### 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).

```
[6] df_products = df_products[cols]

df_products.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1573 entries, 0 to 1572
Data columns (total 22 columns):
```

### 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.

```
df_products.isna().sum()

title      0
price      0
retail_price  0
currency_buyer  0
units_sold  0
uses_ad_boosts  0
rating     0
rating_count  0
badges_count  0
badge_product_quality  0
badge_fast_shipping  0
tags       0
product_color  41
product_variation_size_id  14
product_variation_inventory  0
shipping_is_express  0
countries_shipped_to  0
inventory_total  0
has_urgency_banner  1100
origin_country  17
merchant_rating_count  0
merchant_rating  0
dtype: int64
```

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.

```
[9] 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()
```

title	0
price	0
retail_price	0
currency_buyer	0
units_sold	0
uses_ad_boosts	0
rating	0
rating_count	0
badges_count	0
badge_product_quality	0
badge_fast_shipping	0
tags	0
product_color	0
product_variation_size_id	0
product_variation_inventory	0
shipping_is_express	0
countries_shipped_to	0
inventory_total	0
has_urgency_banner	0
origin_country	0
merchant_rating_count	0
merchant_rating	0
dtype: int64	

## 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.

```
[11] df_products.describe()
```

	price	retail_price	units_sold	uses_ad_boosts	rating	rating_count	badges_count	badge_product_quality	badge_fast_shipping	product_variation_inventory	shipping_is_expr
count	1573.000000	1573.000000	1573.000000	1573.000000	1573.000000	1573.000000	1573.000000	1573.000000	1573.000000	1573.000000	1573.000000
mean	8.325372	23.288620	4339.005086	0.432931	3.820896	889.659250	0.105531	0.074380	0.012715	33.081373	0.002
std	3.932030	30.357863	9356.539302	0.495639	0.515374	1983.928834	0.340709	0.262472	0.112075	21.353137	0.050
min	1.000000	1.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000
25%	5.810000	7.000000	100.000000	0.000000	3.550000	24.000000	0.000000	0.000000	0.000000	6.000000	0.000
50%	8.000000	10.000000	1000.000000	0.000000	3.850000	150.000000	0.000000	0.000000	0.000000	50.000000	0.000
75%	11.000000	26.000000	5000.000000	1.000000	4.110000	855.000000	0.000000	0.000000	0.000000	50.000000	0.000
max	49.000000	252.000000	100000.000000	1.000000	5.000000	20744.000000	3.000000	1.000000	1.000000	50.000000	1.000

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.

```
[14] categorical_cols

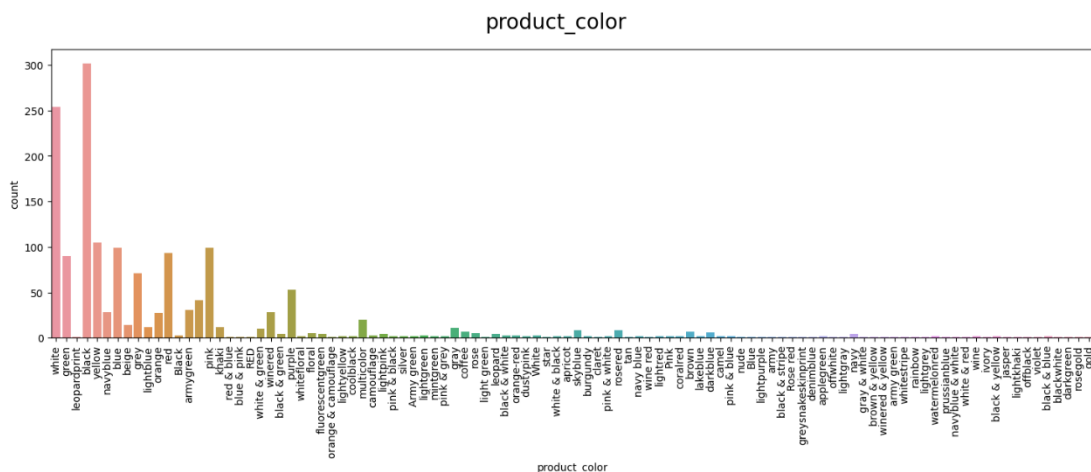
['title',
 'currency_buyer',
 'tags',
 'product_color',
 'product_variation_size_id',
 'origin_country']

[15] df_products["tags"] # Esta precisará de um tratamento

0      Summer,Fashion,womenunderwearsuit,printedpajam...
1      Mini,womens dresses,Summer,Patchwork,fashion d...
2      Summer,cardigan,women beachwear,chiffon,Sexy w...
3      Summer,Shorts,Cotton,Cotton T Shirt,Sleeve,pri...
4      Summer,Plus Size,Lace,Casual pants,Bottom,pant...
...
1568    bohenia,Plus Size,dandelionfloralprinted,short...
1569    Summer,Panties,Elastic,Lace,Casual pants,casua...
1570    runningshort,Beach Shorts,beachpant,menbeachsh...
1571    Summer,fashion women,Fashion,Lace,Dresses,Dres...
1572    Summer,Leggings,slim,Yoga,pants,Slim Fit,Women...
Name: tags, Length: 1573, dtype: object

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()
```

