Documentação do Código de Análise de Dados de Pokémons

Este código realiza uma análise de dados utilizando um dataset de Pokémons, com o objetivo de prever se um Pokémon é lendário ou não. A análise segue as etapas descritas abaixo:

1. Carregamento dos Dados

* Importa as bibliotecas necessárias: `pandas`, `numpy`, `sklearn`, `matplotlib` e `seaborn`.
* Carrega o arquivo `Pokemon.csv` usando `pd.read_csv()`.
* Exibe os 5 primeiros registros do DataFrame usando `df.head()`.
* Remove a coluna '#' do DataFrame usando `df.drop()`.
* Traduz e renomeia as colunas para uma melhor legibilidade.
* Exibe os 5 primeiros registros do DataFrame após a renomeação das colunas.

2. Transformação de Variáveis

* Avalia a necessidade de transformar variáveis em outras escalas. Neste caso, não há necessidade de escalar as variáveis, pois o dataset é pequeno e autoexplicativo.

3. Codificação de Variáveis Categóricas

* Utiliza `LabelEncoder` para transformar a variável categórica "lendario" em valores numéricos (0 e 1).
* Exibe os 5 primeiros registros do DataFrame após a codificação.

4. Normalização de Variáveis

* Realiza uma análise descritiva do DataFrame usando `df.describe()` para verificar valores discrepantes e outliers.
* Preenche os valores NaN da coluna "tipo_2" com "None" usando `df['tipo_2'].fillna()`.
* Verifica se há valores NaN no DataFrame usando `df.isna().sum()`.
* Utiliza `StandardScaler` para normalizar as variáveis numéricas do DataFrame, exceto "lendario" e "geracao".
* Cria um DataFrame com os valores normalizados usando `df_norm.copy()`.
* Exibe os 5 primeiros registros do DataFrame normalizado.
* Exibe os 5 primeiros registros do DataFrame original.
* Calcula a proporção de Pokémon lendários no dataset.
* Remove as colunas do tipo string do DataFrame normalizado para aplicar o SMOTE.

5. Balanceamento de Classes (Oversampling)

* Utiliza `SMOTE` (Synthetic Minority Over-sampling Technique) para balancear as classes, criando exemplos sintéticos para a classe minoritária (lendários).
* Separa as features (X) e o target (y).
* Divide os dados em conjuntos de treino e teste (80% treino e 20% teste).
* Normaliza os dados de treino e teste usando `MinMaxScaler`.
* Aplica o SMOTE aos dados de treino normalizados.
* Verifica as dimensões do conjunto de dados após o oversampling.
* Exibe alguns exemplos do conjunto de dados após o oversampling.
* Verifica a contagem de exemplos em cada classe após o oversampling.

6. Modelagem

* **SVM (Support Vector Machine):**
    * Treina um modelo SVM com kernel 'rbf' usando os dados balanceados.
    * Realiza previsões no conjunto de teste.
    * Imprime as métricas de avaliação: acurácia, revocação e precisão.
* **KNN (K-Nearest Neighbors):**
    * Treina um modelo KNN com k=3 usando os dados balanceados.
    * Realiza previsões no conjunto de teste.
    * Imprime as métricas de avaliação: acurácia, revocação e precisão.

7. Balanceamento de Classes (Undersampling)

* **Undersampling Aleatório:**
    * Define o número máximo de exemplos por classe.
    * Filtra as classes com mais exemplos do que o limite.
    * Seleciona aleatoriamente exemplos das classes maiores que o limite.
    * Combina os exemplos das classes maiores que o limite com os exemplos das classes menores que o limite.
    * O DataFrame resultante possui a classe "lendario" balanceada.
* **Undersampling por Centróides:**
    * Utiliza `ClusterCentroids` para realizar undersampling da classe majoritária, selecionando centróides dos clusters.
    * Verifica as dimensões do conjunto de dados após o undersampling.
    * Verifica a contagem de exemplos em cada classe após o undersampling.

8. Análise de Correlação

* Utiliza um gráfico de calor para visualizar a matriz de correlação entre as variáveis, utilizando os métodos de Pearson e Spearman.
* Utiliza `ppscore` para calcular a matriz de correlação de uma forma mais textual e detalhada, com mais informações sobre cada comparação entre as variáveis.

9. Seleção de Variáveis

* Utiliza `SelectKBest` com o score `chi2` para selecionar as k (4 neste caso) variáveis mais importantes para a previsão.
* Verifica quais variáveis foram selecionadas.

10. Visualização

* Cria um boxplot para visualizar a distribuição das características dos Pokémons.
* Cria um gráfico de barras para visualizar a quantidade de tipos por geração dos Pokémons.
* Cria um gráfico de barras para visualizar a importância das variáveis selecionadas pelo `SelectKBest`.

Observações

* O código inclui comentários explicando cada passo da análise.
* A análise é realizada utilizando diferentes técnicas de balanceamento de classes, como oversampling e undersampling.
* A seleção de variáveis é realizada utilizando o score `chi2`.
* O código inclui várias visualizações para auxiliar na análise dos dados.