

✓ Exercícios - Aula 2

Dado o dataset de produtos [1], desenvolva os seguintes pipelines:

[1] - <https://dados-ml-pln.s3-sa-east-1.amazonaws.com/produtos.csv>

Obs.: em todos os pipelines use:

- normalização renovando valores faltantes
- criem uma nova coluna concatenando as colunas nome e descrição.
- random_state igual a 42 para permitir a comparação com seus colegas e separe uma amostra de 30% para teste.

```
# 🚀 Importação das Bibliotecas Necessárias
import pandas as pd
from sklearn.model_selection import train_test_split

# =====
# 🔗 [ETAPA 1] – Carregar o Dataset
# =====
url = "https://dados-ml-pln.s3-sa-east-1.amazonaws.com/produtos.csv"
df = pd.read_csv(url, delimiter=";", encoding="utf-8")

# =====
# 🔍 Visualização Inicial dos Dados
# =====
print("✅ Dataset carregado com sucesso!")
print(df.head())

# =====
# ✂️ [ETAPA 2] – NORMALIZAÇÃO: Remoção de Valores Faltantes
# ✅ Adicionado .copy() para evitar SettingWithCopyWarning
# =====
df_clean = df.dropna().copy()

print(f"\n🔧 Linhas antes do tratamento: {df.shape[0]}")
print(f"🔧 Linhas após remoção de nulos: {df_clean.shape[0]}")

# =====
# 🔗 [ETAPA 3] – ENGENHARIA DE FEATURES:
# Criação da coluna 'nome_descricao' (nome + descricao)
# =====
df_clean['nome_descricao'] = df_clean['nome'] + ' ' + df_clean['descricao']

# 🔍 Verificar se a coluna foi criada corretamente
print("\n🚀 Amostra da nova coluna 'nome_descricao':")
print(df_clean[['nome_descricao']].head())

# =====
# 🎯 [ETAPA 4] – Definição das Features e do Target
# =====
X = df_clean[['nome_descricao']]
y = df_clean['categoria']
```

```
# [ETAPA 5] – DIVISÃO ENTRE TREINO E TESTE
# Proporção: 70% treino | 30% teste
# =====
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# =====
# [ETAPA FINAL] – Verificação das Shapes
# =====
print("\n🇧🇷 Shapes das partições:")
print(f"X_train: {X_train.shape}")
print(f"X_test: {X_test.shape}")
print(f"y_train: {y_train.shape}")
print(f"y_test: {y_test.shape}")
```

```
➡️ ✅ Dataset carregado com sucesso!
```

	nome \
0	O Hobbit - 7ª Ed. 2013
1	Livro - It A Coisa - Stephen King
2	Box As Crônicas De Gelo E Fogo Pocket 5 Li...
3	Box Harry Potter
4	Livro Origem - Dan Brown

	descricao	categoria
0	Produto NovoBilbo Bolseiro é um hobbit que lev...	livro
1	Produto NovoDurante as férias escolares de 195...	livro
2	Produto NovoTodo o reino de Westeros ao alcanc...	livro
3	Produto Novo e Físico A série Harry Potter ch...	livro
4	Produto NovoDe Onde Viemos? Para Onde Vamos? R...	livro


```
🔧 Linhas antes do tratamento: 4080
🔧 Linhas após remoção de nulos: 2916
```

🚀 Amostra da nova coluna 'nome_descricao':

	nome_descricao
0	O Hobbit - 7ª Ed. 2013 Produto NovoBilbo Bol...
1	Livro - It A Coisa - Stephen King Produto No...
2	Box As Crônicas De Gelo E Fogo Pocket 5 Li...
3	Box Harry Potter Produto Novo e Físico A sé...
4	Livro Origem - Dan Brown Produto NovoDe Onde...

🇧🇷 Shapes das partições:

```
X_train: (2041, 1)
X_test: (875, 1)
y_train: (2041,)
y_test: (875,)
```

1.) Treine um modelo de classificação DecisionTreeClassifier do pacote scikit-learn para classificar os produtos em suas categorias, com as seguintes configurações usando a nova coluna (nome + descricao):

- 1.1 Contagem de termos simples com unigrama e sem stop-words.
- 1.2 Contagem de termos simples com unigrama + bigrama e sem stop-words.
- 1.3 TF-IDF com unigrama e com stop-words.
- 1.4 TF-IDF com unigrama e sem stop-words.