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.



```
[24] from wordcloud import WordCloud, STOPWORDS

[25] import matplotlib.pyplot as plt
word_string=" ".join(df_products['tags'].str.lower())
wordcloud = WordCloud(stopwords=STOPWORDS).generate(word_string)

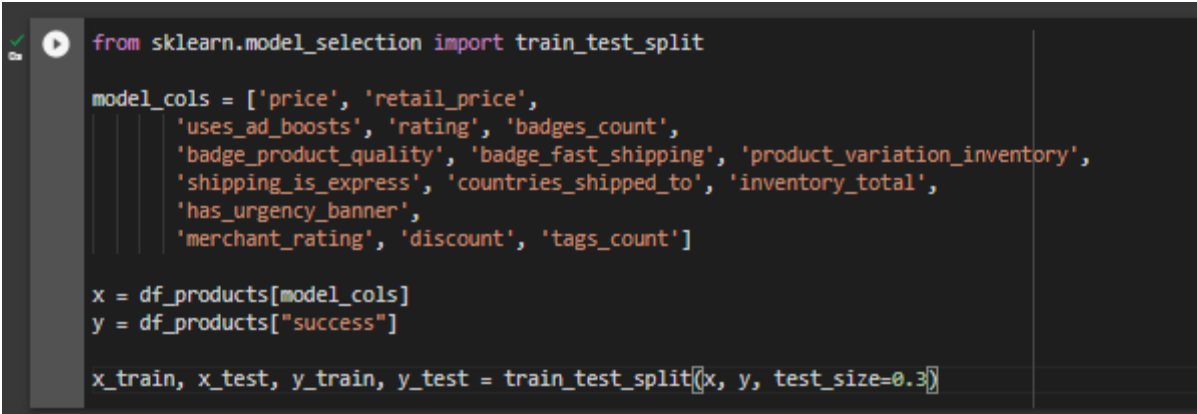
plt.subplots(figsize=(15,15))
plt.clf()
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```



## 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.



```
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"]

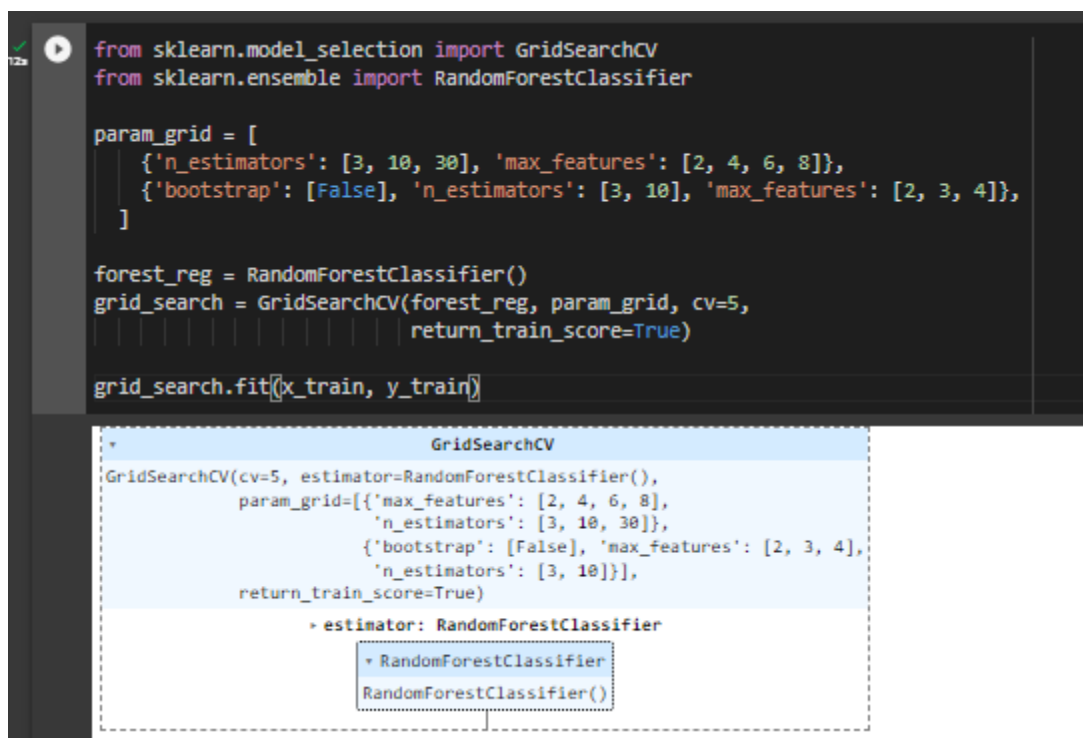
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)
```

Em seguida, 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.



