Lab 03 - Principais abordagens em IA - aprendizagem de máquina (Machine Learning)

Ex 1: Pandas, Regressão Logistica e Titanic

Vamos fazer um modelo de regressão logística com o dataset Titanic que permita prever se um individuo sobreviveu ou não.

1. Importação das Bibliotecas

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
import seaborn as sns
import matplotlib.pyplot as plt
```

2. Carregar e Inspecionar o Dataset Titanic

Carregar o dataset Titanic diretamente da internet e fazer uma inspeção inicial.

```
# Carregar o dataset Titanic
url =
'https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv'
df = pd.read_csv(url)

# Visualizar as primeiras linhas e as informações gerais
print(df.head())
print(df.info())
print(df.describe())
```

3. Análise Exploratória e Limpeza de Dados

a) Análise de Valores Ausentes

Vamos identificar colunas com valores ausentes e tomar decisões para tratá-los.

```
# Contagem de valores ausentes
print("Valores Ausentes por Coluna:")
print(df.isnull().sum())
```

b) Tratamento de Valores Ausentes e Limpeza de Dados

- **Idade (Age)**: Preenchimento com a mediana, pois a idade pode ter uma grande variação.
- **Cabine (Cabin)**: Remoção da coluna, pois possui muitos valores ausentes.
- **Porto de Embarque (Embarked)**: Substituição pelo valor mais frequente.

```
# Preencher valores ausentes em 'Age' com a mediana
df['Age'].fillna(df['Age'].median(), inplace=True)

# Remover coluna 'Cabin' devido à alta quantidade de valores ausentes
df.drop(['Cabin'], axis=1, inplace=True)

# Preencher valores ausentes em 'Embarked' com o valor mais frequente
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)
```

c) Conversão de Variáveis Categóricas para Numéricas

Convertemos variáveis categóricas (e.g., 'Sex', 'Embarked') em variáveis dummy para o modelo.

```
# Converter 'Sex' e 'Embarked' em variáveis dummy
df = pd.get_dummies(df, columns=['Sex', 'Embarked'], drop_first=True)
```

d) Remoção de Colunas Não Relevantes

Remover colunas que não têm valor preditivo direto para o modelo, como ‘PassengerId’, ‘Name’, e ‘Ticket’.

```
# Remover colunas irrelevantes
df.drop(['PassengerId', 'Name', 'Ticket'], axis=1, inplace=True)
```

4. Divisão dos Dados em Treino, Validação e Teste

Divida o dataset em três partes: treino (60%), validação (20%) e teste (20%).

```
# Dividir dados em X e y
X = df.drop('Survived', axis=1)
y = df['Survived']

# Dividir em treino e temp (80%) e teste (20%)
X_train_temp, X_test, y_train_temp, y_test = train_test_split(X, y, test_size=0.2,
```

```
random_state=42)

# Dividir novamente o conjunto de treino temporário em treino (60%) e validação
(20%)
X_train, X_val, y_train, y_val = train_test_split(X_train_temp, y_train_temp,
test_size=0.25, random_state=42)
```

5. Treino do Modelo de Regressão Logística

```
# Inicializar o modelo de regressão logística
modelo = LogisticRegression(max_iter=200)

# Treinar o modelo
modelo.fit(X_train, y_train)
```

6. Avaliação do Modelo

a) Avaliação com o Conjunto de Validação

Calcule a precisão e outros indicadores com o conjunto de validação.

```
# Previsões no conjunto de validação
y_val_pred = modelo.predict(X_val)

# Avaliar precisão e exibir o relatório de classificação
print(f"Precisão no Conjunto de Validação: {accuracy_score(y_val,
y_val_pred):.2f}")
print("Relatório de Classificação (Validação):")
print(classification_report(y_val, y_val_pred))
```

b) Avaliação Final com o Conjunto de Teste

Após validação, use o conjunto de teste para avaliação final.