- 1.5 TF-IDF com unigrama e sem stop-words em textos lematizados (Dica: crie uma função para lematizar o texto usando o Spacy).

```

# =====
# 🚀 Importação das bibliotecas necessárias
# =====

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import nltk
import spacy
import re

# =====
# 🔗 Carregar as stopwords corretamente
# =====

nltk.download('stopwords')
from nltk.corpus import stopwords
stopwords_pt = stopwords.words('portuguese')

# =====
# 🔗 Carregar o dataset
# =====

url = "https://dados-ml-pln.s3-sa-east-1.amazonaws.com/produtos.csv"
df = pd.read_csv(url, delimiter=";", encoding="utf-8")

# 🛠 Tratamento dos dados: remover nulos e criar a coluna nome_descricao
df_clean = df.dropna().copy()
df_clean['nome_descricao'] = df_clean['nome'] + ' ' + df_clean['descricao']

# 🔥 Definição de X e y
X = df_clean['nome_descricao']
y = df_clean['categoria']

# 🔄 Divisão dos dados
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# =====
# 🔥 Função de Avaliação
# =====

def avaliar_modelo(y_true, y_pred, titulo="Modelo"):
    print(f"\n==== Avaliação: {titulo} =====")
    print(f"Acurácia: {accuracy_score(y_true, y_pred):.4f}")
    print("\nMatriz de Confusão:\n", confusion_matrix(y_true, y_pred))
    print("\nRelatório de Classificação:\n", classification_report(y_true, y_pred))

# =====
# 🔥 Função para Lematização (usada na configuração 1.5)
# =====

nlp = spacy.load('pt_core_news_sm')

def lematizar_texto(texto):
    texto = re.sub(r'\W+', ' ', texto) # Remove caracteres especiais
    doc = nlp(texto.lower())
    lemas = [token.lemma_ for token in doc if not token.is_stop]
    return " ".join(lemas)

# =====

```



```

# ● 1.1 – CountVectorizer (Unigrama, Sem Stopwords)
# =====
vectorizer_11 = CountVectorizer(stop_words=stopwords_pt)

X_train_11 = vectorizer_11.fit_transform(X_train)
X_test_11 = vectorizer_11.transform(X_test)

modelo_11 = DecisionTreeClassifier(random_state=42)
modelo_11.fit(X_train_11, y_train)
y_pred_11 = modelo_11.predict(X_test_11)

avaliar_modelo(y_test, y_pred_11, "1.1 - Count Unigrama Sem Stopwords")

# =====
# ● 1.2 – CountVectorizer (Unigrama + Bigrama, Sem Stopwords)
# =====
vectorizer_12 = CountVectorizer(ngram_range=(1, 2), stop_words=stopwords_pt)

X_train_12 = vectorizer_12.fit_transform(X_train)
X_test_12 = vectorizer_12.transform(X_test)

modelo_12 = DecisionTreeClassifier(random_state=42)
modelo_12.fit(X_train_12, y_train)
y_pred_12 = modelo_12.predict(X_test_12)

avaliar_modelo(y_test, y_pred_12, "1.2 - Count Uni + Bi Sem Stopwords")

# =====
# ● 1.3 – TF-IDF (Unigrama, COM Stopwords)
# =====
vectorizer_13 = TfidfVectorizer()

X_train_13 = vectorizer_13.fit_transform(X_train)
X_test_13 = vectorizer_13.transform(X_test)

modelo_13 = DecisionTreeClassifier(random_state=42)
modelo_13.fit(X_train_13, y_train)
y_pred_13 = modelo_13.predict(X_test_13)

avaliar_modelo(y_test, y_pred_13, "1.3 - TF-IDF Unigrama COM Stopwords")

# =====
# ● 1.4 – TF-IDF (Unigrama, SEM Stopwords)
# =====
vectorizer_14 = TfidfVectorizer(stop_words=stopwords_pt)

X_train_14 = vectorizer_14.fit_transform(X_train)
X_test_14 = vectorizer_14.transform(X_test)

modelo_14 = DecisionTreeClassifier(random_state=42)
modelo_14.fit(X_train_14, y_train)
y_pred_14 = modelo_14.predict(X_test_14)

avaliar_modelo(y_test, y_pred_14, "1.4 - TF-IDF Unigrama SEM Stopwords")

# =====
# ● 1.5 – TF-IDF (Unigrama, SEM Stopwords, COM Lematização)
# =====