```
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)
```

The screenshot also shows the interactive representation of the `GridSearchCV` object, which displays the following attributes:

- `GridSearchCV`
  - `GridSearchCV(cv=5, estimator=RandomForestClassifier(), param_grid=[{'max_features': [2, 4, 6, 8], 'n_estimators': [3, 10, 30]}, {'bootstrap': [False], 'max_features': [2, 3, 4], 'n_estimators': [3, 10]}], return_train_score=True)`
  - `estimator: RandomForestClassifier`
    - `RandomForestClassifier`
      - `RandomForestClassifier()`

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.

```
[50] grid_search.best_params_
      rf_model = grid_search.best_estimator_

from sklearn.metrics import classification_report, confusion_matrix

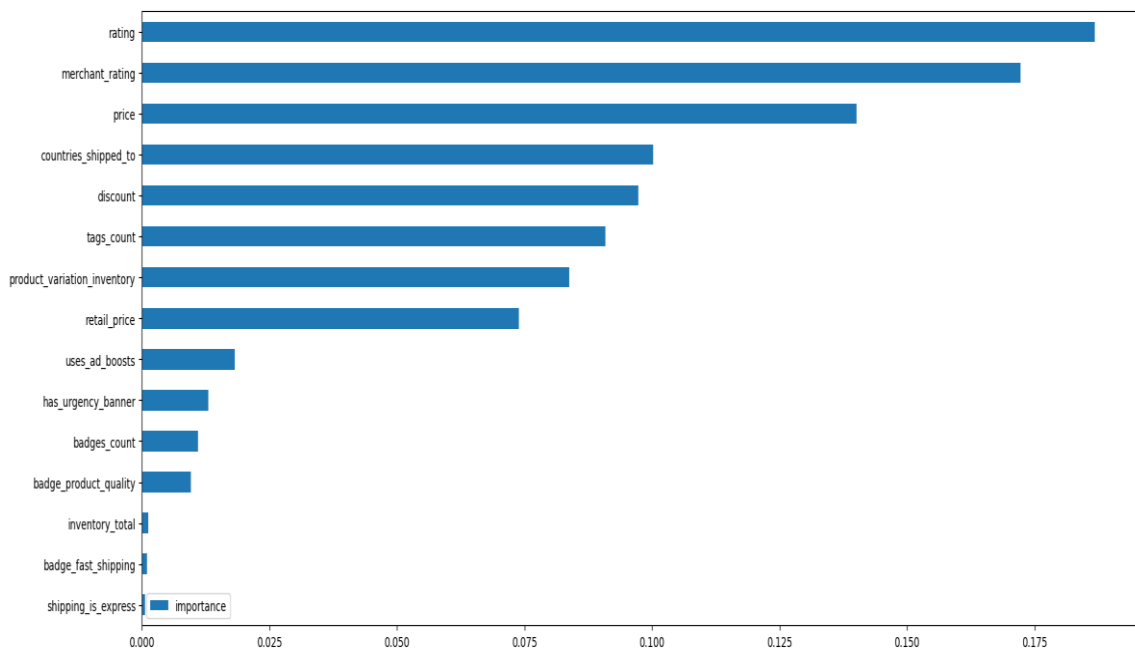
y_pred = rf_model.predict(x_test)
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.72	0.73	0.73	237
1	0.73	0.72	0.72	235
accuracy			0.73	472
macro avg	0.73	0.73	0.73	472
weighted avg	0.73	0.73	0.73	472

```
[[174  63]
 [ 66 169]]
```

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.

