Atividades de Pré-processamento com Seaborn, Pandas, Scikit-learn e o Dataset Titanic

Prof. Dr. Fabiano B. Menegidio

1 Introdução

Neste documento, exploraremos 25 atividades que envolvem o uso das bibliotecas **Pandas**, **Seaborn**, **Scikit-learn** e o dataset **Titanic**. As atividades são focadas em tarefas de pré-processamento de dados, essenciais para preparar os dados para a modelagem de aprendizado de máquina. Cada atividade será resolvida detalhadamente com explicações de cada função usada.

2 Atividades Resolvidas

2.1 Atividade 1: Carregar o Dataset Titanic com o Seaborn e visualizar as primeiras linhas

```
import seaborn as sns
import pandas as pd

# Carregar o dataset Titanic
df = sns.load_dataset('titanic')

# Exibir as primeiras 5 linhas
print(df.head())
```

Explicação:

- O `sns.load_dataset()` carrega um dataset embutido no Seaborn. Aqui, estamos carregando o dataset Titanic.
- O comando `df.head()` exibe as primeiras 5 linhas do dataset.

2.2 Atividade 2: Identificar valores ausentes no dataset

```
# Verificar valores nulos por coluna
print(df.isnull().sum())
```

Explicação:

- O método `isnull()` retorna um dataframe booleano indicando se um valor está ausente.
- O `sum()` soma os valores nulos por coluna.

2.3 Atividade 3: Remover linhas com valores ausentes

```
# Remover linhas com valores ausentes
df_cleaned = df.dropna()
print(df_cleaned.isnull().sum())
```

Explicação:

- `dropna()` remove todas as linhas que contêm valores nulos.
- A função `isnull().sum()` é usada novamente para verificar se há valores ausentes após a remoção.

2.4 Atividade 4: Preencher valores ausentes com a média

```
from sklearn.impute import SimpleImputer

# Preencher valores ausentes na coluna 'age' com a m dia
imputer = SimpleImputer(strategy='mean')
df['age'] = imputer.fit_transform(df[['age']])
```

Explicação:

- `SimpleImputer` é uma função do `scikit-learn` usada para preencher valores ausentes. Aqui usamos a estratégia da média.
- `fit_transform()` ajusta o imputador aos dados e preenche os valores nulos.

2.5 Atividade 5: Criar um gráfico de contagem de passageiros por sexo

```
# Criar gr fico de contagem por sexo
sns.countplot(x='sex', data=df)
```

Explicação:

- `countplot()` cria um gráfico de contagem que exibe quantos homens e quantas mulheres estavam a bordo.
- O parâmetro `data` define o dataframe usado, e `x` define a variável categórica.

2.6 Atividade 6: Criar um boxplot da distribuição de tarifas por classe de passageiros

```
# Boxplot para distribui  o de tarifas por classe
sns.boxplot(x='class', y='fare', data=df)
```

Explicação:

- `boxplot()` mostra a distribuição de `fare` (tarifa) para cada classe de passageiros.

2.7 Atividade 7: Aplicar Min-Max Scaling na coluna de tarifas

```
from sklearn.preprocessing import MinMaxScaler

# Normalizar a coluna 'fare'
scaler = MinMaxScaler()
df['fare_scaled'] = scaler.fit_transform(df[['fare']])
```

Explicação:

- `MinMaxScaler()` ajusta os dados no intervalo [0, 1].
- `fit_transform()` aplica a transformação aos dados.

2.8 Atividade 8: Criar gráfico de dispersão entre idade e tarifa, colorido por sexo

```
# Gr fico de dispers o entre 'age' e 'fare', colorido por sexo
sns.scatterplot(x='age', y='fare', hue='sex', data=df)
```

Explicação:

- `scatterplot()` exibe a relação entre duas variáveis numéricas.
- O argumento `hue` adiciona cores diferentes para cada valor de `sex`.

2.9 Atividade 9: One-hot encoding para a variável 'embarked'

```
# Aplicar One-hot encoding  coluna 'embarked'
df_encoded = pd.get_dummies(df, columns=['embarked'], drop_first=True)
```

Explicação:

- `get_dummies()` cria colunas binárias para variáveis categóricas.
- O parâmetro `drop_first=True` remove a primeira coluna para evitar multicolinearidade.