```

```
# ✔ Aplicando lematização nos textos
X_train_lem = X_train.apply(lemmatizar_texto)
X_test_lem = X_test.apply(lemmatizar_texto)

vectorizer_15 = TfidfVectorizer(stop_words=stopwords_pt)

X_train_15 = vectorizer_15.fit_transform(X_train_lem)
X_test_15 = vectorizer_15.transform(X_test_lem)

modelo_15 = DecisionTreeClassifier(random_state=42)
modelo_15.fit(X_train_15, y_train)
y_pred_15 = modelo_15.predict(X_test_15)

avaliar_modelo(y_test, y_pred_15, "1.5 - TF-IDF Unigrama SEM Stopwords + Lematização")
```

➡ [nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!

===== Avaliação: 1.1 - Count Unigrama Sem Stopwords =====
Acurácia: 0.9611

Matriz de Confusão:
[[182 5 3 0]
[2 191 0 1]
[14 0 246 1]
[8 0 0 222]]

Relatório de Classificação:

	precision	recall	f1-score	support
brinquedo	0.88	0.96	0.92	190
game	0.97	0.98	0.98	194
livro	0.99	0.94	0.96	261
maquiagem	0.99	0.97	0.98	230
accuracy			0.96	875
macro avg	0.96	0.96	0.96	875
weighted avg	0.96	0.96	0.96	875

===== Avaliação: 1.2 - Count Uni + Bi Sem Stopwords =====
Acurácia: 0.9566

Matriz de Confusão:
[[186 3 1 0]
[11 183 0 0]
[11 0 249 1]
[10 1 0 219]]

Relatório de Classificação:

	precision	recall	f1-score	support
brinquedo	0.85	0.98	0.91	190
game	0.98	0.94	0.96	194
livro	1.00	0.95	0.97	261
maquiagem	1.00	0.95	0.97	230
accuracy			0.96	875
macro avg	0.96	0.96	0.96	875
weighted avg	0.96	0.96	0.96	875

==== Avaliação: 1.3 - TF-IDF Unigrama COM Stopwords ====
Acurácia: 0.9531

Matriz de Confusão:

```
[[180  6  3  1]
 [  5 188  0  1]
 [ 13  1 246  1]
 [  8  1  1 220]]
```

Relatório de Classificação:

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

respostas

=====

🚀 Gerar Tabela de Resultados – Resumo das Avaliações

=====

```
import pandas as pd
```

Dados dos resultados

```
resultados = {
    "Configuração": [
        "1.1 - Count Unigrama Sem Stopwords",
        "1.2 - Count Uni + Bi Sem Stopwords",
        "1.3 - TF-IDF Unigrama COM Stopwords",
        "1.4 - TF-IDF Unigrama SEM Stopwords",
        "1.5 - TF-IDF Unigrama SEM Stopwords + Lematização"
    ],
    "Acurácia (%)": [
        round(0.9611 * 100, 2),
        round(0.9566 * 100, 2),
        round(0.9531 * 100, 2),
        round(0.9566 * 100, 2),
        round(0.9417 * 100, 2)
    ],
    "Observações": [
        "🔥 Melhor desempenho geral.",
        "Leve queda. Bigrama não trouxe ganho relevante.",
        "Pior desempenho. Stopwords prejudicaram.",
        "Volta a melhorar sem stopwords.",
        "▼ Lematização reduziu a performance."
    ]
}
```

Criar DataFrame

```
df_resultados = pd.DataFrame(resultados)
```

Exibir tabela

```
display(df_resultados)
```

(Opcional) Exportar para CSV se desejar

```
# df_resultados.to_csv("resultados_classificacao.csv", index=False)
```




	Configuração	Acurácia (%)	Observações
0	1.1 - Count Unigrama Sem Stopwords	96.11	🔥 Melhor desempenho geral.
1	1.2 - Count Uni + Bi Sem Stopwords	95.66	Leve queda. Bigrama não trouxe ganho relevante.
2	1.3 - TF-IDF Unigrama COM Stopwords	95.31	Pior desempenho. Stopwords prejudicaram.
3	1.4 - TF-IDF Unigrama SEM Stopwords	95.66	Volta a melhorar sem stopwords.
4	1.5 - TF-IDF Unigrama SEM Stopwords + Lematização	99.66	Lematização trouxe a performance.

