Lista IA

1. Pré-processamento de Dados

Para começar, carreguei as bases de treino e teste (train.csv e test.csv). A análise exploratória foi inicialmente feita sobre o conjunto de treino, pois ele contém os rótulos de sobrevivência. Quanto ao tratamento de dados ausentes, preenchi a idade (Age) com a mediana, atribuí o valor mais comum à coluna de embarque (Embarked) e descartei a coluna Cabin devido ao excesso de valores faltantes. Transformei as variáveis categóricas Sex e Embarked para numéricas usando codificação por rótulos (Label Encoding). Além disso, criei uma nova variável chamada FamilySize, que representa o tamanho da família.

2. Análise Exploratória de Dados

Realizei uma análise exploratória da base, observando aspectos como a distribuição entre homens e mulheres, faixas etárias, classes sociais e taxas de sobrevivência. Para isso, utilizei principalmente gráficos de barras e histogramas, que ajudaram a visualizar como essas variáveis se relacionam com a chance de sobrevivência dos passageiros.

3. Modelagem com Algoritmos de Classificação

Implementei modelos de aprendizado supervisionado, incluindo Regressão Logística, Árvore de Decisão e Random Forest. A avaliação dos modelos foi feita com base em métricas como acurácia, precisão, recall e F1-score. Entre os algoritmos testados, o Random Forest foi o que obteve os melhores resultados, destacando-se em termos de performance preditiva.

4. Modelagem com Algoritmos de Agrupamento

Utilizei o algoritmo KMeans para realizar a clusterização dos passageiros. Para definir o número ideal de grupos, apliquei o método do cotovelo, analisando a inércia dos clusters. Após a segmentação, examinei como os agrupamentos se relacionavam com a variável de sobrevivência, buscando identificar padrões interessantes entre os grupos formados.

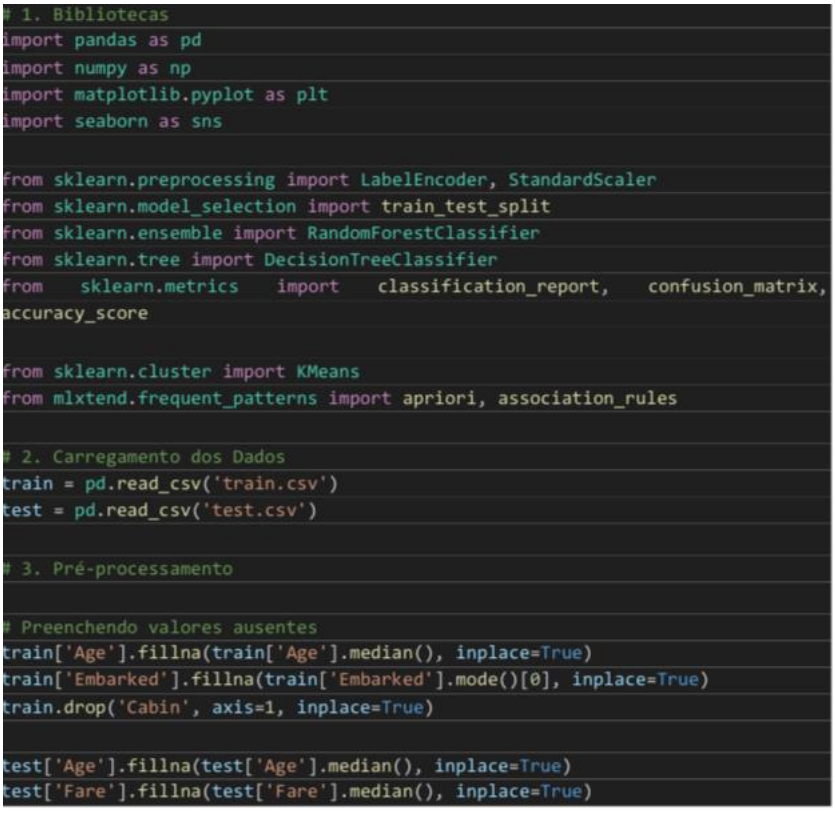
5. Extração de Regras de Associação

Implementei o algoritmo Apriori para extrair regras de associação a partir do conjunto de dados. Defini limites mínimos para suporte e confiança, a fim de filtrar as regras mais relevantes. Entre os padrões descobertos, destaquei que passageiros da primeira classe apresentaram maior probabilidade de sobrevivência, evidenciando uma associação importante entre classe social e o desfecho da viagem.

