

MaltMetrics – Métricas e Análises de Cervejas

Desenvolvido por: Leidiane Andrade Curso: Ciência de Dados – UFMS Tema: Aprendizado de Máquina Supervisionado e Não Supervisionado

Introdução

Neste projeto, utilizei um conjunto de dados com milhares de avaliações de cervejas do mundo todo. Cada avaliação traz informações como o aroma, o sabor, a aparência, o teor alcoólico e a nota geral atribuída pelos consumidores.

O objetivo foi ensinar o computador a reconhecer padrões nesses dados, ou seja, descobrir o que faz uma cerveja receber boas avaliações. Para isso, usei técnicas de aprendizado de máquina, em que o computador “aprende” com exemplos em vez de ser programado passo a passo.

O projeto foi dividido em etapas:

Análise Exploratória: entender os dados, visualizar distribuições e identificar padrões.

Pré-processamento: limpar, padronizar e preparar os dados para o treinamento.

Modelagem Supervisionada: aplicar modelos como Regressão Logística, Random Forest e XGBoost para prever se uma cerveja seria bem avaliada.

Modelagem Não Supervisionada: usar o algoritmo K-Means para agrupar cervejas semelhantes com base em suas características.

Avaliação e Conclusões: comparar o desempenho dos modelos e interpretar os resultados obtidos.

✓ Instalação de dependências

```
!pip install -q pandas numpy matplotlib seaborn scikit-learn xgboost
```

✓ Importação das bibliotecas

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix, classification_report
from sklearn.cluster import KMeans
import xgboost as xgb
import warnings
warnings.filterwarnings("ignore")
```

✓ Carregamento dos dados

```
from google.colab import files

print("📁 Faça o upload do arquivo beer_reviews.csv do seu computador:")
uploaded = files.upload()
```


📁 Faça o upload do arquivo beer_reviews.csv do seu computador:

Escolher Ficheiros beer_reviews.csv

beer_reviews.csv(text/csv) - 180174429 bytes, last modified: 21/10/2025 - 100% done

Saving beer_reviews.csv to beer_reviews (1).csv

```
# Carregar o arquivo enviado
df = pd.read_csv("beer_reviews.csv")
print("\n✅ Dataset carregado com sucesso!")
print("Dimensões:", df.shape)
df.head()
```

 Dataset carregado com sucesso!

Dimensões: (1586614, 13)

| | brewery_id | brewery_name | review_time | review_overall | review_aroma | review_appearance | review_profile | beer_style | review |
|---|------------|-------------------------|-------------|----------------|--------------|-------------------|----------------|--------------------------------|--------|
| 0 | 10325 | Vecchio Birraio | 1234817823 | 1.5 | 2.0 | 2.5 | stcules | Hefeweizen | |
| 1 | 10325 | Vecchio Birraio | 1235915097 | 3.0 | 2.5 | 3.0 | stcules | English Strong Ale | |
| 2 | 10325 | Vecchio Birraio | 1235916604 | 3.0 | 2.5 | 3.0 | stcules | Foreign / Export Stout | |
| 3 | 10325 | Vecchio Birraio | 1234725145 | 3.0 | 3.0 | 3.5 | stcules | German Pilsener | |
| 4 | 1075 | Caldera Brewing Company | 1293735206 | 4.0 | 4.5 | 4.0 | johnmichaelsen | American Double / Imperial IPA | |

Analise Exploratória

```

print("\nInformações gerais:")
df.info()

print("\nEstatísticas descritivas:")
display(df.describe())

print("\nValores nulos por coluna:")
print(df.isna().sum())

# Distribuição da nota geral
plt.figure(figsize=(8,4))
sns.histplot(df['review_overall'], bins=20, kde=True)
plt.title("Distribuição das notas gerais das cervejas")
plt.xlabel("Nota geral (0 a 5)")
plt.show()

# Top 10 estilos mais avaliados
top_styles = df['beer_style'].value_counts().nlargest(10)
plt.figure(figsize=(10,4))
sns.barplot(x=top_styles.values, y=top_styles.index)
plt.title("Top 10 estilos mais populares")
plt.xlabel("Número de avaliações")
plt.show()
```



```
Informações gerais:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1586614 entries, 0 to 1586613
Data columns (total 13 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   brewery_id            1586614 non-null  int64
1   brewery_name          1586599 non-null  object
2   review_time           1586614 non-null  int64
3   review_overall        1586614 non-null  float64
4   review_aroma          1586614 non-null  float64
5   review_appearance     1586614 non-null  float64
6   review_profilename    1586266 non-null  object
7   beer_style            1586614 non-null  object
```

Pré Processamento