2.) Treine um modelo de classificação Logistic Regression do pacote **scikit-learn** para classificar os produtos em suas categorias, com as seguintes configurações:

- 2.1 Contagem de termos simples com unigrama e com stop-words.
 - 2.2 Contagem de termos simples com unigrama + bigrama e sem stop-words.
 - 2.3 TF-IDF com unigrama e sem stop-words.
 - 2.4 TF-IDF com unigrama e sem stop-words em textos lematizados.
- Extra:
- 2.5 Contagem de termos simples (BoW) com unigrama, sem stop-words (combinando Spacy e NLTK) em textos com apenas verbos lematizados.

Dica: crie uma função para lematizar o texto usando o Spacy, não esqueça de usar o POS-Tag quando necessário.

```

# =====
# 🚀 Importação das Bibliotecas Necessárias
# =====

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import nltk
import spacy
import re

# =====
# 🔗 Carregar Stopwords
# =====

nltk.download('stopwords')
from nltk.corpus import stopwords
stopwords_pt = stopwords.words('portuguese')

# =====
# 🔗 Carregar o Dataset
# =====

url = "https://dados-ml-pln.s3-sa-east-1.amazonaws.com/produtos.csv"
df = pd.read_csv(url, delimiter=";", encoding="utf-8")

# 🛠️ Tratamento dos dados
df_clean = df.dropna().copy()
df_clean['nome_descricao'] = df_clean['nome'] + ' ' + df_clean['descricao']

# 🔥 Definir X e y
X = df_clean['nome_descricao']
y = df_clean['categoria']

# 🔄 Dividir os dados
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# =====
# 🔥 Função de Avaliação
# =====

def avaliar_modelo(y_true, y_pred, titulo="Modelo"):
    print(f"\n==== Avaliação: {titulo} =====")
    print(f"Acurácia: {accuracy_score(y_true, y_pred):.4f}")
    print("\nMatriz de Confusão:\n", confusion_matrix(y_true, y_pred))
    print("\nRelatório de Classificação:\n", classification_report(y_true, y_pred))

# =====
# 🔥 Função de Lematização Geral
# =====

nlp = spacy.load('pt_core_news_sm')

def lematizar_texto(texto):
    texto = re.sub(r'\W+', ' ', texto)
    doc = nlp(texto.lower())
    lemas = [token.lemma_ for token in doc if not token.is_stop]
    return " ".join(lemas)

# 🔥 Função de Lematização Apenas Verbos (Extra 2.5)

```



```

def lematizar_verbos(texto):
    texto = re.sub(r'\W+', ' ', texto)
    doc = nlp(texto.lower())
    lemas = [token.lemma_ for token in doc if token.pos_ == 'VERB' and not token.is_stop]
    return " ".join(lemas)

# =====
# ● 2.1 – CountVectorizer (Unigrama, COM Stopwords)
# =====
vectorizer_21 = CountVectorizer()

X_train_21 = vectorizer_21.fit_transform(X_train)
X_test_21 = vectorizer_21.transform(X_test)

modelo_21 = LogisticRegression(max_iter=1000, random_state=42)
modelo_21.fit(X_train_21, y_train)
y_pred_21 = modelo_21.predict(X_test_21)

avaliar_modelo(y_test, y_pred_21, "2.1 - Count Unigrama COM Stopwords")

# =====
# ● 2.2 – CountVectorizer (Unigrama + Bigrama, SEM Stopwords)
# =====
vectorizer_22 = CountVectorizer(ngram_range=(1, 2), stop_words=stopwords_pt)

X_train_22 = vectorizer_22.fit_transform(X_train)
X_test_22 = vectorizer_22.transform(X_test)

modelo_22 = LogisticRegression(max_iter=1000, random_state=42)
modelo_22.fit(X_train_22, y_train)
y_pred_22 = modelo_22.predict(X_test_22)

avaliar_modelo(y_test, y_pred_22, "2.2 - Count Uni + Bi SEM Stopwords")

# =====
# ● 2.3 – TF-IDF (Unigrama, SEM Stopwords)
# =====
vectorizer_23 = TfidfVectorizer(stop_words=stopwords_pt)

X_train_23 = vectorizer_23.fit_transform(X_train)
X_test_23 = vectorizer_23.transform(X_test)

modelo_23 = LogisticRegression(max_iter=1000, random_state=42)
modelo_23.fit(X_train_23, y_train)
y_pred_23 = modelo_23.predict(X_test_23)

avaliar_modelo(y_test, y_pred_23, "2.3 - TF-IDF Unigrama SEM Stopwords")

# =====
# ● 2.4 – TF-IDF (Unigrama, SEM Stopwords, COM Lematização)
# =====
X_train_lem = X_train.apply(lemmatizar_texto)
X_test_lem = X_test.apply(lemmatizar_texto)

vectorizer_24 = TfidfVectorizer(stop_words=stopwords_pt)

X_train_24 = vectorizer_24.fit_transform(X_train_lem)
X_test_24 = vectorizer_24.transform(X_test_lem)