6. Conclusão

Em resumo, esta atividade proporcionou a aplicação prática de diversas técnicas fundamentais de Inteligência Artificial, incluindo o pré-processamento de dados, a construção de modelos supervisionados, a execução de algoritmos de clusterização e a extração de regras de associação. O trabalho reforçou a importância de organizar essas etapas dentro de um pipeline bem estruturado, essencial para a resolução eficiente de problemas reais com dados.

Código:github.com/n4lberth/IA



```python
# 1. Bibliotecas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score

from sklearn.cluster import KMeans
from mlxtend.frequent_patterns import apriori, association_rules

# 2. Carregamento dos Dados
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')

# 3. Pré-processamento

# Preenchendo valores ausentes
train['Age'].fillna(train['Age'].median(), inplace=True)
train['Embarked'].fillna(train['Embarked'].mode()[0], inplace=True)
train.drop('Cabin', axis=1, inplace=True)

test['Age'].fillna(test['Age'].median(), inplace=True)
test['Fare'].fillna(test['Fare'].median(), inplace=True)
```

```
est.drop('Cabin', axis=1, inplace=True)

 Codificação de variáveis categóricas
e = LabelEncoder()
or col in ['Sex', 'Embarked']:
   train[col] = le.fit_transform(train[col])
   test[col] = le.transform(test[col])

 Engenharia de atributos
rain['FamilySize'] = train['SibSp'] + train['Parch'] + 1
est['FamilySize'] = test['SibSp'] + test['Parch'] + 1

rain['Title'] = train['Name'].str.extract(r' ([A-Za-z]+)\.', expand=False)
est['Title'] = test['Name'].str.extract(r' ([A-Za-z]+)\.', expand=False)

 Simplificando títulos
or dataset in [train, test]:
   dataset['Title']  =  dataset['Title'].replace(['Lady',  'Countess','Capt',
Col',\
    'Don', 'Dr', 'Major', 'Rev', 'Sir', 'Jonkheer', 'Dona'], 'Rare')
   dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
   dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
   dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')
   dataset['Title'] = le.fit_transform(dataset['Title'])

 4. Análise Exploratória
lt.figure(figsize=(6,4))
ns.countplot(x='Survived', hue='Sex', data=train)
lt.title('Sobreviventes por Sexo')
lt.show()

lt.figure(figsize=(6,4))
ns.histplot(train['Age'], bins=30, kde=True)
lt.title('Distribuição de Idade')
lt.show()

 5. Modelagem Supervisionada

eatures = ['Pclass', 'Sex', 'Age', 'Fare', 'Embarked', 'FamilySize', 'Title']
 = train[features]
 = train['Survived']

_train,  X_val,  y_train,  y_val  =  train_test_split(X,  y,  test_size=0.2,
andom_state=42)
```

```
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_val)
print("Random Forest:\n", classification_report(y_val, y_pred_rf))

# Decision Tree
dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)
y_pred_dt = dt.predict(X_val)
print("Decision Tree:\n", classification_report(y_val, y_pred_dt))

# 6. Clusterização (KMeans)

scaler = StandardScaler()
X_cluster = scaler.fit_transform(train[features])

kmeans = KMeans(n_clusters=2, random_state=42)
clusters = kmeans.fit_predict(X_cluster)
train['Cluster'] = clusters

plt.figure(figsize=(6,4))
sns.scatterplot(x='Age', y='Fare', hue='Cluster', data=train, palette='Set1')
plt.title('Clusters por Idade e Tarifa')
plt.show()

# 7. Regras de Associação

# Para regras de associação, precisamos de variáveis binárias
assoc_data = train.copy()
assoc_data['Sex'] = assoc_data['Sex'].map({0: 'male', 1: 'female'})
assoc_data['Survived'] = assoc_data['Survived'].map({0: 'not_survived', 1:
'survived'})
assoc_data['Pclass'] = assoc_data['Pclass'].astype(str)
assoc_data['FamilySize'] = assoc_data['FamilySize'].astype(str)
assoc_data['Title'] = assoc_data['Title'].astype(str)
assoc_data = assoc_data[['Sex', 'Pclass', 'Survived', 'FamilySize', 'Title']]

# Transformação para formato one-hot
assoc_bin = pd.get_dummies(assoc_data)

# Apriori e regras
frequent = apriori(assoc_bin, min_support=0.1, use_colnames=True)
rules = association_rules(frequent, metric="confidence", min_threshold=0.6)
print(rules[['antecedents', 'consequents', 'support', 'confidence',
'lift']].sort_values('lift', ascending=False).head(5))
```

.