```
# Previsões no conjunto de teste
y_test_pred = modelo.predict(X_test)

# Avaliar precisão e exibir o relatório de classificação
print(f"Precisão no Conjunto de Teste: {accuracy_score(y_test, y_test_pred):.2f}")
print("Relatório de Classificação (Teste):")
print(classification_report(y_test, y_test_pred))
```

7. Previsão para um Novo Caso

Para prever a sobrevivência de um passageiro com informações fictícias, crie um novo dataframe de entrada.

Novo Passageiro Exemplo

Um passageiro hipotético:

- Pclass: 2
- Age: 28
- SibSp: 0
- Parch: 0
- Fare: 20
- Sex_male: 1
- Embarked_Q: 0
- Embarked_S: 1

```
# Criar um novo caso para previsão
novo_caso = pd.DataFrame({
    'Pclass': [2],
    'Age': [28],
    'SibSp': [0],
    'Parch': [0],
    'Fare': [20],
    'Sex_male': [1],
    'Embarked_Q': [0],
    'Embarked_S': [1]
})

# Fazer a previsão
sobreviveu = modelo.predict(novo_caso)
print(f"Previsão de Sobrevivência do Novo Passageiro: {'Sobreviveu' if
sobreviveu[0] == 1 else 'Não Sobreviveu'}")
```

8. Análise dos Coeficientes

Examine os coeficientes do modelo para entender o impacto das variáveis.

```
# Coeficientes do modelo
coeficientes = pd.DataFrame(modelo.coef_, columns=X.columns)
print("Coeficientes do Modelo:")
print(coeficientes.T)
```

Ex 2: Trabalhar dados, sem se saber o que são

Mas fazer a limpar os dados de um dataset desconhecido! **Mas cuidado que desta vez o código não é só copy-paste**

Passo 1: Instalação e Importação dos Pacotes

Tarefa: Instalar e carregar as bibliotecas necessárias. Vamos usar `pandas` para manipulação de dados, `numpy` para operações matemáticas e `pickle` para guardar os resultados.

```
import pandas as pd
import numpy as np
import pickle
```

Passo 2: Carregar o Conjunto de Dados

Tarefa: Carregue o ficheiro CSV que contém os dados e verifica a sua estrutura.

```
# Carregar o dataset
df = pd.read_csv("dataset.csv")

# Verificar a estrutura dos dados
df.shape
df.head()
```

Passo 3: Análise Exploratória dos Dados

Tarefa: Explore os dados para detetar potenciais problemas como valores nulos e faça a conversão da variável alvo para numérica.

```
# Verificar se há valores em falta
df.isnull().values.any()

# Converter a variável LABEL_TARGET para valores numéricos
df["LABEL_TARGET"] = df["LABEL_TARGET"].astype(int)
```

Passo 4: Divisão dos Dados em Treino, Validação e Teste

Tarefa: Divida os dados em subconjuntos de treino (70%), teste (15%) e validação (15%) para garantir que os modelos são testados corretamente.

```
# Criar subconjuntos com amostragem aleatória
df_sample_30 = df.sample(frac=0.3)

# Conjunto de Teste e Validação
df_test = df_sample_30.sample(frac=0.5)
df_valid = df_sample_30.drop(df_test.index)

# Conjunto de Treino
df_train = df.drop(df_sample_30.index)
```

Passo 5: Balanceamento de Classes

Tarefa: Muitas vezes, os conjuntos de dados têm classes desequilibradas. Vamos aplicar subamostragem para balancear as classes.

```
# Separar as classes positivas e negativas
df_train_pos = df_train[df_train.LABEL_TARGET == 1]
df_train_neg = df_train[df_train.LABEL_TARGET == 0]

# Encontrar o tamanho mínimo entre as duas classes
min_value = min(len(df_train_pos), len(df_train_neg))

# Subamostragem para balancear as classes
df_train_final = pd.concat([df_train_pos.sample(n=min_value),
df_train_neg.sample(n=min_value)])

# Verificar o balanceamento das classes
df_train_final.LABEL_TARGET.value_counts()
```

Passo 6: Guardar os Resultados

Tarefa: Guarde os conjuntos de treino, validação e teste para uso futuro. Também vamos guardar os nomes das colunas preditoras para facilitar o carregamento posterior.

```
# Guardar os datasets em CSV
df_train_final.to_csv('train_final_data.csv', index=False)
df_valid.to_csv('validation_data.csv', index=False)
df_test.to_csv('test_data.csv', index=False)

# Guardar os nomes das colunas de preditores
input_columns = df.columns.tolist()
pickle.dump(input_columns, open('input_columns.sav', 'wb'))
```