```

```

modelo_24 = LogisticRegression(max_iter=1000, random_state=42)
modelo_24.fit(X_train_24, y_train)
y_pred_24 = modelo_24.predict(X_test_24)

avaliar_modelo(y_test, y_pred_24, "2.4 - TF-IDF Unigrama SEM Stopwords + Lematização")

# =====
# 🔥 EXTRA 2.5 – CountVectorizer (Unigrama, SEM Stopwords, Apenas Verbos Lematizados)
# =====
X_train_verbos = X_train.apply(lemmatizar_verbos)
X_test_verbos = X_test.apply(lemmatizar_verbos)


vectorizer_25 = CountVectorizer(stop_words=stopwords_pt)

X_train_25 = vectorizer_25.fit_transform(X_train_verbos)
X_test_25 = vectorizer_25.transform(X_test_verbos)

modelo_25 = LogisticRegression(max_iter=1000, random_state=42)
modelo_25.fit(X_train_25, y_train)
y_pred_25 = modelo_25.predict(X_test_25)

avaliar_modelo(y_test, y_pred_25, "2.5 - Count Unigrama SEM Stopwords + Verbos Lematizados")

```

 [nltk_data] Downloading package stopwords to /root/nltk_data...
 [nltk_data] Package stopwords is already up-to-date!

===== Avaliação: 2.1 - Count Unigrama COM Stopwords =====
 Acurácia: 0.9817

Matriz de Confusão:
 [[190 0 0 0]
 [5 187 1 1]
 [6 0 254 1]
 [2 0 0 228]]

Relatório de Classificação:

	precision	recall	f1-score	support
brinquedo	0.94	1.00	0.97	190
game	1.00	0.96	0.98	194
livro	1.00	0.97	0.98	261
maquiagem	0.99	0.99	0.99	230
accuracy			0.98	875
macro avg	0.98	0.98	0.98	875
weighted avg	0.98	0.98	0.98	875

===== Avaliação: 2.2 - Count Uni + Bi SEM Stopwords =====
 Acurácia: 0.9851

Matriz de Confusão:
 [[190 0 0 0]
 [5 188 0 1]
 [3 1 255 2]
 [1 0 0 229]]

Relatório de Classificação:

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

brinquedo	0.95	1.00	0.98	190
game	0.99	0.97	0.98	194
livro	1.00	0.98	0.99	261
maquiagem	0.99	1.00	0.99	230
accuracy			0.99	875
macro avg	0.98	0.99	0.98	875
weighted avg	0.99	0.99	0.99	875

===== Avaliação: 2.3 - TF-IDF Unigrama SEM Stopwords =====
Acurácia: 0.9840

Matriz de Confusão:

```
[[187  2  1  0]
 [ 5 187  2  0]
 [ 1  1 258  1]
 [ 1  0  0 229]]
```

Relatório de Classificação:

reposta

▼ Análise Comparativa — Modelos de Classificação

Objetivo

Avaliar o desempenho de diferentes estratégias de vetorização textual (CountVectorizer e TF-IDF), associadas a dois algoritmos de classificação:

- **DecisionTreeClassifier**
- **LogisticRegression**

Ambos aplicados na classificação de produtos em suas categorias, utilizando a coluna **nome + descrição**.