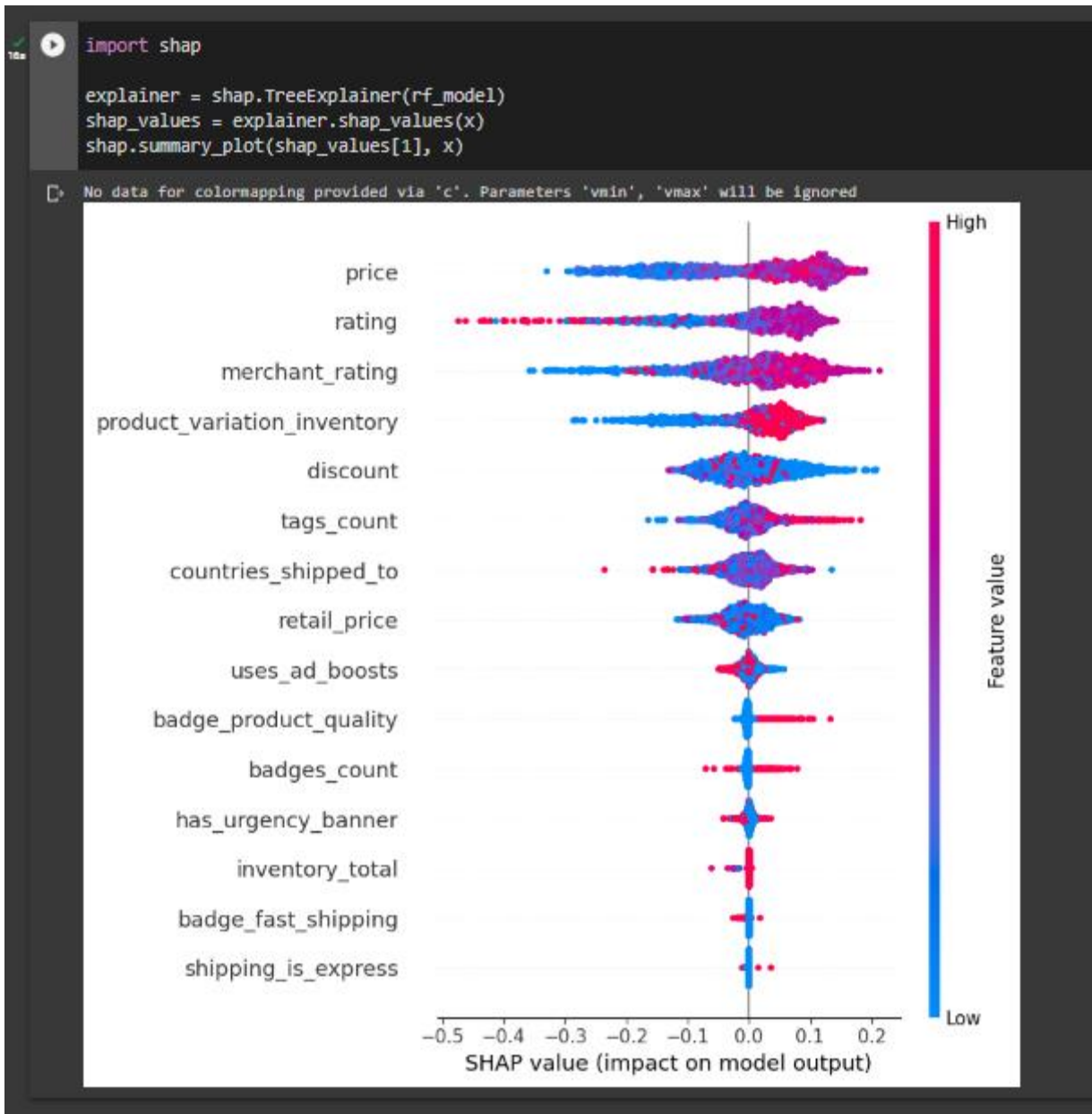


Com o gráfico, é possível compreender a importância de determinar variáveis como “rating”, “merchant\_rating”, “price” e “countries\_shipped\_to”. Contudo, para ter um parecer analítico, utilizou-se a biblioteca “shap” para fazer uma simplificação do modelo e entender o impacto, e assim, tomar uma decisão 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>

