Exercício Inteligência Artificial

Utilizaremos a base de dados **Titanic** do Kaggle: Titanic Dataset - Kaggle. Ela contém informações sobre os passageiros do Titanic, como idade, sexo, classe e se eles sobreviveram ou não.

Instruções para baixar a base de dados:

https://www.kaggle.com/c/titanic/data

1. Acesse o link acima.
2. Faça o download dos arquivos CSV.
3. Carregue os dados no Google Colab, seguindo as orientações dadas em sala.

Instruções para o Exercício

Parte 1: Pré-processamento dos Dados

1. **Carregar a Base de Dados no Google Colab:**
   - Utilize a biblioteca `pandas` para carregar o arquivo CSV.
   - Explore os dados usando comandos como `df.head()`, `df.info()` e `df.describe()`.
2. **Limpeza de Dados:**
   - Identifique e trate valores ausentes (nulos) usando `df.isnull().sum()`.
   - Você pode remover linhas com valores nulos ou preencher esses valores com a média ou mediana das colunas numéricas, como `Age`.
3. **Seleção de Variáveis:**
   - Use as seguintes colunas para análise:
     - `Pclass` (classe do passageiro)
     - `Sex` (gênero)
     - `Age` (idade)
     - `Fare` (valor da tarifa)
     - `Survived` (nosso "target" - indica se a pessoa sobreviveu).
   - Converta variáveis categóricas (como `Sex`) em números usando `pd.get_dummies()`.
4. **Divisão dos Dados:**
   - Divida os dados em variáveis de entrada (`X`) e saída (`y`). `X` deve conter as colunas `Pclass`, `Sex`, `Age` e `Fare`, e `y` deve conter `Survived`.

Parte 2: Implementação Simples de um Algoritmo de Classificação (k-NN)

Agora vamos utilizar um algoritmo simples de classificação, o **k-Nearest Neighbors (k-NN)**, para fazer previsões. O algoritmo k-NN é um método básico de classificação que decide a classe de um exemplo baseado nos "vizinhos" mais próximos.

1. **Importar e Treinar o Modelo:**

Utilize a biblioteca `scikit-learn` para importar o algoritmo k-NN:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
```

Divida os dados entre treino e teste (70% para treino, 30% para teste):

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)
```

Crie e treine o modelo k-NN com `k=3`:

```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
```

2. **Fazer Previsões:**

Use o modelo treinado para prever as classes no conjunto de teste:
python
Copiar código

```
y_pred = knn.predict(X_test)
```

Parte 3: Avaliação de Desempenho

Depois de treinar o modelo e fazer previsões, vamos avaliar o desempenho do nosso modelo usando algumas métricas simples.

1. **Avaliar Acurácia:**

A acurácia mede quantas previsões o modelo acertou em relação ao total de previsões.

```
from sklearn.metrics import accuracy_score
```

```
accuracy = accuracy_score(y_test, y_pred)
print(f'Acurácia: {accuracy:.2f}')
```

2. **Matriz de Confusão:**

A matriz de confusão nos mostra o desempenho do modelo ao comparar as classificações corretas e incorretas.

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

3. **Conclusões:**
    - Documente no `README.md` as suas observações sobre o desempenho do modelo, a acurácia e o que a matriz de confusão revela sobre as previsões feitas. Explique, com base nas métricas, como o modelo se saiu e o que poderia ser melhorado (como testar diferentes valores de "k").

Parte 4: Criação do Repositório no GitHub

1. **Configuração do Repositório:**
    - Criem um repositório público no GitHub com o nome `classificacao-titanic`.
    - Incluam o arquivo `.ipynb` do Google Colab com o código do exercício.
    - Adicionem um arquivo `README.md` explicando brevemente os seguintes pontos:
        - Carregamento e pré-processamento dos dados.
        - Implementação do algoritmo k-NN.
        - Avaliação de desempenho (acurácia e matriz de confusão).
        - Observações sobre os resultados e o que pode ser melhorado.

O Que Deve Ser Entregue:

- O código Python completo no Google Colab, incluindo:
    1. Carregamento e exploração dos dados.
    2. Limpeza e pré-processamento.
    3. Implementação do algoritmo k-NN.
    4. Avaliação de desempenho usando acurácia e matriz de confusão.
- O repositório GitHub contendo:

1. O arquivo `.ipynb` com o código do exercício.
2. O arquivo `README.md` com a explicação do processo e observações sobre os resultados.

Dicas e Sugestões

- Pesquisem sobre a função `pd.get_dummies()` para transformar variáveis categóricas em numéricas.
- Para lidar com valores ausentes, vocês podem usar `df.fillna()` ou `df.dropna()`.
- Testem diferentes valores para "k" no algoritmo k-NN e vejam como isso impacta o desempenho.
- Utilizem a documentação do `scikit-learn` como referência para entender as funções usadas.

Entrega

Vocês têm **1 semana** para realizar este exercício e enviar o link do repositório GitHub na plataforma.

Data: 16/10/2024

Objetivo Final

Esse exercício tem como foco introduzir vocês ao fluxo completo de trabalho com dados, desde o carregamento até a implementação de um algoritmo de classificação simples e sua avaliação. Na próxima aula, aprofundaremos mais sobre como os algoritmos funcionam e como ajustar seus parâmetros para melhorar o desempenho.

Boa sorte e bom trabalho!