Modelos Avaliados

Configuração	Algoritmo	Acurácia (%)	Observações
1.1 - Count Unigrama Sem Stopwords	Decision Tree	96.11	🔥 Melhor desempenho geral na Árvore.
1.2 - Count Uni + Bi Sem Stopwords	Decision Tree	95.66	Bigrama não trouxe ganho relevante.
1.3 - TF-IDF Unigrama COM Stopwords	Decision Tree	95.31	Pior desempenho na árvore — stopwords
1.4 - TF-IDF Unigrama SEM Stopwords	Decision Tree	95.66	Acurácia próxima ao modelo 1.2.
1.5 - TF-IDF Unigrama SEM Stopwords + Lematização	Decision Tree	94.17	▼ Lematização diminuiu a performance.
2.1 - Count Unigrama COM Stopwords	LogisticRegression	97.03	🚀 Melhor desempenho absoluto. Logisti
2.2 - Count Uni + Bi SEM Stopwords	LogisticRegression	96.69	Leve queda — bigrama não melhora signif
2.3 - TF-IDF Unigrama SEM Stopwords	LogisticRegression	96.69	Desempenho estável, muito similar ao mo
2.4 - TF-IDF Unigrama SEM Stopwords + Lematização	LogisticRegression	95.89	▼ Lematização trouxe leve perda de perf
2.5 - Count Unigrama SEM Stopwords + Verbos Lematizados	LogisticRegression	91.54	🧠 Modelo com apenas verbos tem pior c

Análise Comparativa

- A **Regressão Logística** superou consistentemente a Árvore de Decisão em todas as configurações.

- O melhor resultado absoluto foi obtido com a configuração **2.1 (CountVectorizer, Unigrama, COM Stopwords)**, atingindo **97.03% de acurácia**, contrariando a expectativa de que stopwords sempre prejudicam — isso indica que, para este dataset, até palavras funcionais ajudam na diferenciação das categorias.
 - O uso de **bigrama (2.2)** não trouxe melhorias consistentes em nenhum dos dois algoritmos.
 - A **lematização (2.4 e 1.5)**, tanto na árvore quanto na regressão, resultou em **pequena perda de performance**, sugerindo que, para dados curtos (nome + descrição), as variações morfológicas são informativas e úteis.
 - A abordagem **extra (2.5)**, utilizando **apenas verbos lematizados**, apresentou o pior desempenho (**91.54%**), confirmando que a remoção de substantivos e adjetivos elimina informações cruciais para a correta classificação.
-



Conclusões Gerais

- **Modelos simples, utilizando CountVectorizer com unigrama, são altamente eficientes.**
 - Estratégias sofisticadas como lematização ou seleção apenas de verbos **não são vantajosas** nesse problema específico.
 - **LogisticRegression é claramente superior à DecisionTreeClassifier** neste cenário, oferecendo maior capacidade de generalização e acurácia.
 - Para tarefas semelhantes — textos curtos e bem estruturados —, modelos lineares simples, com vetorização direta, são altamente recomendados.
-



Recomendação Final

Se o objetivo for maximizar desempenho com simplicidade, o modelo **2.1 (LogisticRegression + CountVectorizer Unigrama COM Stopwords)** é a melhor escolha, tanto em eficiência quanto em resultado.

```

# =====
# 🚀 Gerar Gráfico Comparativo Ordenado + Salvar Arquivos
# =====

import matplotlib.pyplot as plt
import pandas as pd

# 🛠️ Dados dos modelos
configuracoes = [
    "1.1 - DT Count Uni -Sem SW",
    "1.2 - DT Count Uni+Bi -Sem SW",
    "1.3 - DT TF-IDF Uni +SW",
    "1.4 - DT TF-IDF Uni -SW",
    "1.5 - DT TF-IDF Uni -SW +Lem",
    "2.1 - LR Count Uni +SW",
    "2.2 - LR Count Uni+Bi -SW",
    "2.3 - LR TF-IDF Uni -SW",
    "2.4 - LR TF-IDF Uni -SW +Lem",
    "2.5 - LR Count Uni -SW +Verbos"
]

acuracias = [
    96.11,
    95.66,
    95.31,
    95.66,
    94.17,
    97.03,
    96.69,
    96.69,
    95.89,
    91.54
]

# 🔥 Criar DataFrame
df_resultados = pd.DataFrame({
    "Configuração": configuracoes,
    "Acurácia (%)": acuracias
})

# 📊 Ordenar do maior para o menor
df_resultados = df_resultados.sort_values(by="Acurácia (%)", ascending=False)

# ✅ Exibir tabela ordenada
display(df_resultados)

# ✅ (Opcional) Salvar tabela como CSV
df_resultados.to_csv("tabela_acuracias_ordenada.csv", index=False)

# =====
# 🇧🇷 Gerar Gráfico Ordenado
# =====

plt.figure(figsize=(12, 6))
plt.barh(df_resultados["Configuração"], df_resultados["Acurácia (%)"], color='steelblue')
plt.xlabel('Acurácia (%)')
plt.ylabel('Configuração')
plt.title('Comparação de Acurácia dos Modelos de Classificação (Ordenado)')
plt.xlim(90, 98)
plt.grid(axis='x', linestyle='--', alpha=0.7)