```
# Remover linhas com valores nulos críticos
df = df.dropna(subset=['review_overall', 'review_aroma', 'review_taste', 'review_palate', 'beer_abv', 'beer_style'])

# Criar variável alvo: 1 se nota >= 4, caso contrário 0
df['positive_review'] = (df['review_overall'] >= 4).astype(int)

# Selecionar features numéricas
features = ['review_aroma', 'review_appearance', 'review_palate', 'review_taste', 'beer_abv']
X = df[features]
y = df['positive_review']

# Divisão treino/teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)

# Padronização
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

75% 2.372000e+03 1.288568e+09 4.500000e+00 4.000000e+00 4.000000e+00 4.000000e+00 4.500000e+00 8.500000e+00 3.9

Modelagem Supervisionada

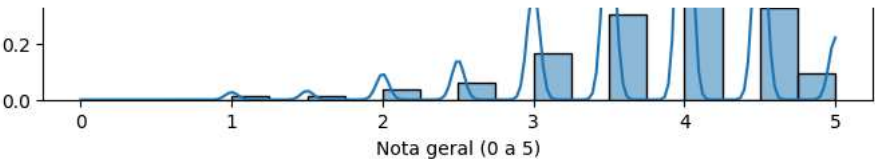
max 2.809300e+04 1.326285e+09 5.000000e+00 5.000000e+00 5.000000e+00 5.000000e+00 5.000000e+00 5.770000e+01 7.7

```
models = {
    "Logistic Regression": LogisticRegression(max_iter=1000),
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42),
    "XGBoost": xgb.XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)
}

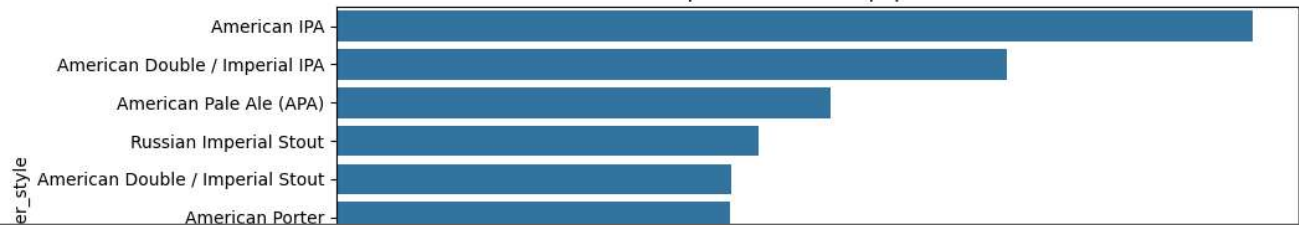
results = {}
for name, model in models.items():
    model.fit(X_train_scaled, y_train)
    preds = model.predict(X_test_scaled)
    probs = model.predict_proba(X_test_scaled)[: ,1]

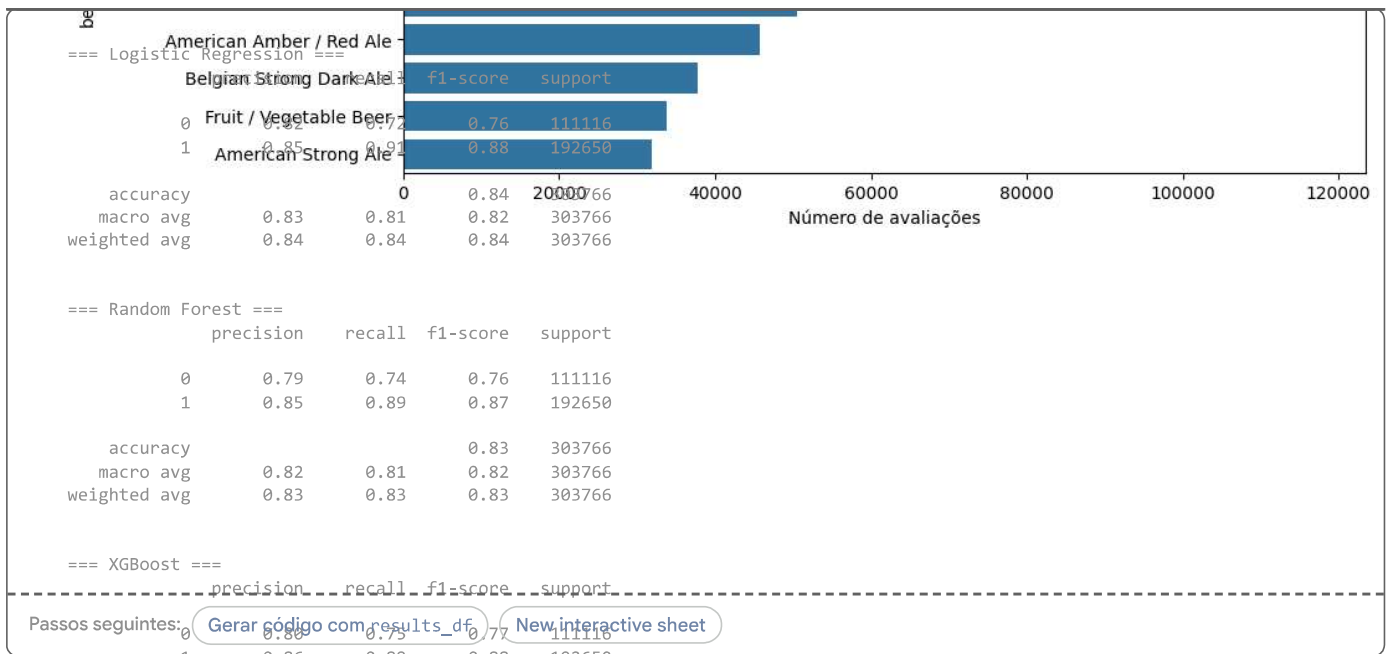
    results[name] = {
        "Accuracy": accuracy_score(y_test, preds),
        "Precision": precision_score(y_test, preds),
        "Recall": recall_score(y_test, preds),
        "F1": f1_score(y_test, preds),
        "ROC-AUC": roc_auc_score(y_test, probs)
    }
    print(f"\n=== {name} ===")
    print(classification_report(y_test, preds))

# Comparar métricas
results_df = pd.DataFrame(results).T.sort_values("ROC-AUC", ascending=False)
print("\nResumo de desempenho:")
display(results_df)
```