2.10 Atividade 10: Criar um gráfico de violin plot para a distribuição de idade por sobreviventes

```
# Criar violin plot para idade, separado por sobreviv ncia
sns.violinplot(x='survived', y='age', data=df)
```

Explicação:

- `violinplot()` mostra a distribuição da idade (`age`) de acordo com a variável de sobrevivência (`survived`).

2.11 Atividade 11: Criar um heatmap da matriz de correlação

```
# Calcular e exibir a matriz de correla   o como um heatmap
corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
```

Explicação:

- `corr()` calcula a matriz de correlação entre as variáveis numéricas.
- `heatmap()` exibe essa matriz como um mapa de calor, com o parâmetro `annot=True` exibindo os valores numéricos.

2.12 Atividade 12: Aplicar Standard Scaling na coluna 'age'

```
from sklearn.preprocessing import StandardScaler

# Padronizar a coluna 'age'
scaler = StandardScaler()
df['age_scaled'] = scaler.fit_transform(df[['age']])
```

Explicação:

- `StandardScaler()` padroniza os dados, ajustando-os para que tenham média 0 e desvio padrão 1.

2.13 Atividade 13: Separar o dataset em treino e teste

```
from sklearn.model_selection import train_test_split

# Separar as vari veis independentes (X) e dependente (y)
X = df.drop(columns=['survived'])
y = df['survived']

# Dividir em treino e teste (80% treino, 20% teste)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random
```

Explicação:

- `train_test_split()` divide os dados em conjuntos de treino e teste.
- O parâmetro `test_size=0.2` define que 20% dos dados serão usados para teste, e 80% para treino.

2.14 Atividade 14: Criar pairplot para variáveis numéricas selecionadas

```
# Criar pairplot para 'age', 'fare' e 'pclass'
sns.pairplot(df[['age', 'fare', 'pclass']])
```

Explicação:

- `pairplot()` exibe gráficos de dispersão para todas as combinações possíveis das variáveis numéricas fornecidas.

2.15 Atividade 15: Criar gráfico de barras com a média da tarifa por classe

```
# Gr fico de barras para a m dia de tarifa por classe
sns.barplot(x='class', y='fare', data=df)
```

Explicação:

- `barplot()` cria um gráfico de barras que mostra a média de `fare` (tarifa) para cada classe (`class`).

2.16 Atividade 16: Criar uma matriz de confusão para o conjunto de teste

```
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Prever (exemplo com previs es aleat rias)
y_pred = [0] * len(y_test)

# Criar a matriz de confus o
cm = confusion_matrix(y_test, y_pred)

# Exibir a matriz de confus o
sns.heatmap(cm, annot=True, fmt="d")
```

Explicação:

- `confusion_matrix()` calcula a matriz de confusão, uma tabela que mostra as previsões corretas e incorretas.
- O `heatmap()` exibe a matriz de confusão como um gráfico de calor.

2.17 Atividade 17: Preencher valores ausentes na coluna 'age' com a mediana

```
# Preencher valores ausentes na coluna 'age' com a mediana
imputer = SimpleImputer(strategy='median')
df['age'] = imputer.fit_transform(df[['age']])
```

Explicação:

- Usamos a estratégia `median` para preencher os valores ausentes com a mediana.

2.18 Atividade 18: Remover outliers da coluna 'fare' usando o IQR

```
# Calcular o IQR (Interquartile Range)
Q1 = df['fare'].quantile(0.25)
Q3 = df['fare'].quantile(0.75)
IQR = Q3 - Q1

# Remover outliers
df = df[~((df['fare'] < (Q1 - 1.5 * IQR)) | (df['fare'] > (Q3 + 1.5 * IQR)))]
```

Explicação:

- O IQR é a diferença entre o primeiro quartil (`Q1`) e o terceiro quartil (`Q3`).
- Valores abaixo de `Q1 - 1.5 * IQR` ou acima de `Q3 + 1.5 * IQR` são considerados outliers e são removidos.