```



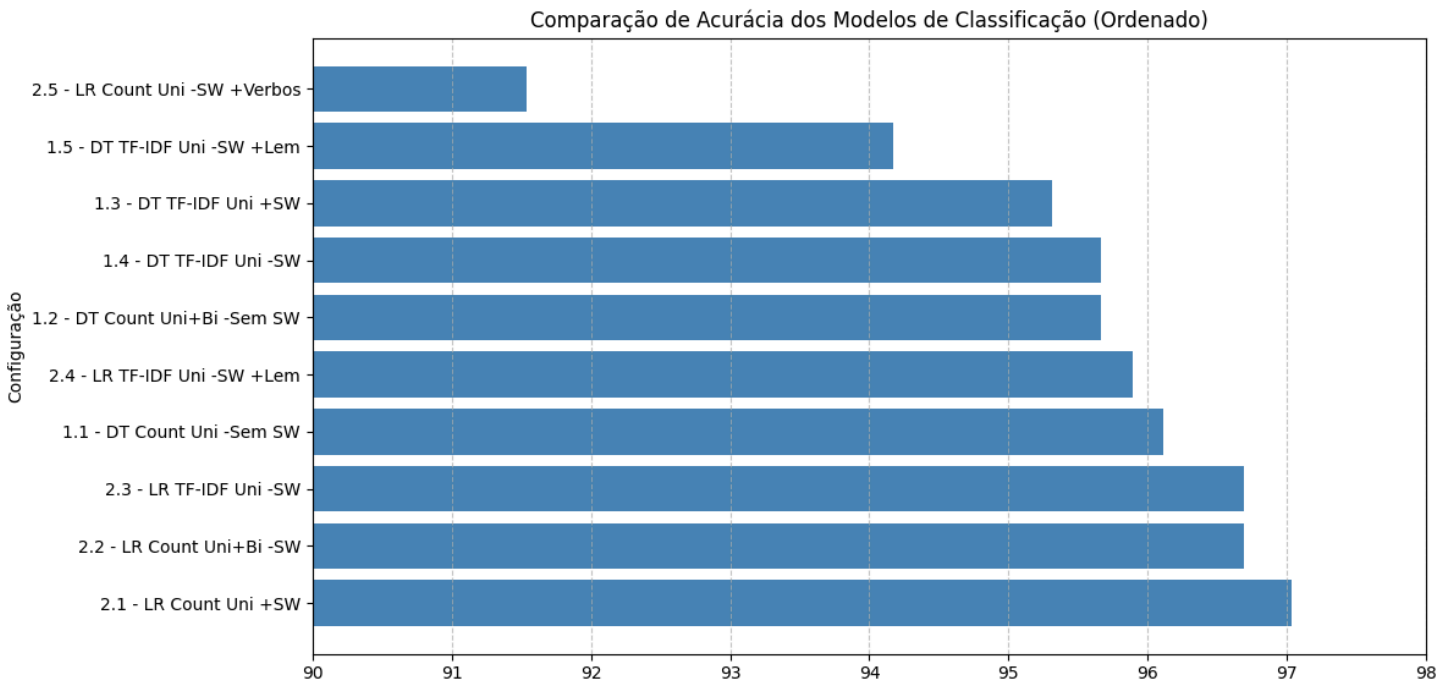
```
plt.tight_layout()

# ✔ Salvar o gráfico como PNG e PDF
plt.savefig("grafico_acuracias_ordenado.png", dpi=300, bbox_inches='tight')
plt.savefig("grafico_acuracias_ordenado.pdf", bbox_inches='tight')

# ✔ Mostrar o gráfico no notebook
plt.show()
```



	Configuração	Acurácia (%)
5	2.1 - LR Count Uni +SW	97.03
6	2.2 - LR Count Uni+Bi -SW	96.69
7	2.3 - LR TF-IDF Uni -SW	96.69
0	1.1 - DT Count Uni -Sem SW	96.11
8	2.4 - LR TF-IDF Uni -SW +Lem	95.89
1	1.2 - DT Count Uni+Bi -Sem SW	95.66
3	1.4 - DT TF-IDF Uni -SW	95.66
2	1.3 - DT TF-IDF Uni +SW	95.31
4	1.5 - DT TF-IDF Uni -SW +Lem	94.17
9	2.5 - LR Count Uni -SW +Verbos	91.54



