**Explicação Machine Learning PI 3 (Formula 1)**

1. Previsão de proximas 10 melhores tempos por volta em certo circuito (em ms)

O código implementa um modelo de previsão de tempos de volta em corridas de Fórmula 1 usando uma rede neural recorrente LSTM. Os dados são mesclados e normalizados, preparados em sequências de 40 pontos, e o modelo é treinado com 50 épocas. As previsões para as próximas 10 corridas são desescalonadas e visualizadas em um gráfico, destacando o histórico e as previsões.

**Importação de Bibliotecas**:

O código começa importando bibliotecas como NumPy, pandas, Matplotlib e Keras, necessárias para manipulação de dados, visualização e construção de redes neurais.

**Mesclagem de Dados:**

Os dados das corridas e resultados são mesclados com base no ID da corrida, utilizando a função merge do pandas. A condição corridas\_x\_resultados['circuito\_id'] == 1 filtra os dados para o Circuito ID 1.

**Normalização dos Dados**:

Os tempos de volta são normalizados usando o Min-Max Scaling com a classe MinMaxScaler do scikit-learn.

**Preparação dos Dados para o Modelo:**

Divide os dados normalizados em sequências de 40 pontos para entrada no modelo.

As entradas (x\_train) e saídas (y\_train) do modelo são criadas com base nessas sequências.

As sequências de entrada (x\_train) e saídas (y\_train) são criadas a partir dos dados normalizados. O loop for cria sequências de 40 pontos de dados

**Reshape dos Dados para o TensorFlow:**

Os dados de entrada (x\_train) são remodelados para atender aos padrões de entrada do TensorFlow, adicionando uma dimensão.

**Criação do Modelo RNN (LSTM):**

* Um modelo sequencial é inicializado.
* Duas camadas LSTM são adicionadas para capturar dependências temporais nos dados.
* Uma camada densa com uma unidade é utilizada para a saída.
* O modelo é compilado com o otimizador 'adam' e a função de perda 'mean\_squared\_error'.

**Treinamento do Modelo:**

O modelo é treinado usando os dados de treinamento (x\_train e y\_train) por 50 épocas, com um tamanho de lote de 8.

**Previsão de Tempos de Volta Futuros:**

São feitas previsões para os próximos 10 tempos de volta usando o modelo treinado. O loop for realiza as previsões e atualiza a entrada para a próxima previsão.

**Desescalonamento das Previsões:**

As previsões normalizadas são desescalonadas para a escala original usando a função inverse\_transform do scaler.

**Criação de Índices para Plotagem:**

Índices são criados para plotar o histórico de tempos de volta e as previsões para as próximas corridas.

**Plotagem dos Resultados:**

Um gráfico Matplotlib é gerado para visualizar os tempos de volta históricos e as previsões para as próximas 10 corridas.

No geral, o código usa uma arquitetura LSTM para modelar padrões temporais nos tempos de volta das corridas e, após treinamento, faz previsões para os próximos tempos de volta, proporcionando uma representação visual do desempenho do modelo.

**\*\*Resumo das funcionalidades e função do código:\*\***

**1. \*\*Mesclagem de dados:\*\***

**- As corridas e resultados são mesclados usando o Pandas, onde a chave de junção é a coluna 'corrida\_id'.**

**- É criado um novo DataFrame `merged\_data` contendo informações combinadas das corridas e resultados para o circuito com 'circuito\_id' igual a 1.**

**2. \*\*Normalização de dados:\*\***

**- Os dados da coluna 'volta\_rapida\_tempo' são normalizados para o intervalo entre 0 e 1 usando `MinMaxScaler` da biblioteca scikit-learn.**

**3. \*\*Preparação de dados para o modelo:\*\***

**- Os dados normalizados são transformados em conjuntos de treinamento (`x\_train` e `y\_train`) para o modelo de previsão.**