2.19 Atividade 19: Detectar multicolinearidade com a matriz de correlação

```
# Criar matriz de correla  o
corr_matrix = df.corr()

# Detectar colinearidade alta (acima de 0.8)
high_corr = corr_matrix[corr_matrix > 0.8]
print(high_corr)
```

Explicação:

- A correlação maior que 0.8 entre variáveis sugere colinearidade. Isso pode afetar negativamente alguns modelos de machine learning.

2.20 Atividade 20: Criar gráficos de dispersão por classe usando FacetGrid

```
# Criar um grid de gr ficos separados por classe
g = sns.FacetGrid(df, col="class")
g.map(sns.scatterplot, "age", "fare")
```

Explicação:

- `FacetGrid()` cria múltiplos gráficos separados por categorias. Aqui, criamos gráficos de dispersão para cada classe (`class`).

2.21 Atividade 21: Normalizar a coluna 'age' usando a normalização Z-score

```
# Normalizar a coluna 'age' com Z-score
df['age_zscore'] = (df['age'] - df['age'].mean()) / df['age'].std()
```

Explicação:

- A normalização Z-score transforma os dados para que a média seja 0 e o desvio padrão seja 1.

2.22 Atividade 22: Criar um gráfico de linha para a média de tarifa por idade

```
# Calcular a m dia de tarifa por idade
age_fare_mean = df.groupby('age')['fare'].mean().reset_index()

# Gr fico de linha mostrando a m dia da tarifa por idade
sns.lineplot(x='age', y='fare', data=age_fare_mean)
```

Explicação:

- O `lineplot()` exibe a variação da tarifa (`fare`) em função da idade (`age`).

2.23 Atividade 23: Criar um gráfico de barras empilhadas mostrando a contagem de passageiros por classe e sexo

```
# Gr fico de barras empilhadas por classe e sexo
sns.countplot(x='class', hue='sex', data=df)
```

Explicação:

- O `countplot()` com `hue='sex'` cria um gráfico de barras empilhadas separando as classes (`class`) por sexo (`sex`).

2.24 Atividade 24: Aplicar Label Encoding na coluna 'sex'

```
from sklearn.preprocessing import LabelEncoder

# Aplicar Label Encoding
le = LabelEncoder()
df['sex_encoded'] = le.fit_transform(df['sex'])
```

Explicação:

- `LabelEncoder()` transforma uma coluna categórica em valores numéricos (ex: `male = 1`, `female = 0`).

2.25 Atividade 25: Usar SelectKBest para selecionar as 5 melhores features

```
from sklearn.feature_selection import SelectKBest, f_classif

# Selecionar as 5 melhores variaveis preditivas
X = df.drop(columns=['survived'])
y = df['survived']
selector = SelectKBest(score_func=f_classif, k=5)
X_new = selector.fit_transform(X, y)
print(X_new)
```

Explicação:

- `SelectKBest()` seleciona as `k` melhores variáveis preditivas com base no teste estatístico de ANOVA (`f_classif`).

3 Atividades Propostas para Prática

Aqui estão 10 atividades para prática, sem solução. Resolva usando o dataset Titanic e as bibliotecas Pandas, Seaborn, e Scikit-learn.

1. Remover valores ausentes da coluna `embarked` e substituí-los pela moda.
2. Criar gráficos de contagem para as variáveis `pclass` e `survived`.
3. Aplicar o RobustScaler nas colunas `age` e `fare`.
4. Criar um gráfico de dispersão entre `fare` e `age`, separado por sexo e classe.
5. Criar um boxplot da variável `age` separada por `pclass` e `survived`.
6. Detectar e remover outliers da coluna `age`.
7. Criar um gráfico de barras mostrando a média da idade por sexo e classe.

8. Aplicar One-hot encoding na variável `pclass`.

9. Criar gráficos de dispersão para as variáveis `fare`, `age` e `pclass` usando `pairplot`.

10. Usar o método `train_test_split()` para dividir os dados em treino (70%) e teste (30%).