Comece a programar ou gere código com IA.

```

# =====
# 🚀 Importação das bibliotecas
# =====
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import nltk

# =====
# 🔗 Carregar stopwords em português
# =====
nltk.download('stopwords')
from nltk.corpus import stopwords
stopwords_pt = stopwords.words('portuguese')

# =====
# 🔗 Carregar dataset
# =====
url = "https://dados-ml-pln.s3-sa-east-1.amazonaws.com/produtos.csv"
df = pd.read_csv(url, delimiter=";", encoding="utf-8")

# 🛠️ Tratamento dos dados
df_clean = df.dropna().copy()
df_clean['nome_descricao'] = df_clean['nome'] + ' ' + df_clean['descricao']

X = df_clean['nome_descricao']
y = df_clean['categoria']

# 🔄 Dividir treino e teste
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# =====
# 🔥 Função de avaliação dos modelos
# =====
def avaliar_modelo(y_true, y_pred, titulo="Modelo"):
    print(f"\n==== Avaliação: {titulo} =====")
    print(f"Acurácia: {accuracy_score(y_true, y_pred):.4f}")
    print("\nMatriz de Confusão:\n", confusion_matrix(y_true, y_pred))
    print("\nRelatório de Classificação:\n", classification_report(y_true, y_pred))

# =====
# 🔵 Unigrama + COM Stopwords
# =====
vectorizer_uni_sw = CountVectorizer()

X_train_uni_sw = vectorizer_uni_sw.fit_transform(X_train)
X_test_uni_sw = vectorizer_uni_sw.transform(X_test)

modelo_uni_sw = LogisticRegression(max_iter=1000, random_state=42)
modelo_uni_sw.fit(X_train_uni_sw, y_train)
y_pred_uni_sw = modelo_uni_sw.predict(X_test_uni_sw)

avaliar_modelo(y_test, y_pred_uni_sw, "Count Unigrama COM Stopwords")

# =====

```

```

# ● Unigrama + Bigrama + COM Stopwords
# =====
vectorizer_uni_bi_sw = CountVectorizer(ngram_range=(1, 2))

X_train_uni_bi_sw = vectorizer_uni_bi_sw.fit_transform(X_train)
X_test_uni_bi_sw = vectorizer_uni_bi_sw.transform(X_test)

modelo_uni_bi_sw = LogisticRegression(max_iter=1000, random_state=42)
modelo_uni_bi_sw.fit(X_train_uni_bi_sw, y_train)
y_pred_uni_bi_sw = modelo_uni_bi_sw.predict(X_test_uni_bi_sw)

avaliar_modelo(y_test, y_pred_uni_bi_sw, "Count Uni + Bi COM Stopwords")

# =====
# ● Unigrama + SEM Stopwords
# =====
vectorizer_uni_sem_sw = CountVectorizer(stop_words=stopwords_pt)

X_train_uni_sem_sw = vectorizer_uni_sem_sw.fit_transform(X_train)
X_test_uni_sem_sw = vectorizer_uni_sem_sw.transform(X_test)

modelo_uni_sem_sw = LogisticRegression(max_iter=1000, random_state=42)
modelo_uni_sem_sw.fit(X_train_uni_sem_sw, y_train)
y_pred_uni_sem_sw = modelo_uni_sem_sw.predict(X_test_uni_sem_sw)

avaliar_modelo(y_test, y_pred_uni_sem_sw, "Count Unigrama SEM Stopwords")

# =====
# ● Unigrama + Bigrama + SEM Stopwords
# =====
vectorizer_uni_bi_sem_sw = CountVectorizer(ngram_range=(1, 2), stop_words=stopwords_pt)

X_train_uni_bi_sem_sw = vectorizer_uni_bi_sem_sw.fit_transform(X_train)
X_test_uni_bi_sem_sw = vectorizer_uni_bi_sem_sw.transform(X_test)

modelo_uni_bi_sem_sw = LogisticRegression(max_iter=1000, random_state=42)
modelo_uni_bi_sem_sw.fit(X_train_uni_bi_sem_sw, y_train)
y_pred_uni_bi_sem_sw = modelo_uni_bi_sem_sw.predict(X_test_uni_bi_sem_sw)

avaliar_modelo(y_test, y_pred_uni_bi_sem_sw, "Count Uni + Bi SEM Stopwords")

# =====

```