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/1XZiCewwfUcWlgg3TYaZRI\\_r-KxVyHDdU](https://drive.google.com/drive/u/0/folders/1XZiCewwfUcWlgg3TYaZRI_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["re-
tail_price"] - df_products["price"]

fig, ax = plt.subplots(figsize=(20, 6))
sns.distplot(df_products.loc[df_products["suc-
cess"] == 1, "discount"], label="1")
sns.distplot(df_products.loc[df_products["suc-
cess"] == 0, "discount"], label="0")
plt.legend()
```

```
df_products.head(1)
```

```
df_products.loc[df_products["suc-
cess"] == 0, "uses_ad_boosts"].value_counts()
```

```
df_products.loc[df_products["suc-
cess"] == 1, "uses_ad_boosts"].value_counts()
```

```
fig, ax = plt.subplots(figsize=(20, 6))
sns.distplot(df_products.loc[df_products["suc-
cess"] == 1, "rating"], label="1", color="blue")
sns.distplot(df_products.loc[df_products["suc-
cess"] == 0, "rating"], label="0", color="orange")
plt.legend()
```

```
df_products.groupby(["success", "badges_count"]).count()[["ti-
tle"]].pivot_table(index="success", col-
umns="badges_count").fillna(0)
```

```
df_products.groupby(["success", "badge_product_qual-
ity"]).count()[["title"]].pivot_table(index="success", col-
umns="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_ex-
press"]).count()[["title"]].pivot_table(index="success", col-
umns="shipping_is_express").fillna(0)
```

```
df_products["discount"] = df_products["re-
tail_price"] - df_products["price"]

fig, ax = plt.subplots(figsize=(20, 5))
sns.distplot(df_products.loc[df_products["suc-
cess"] == 1, "countries_shipped_to"], label="1")
sns.distplot(df_products.loc[df_products["suc-
cess"] == 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', 'prod-
uct_variation_inventory',
               'shipping_is_express', 'countries_shipped_to', 'inven-
tory_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_fea-
tures': [2, 4, 6, 8]},
    {'bootstrap': [False], 'n_estimators': [3, 10], 'max_fea-
tures': [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, confu-
sion_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_impor-
tances_,
                                   index = x.columns,
                                   columns=['im-
portance']).sort_values('importance', ascending=True)

fig, ax = plt.subplots(figsize=(20, 8))
feature_importances.plot(kind="barh", ax=ax)
```

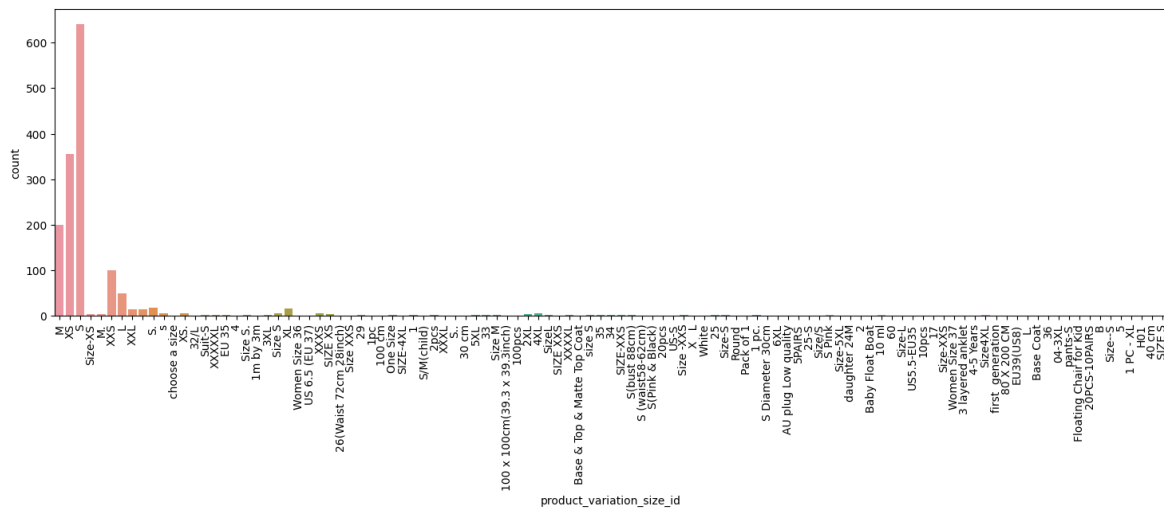
```
!pip install shap
```

```
import shap

explainer = shap.TreeExplainer(rf_model)
shap_values = explainer.shap_values(x)
shap.summary_plot(shap_values[1], x)
```

## Gráficos

product\_variation\_size\_id



price