**- As sequências de 40 pontos de dados são usadas como entradas (`x\_train`), e o próximo ponto de dados é usado como saída (`y\_train`).**

**4. \*\*Construção do modelo LSTM:\*\***

**- Um modelo de rede neural recorrente (LSTM) é construído usando a biblioteca Keras. O modelo possui duas camadas LSTM e uma camada densa (fully connected).**

**- O modelo é compilado usando o otimizador 'adam' e a função de perda 'mean\_squared\_error'.**

**5. \*\*Treinamento do modelo:\*\***

**- O modelo é treinado usando os dados de treinamento (`x\_train` e `y\_train`) por 50 épocas com um tamanho de lote de 8.**

**6. \*\*Previsão de tempos de volta futuros:\*\***

**- São feitas previsões para os próximos 10 eventos de corrida usando o modelo treinado.**

**- As previsões são desnormalizadas para obter os valores reais de tempo de volta.**

**7. \*\*Visualização dos resultados:\*\***

**- Um gráfico é gerado para visualizar o histórico de melhores tempos de volta e as previsões para as próximas 10 corridas.**

**- Uma banda de confiança é adicionada em torno das previsões, indicando a incerteza.**

**8. \*\*Adição de barras de erro:\*\***

**- Marcadores de erro são adicionados às previsões para representar a banda de confiança.**

**- O tipo de marcador de erro é ajustado usando `plt.errorbar`.**

**9. \*\*Ajustes estéticos e legenda:\*\***

**- O título, rótulos dos eixos, grade e legenda são adicionados ao gráfico para torná-lo informativo.**

**\*\*Objetivo do Código:\*\***

**- O código tem como objetivo criar um modelo LSTM para prever os melhores tempos de volta em corridas de um determinado circuito.**

**- Além da previsão, o código visualiza as previsões futuras, destacando uma banda de confiança ao redor das previsões para indicar a incerteza associada.**

1. **Prevê a proxima o melhor tempo em um certo circuito e certo piloto(em ms)**

**Este código realiza previsões de tempos de volta para o próximo evento de corrida de um piloto específico (identificado pelo `piloto\_id`). Aqui estão as etapas principais:**

**1. \*\*Filtragem e Mesclagem de Dados:\*\***

- Filtra os resultados para o piloto específico e mescla esses resultados com os dados das corridas com base no ID da corrida e no circuito (ID 14).

**2. \*\*Normalização dos Dados:\*\***

- Normaliza os tempos de volta para estarem no intervalo entre 0 e 1 usando Min-Max Scaling.

**3. \*\*Preparação dos Dados para o Modelo:\*\***

- Verifica se há dados suficientes para treinar o modelo.

- Define o número de amostras de treinamento (`time\_steps`).

- Cria sequências de entrada (`x\_train`) e saída (`y\_train`) para o modelo LSTM.

**4. \*\*Criação e Treinamento do Modelo RNN (LSTM):\*\***

- Cria uma rede neural sequencial com duas camadas LSTM e uma camada densa para a saída.

- Treina o modelo usando os dados de treinamento com um otimizador 'adam' e a função de perda 'mean\_squared\_error'.

**5. \*\*Previsão para a Próxima Corrida:\*\***

- Verifica se há dados suficientes para prever a próxima corrida.

- Se sim, faz a previsão do tempo de volta para a próxima corrida usando o modelo treinado.

- Desescala a previsão para a escala original.

**6. \*\*Plotagem e Exibição de Resultados:\*\***

- Gera um gráfico mostrando o histórico de tempos de volta, destaca a previsão para a próxima corrida e exibe a porcentagem de erro em relação ao tempo real da última corrida.

**7. \*\*Avaliação da Precisão:\*\***

- Calcula a porcentagem de erro entre a previsão e o tempo real da última corrida.

**O código fornece uma visualização gráfica das previsões de tempos de volta para a próxima corrida do piloto, incluindo uma avaliação da precisão do modelo em relação aos dados históricos.**