Top 10 estilos mais populares





✓ Análise Não Supervisionada - (K-MEANS)

```

macro avg 0.83 0.82 0.82 303766
weighted avg 0.84 0.84 0.84 303766

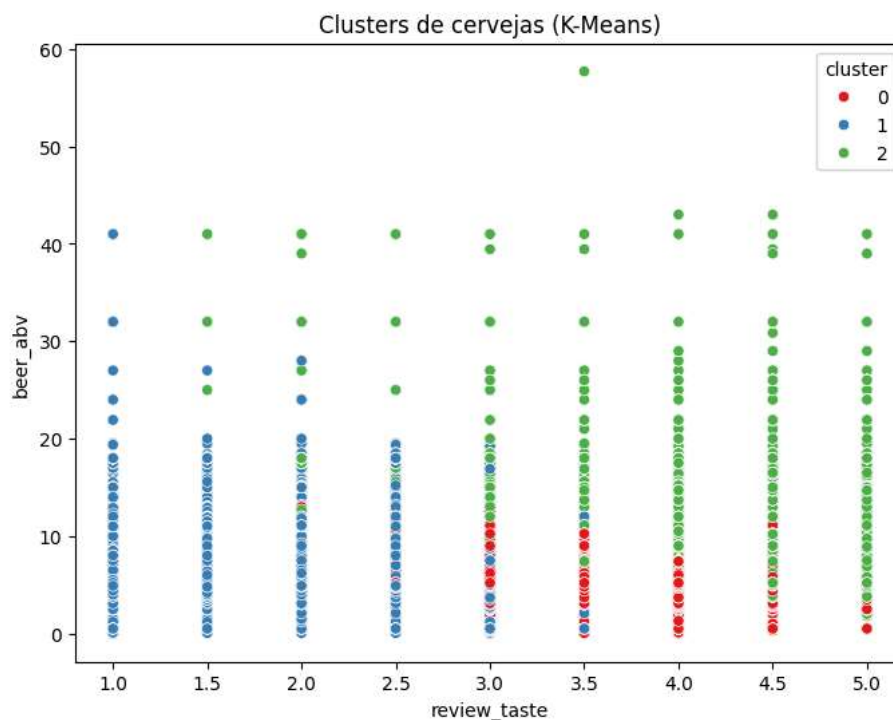
```

```

kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X_train_scaled)
df["cluster"] = kmeans.predict(scaler.transform(X[features]))

plt.figure(figsize=(8,6))
sns.scatterplot(x="review_taste", y="beer_abv", hue="cluster", data=df, palette="Set1")
plt.title("Clusters de cervejas (K-Means)")
plt.show()

```



✓ Conclusões

```

best_model = results_df.index[0]
print(f"\n✓ Melhor modelo: {best_model}")
print("Maior desempenho ROC-AUC:", round(results_df.loc[best_model, 'ROC-AUC'], 3))

```