Semana 05 – Exercício Semanal

Relatório do Modelo Preditivo - Titanic

PRO6006 - Métodos de Otimização Não Linear com aplicações em aprendizado de máquina

Nathan Sampaio Santos – 8988661

Este relatório detalha o processo de criação de um modelo de machine learning para prever a sobrevivência de passageiros do Titanic, com base no código fornecido. O objetivo final é gerar um arquivo submission.csv para a competição do Kaggle.

1. PRÉ-PROCESSAMENTO DOS DADOS

A etapa mais crítica do projeto é a preparação dos dados. Para garantir consistência e reusabilidade, foi criada uma função central chamada treat_data.

1.1. A Função treat_data

Esta função é responsável por receber um DataFrame bruto e executar as seguintes transformações:

- Imputação de Valores Ausentes: Valores nulos (NaN) nas colunas Age (Idade) e Fare (Tarifa) são preenchidos com a média da respectiva coluna. Isso evita a perda de dados.

```
treated_df['Age'] = treated_df['Age'].fillna(treated_df['Age'].mean())
treated_df['Fare'] = treated_df['Fare'].fillna(treated_df['Fare'].mean())
```

- Seleção de Colunas: Apenas as colunas relevantes para o treinamento (Pclass, Sex, Age, etc.) são mantidas.
- Codificação de Variáveis Categóricas: As colunas com texto (Sex, Pclass, Embarked) são convertidas em formato numérico usando a técnica de *one-hot encoding*. Isso é essencial, pois os modelos de redes neurais só aceitam números como entrada.

```
treated_df = pd.get_dummies(treated_df, columns=categorical_columns,
drop_first=True, dtype=int)
```

A função também possui um parâmetro trainning, que de forma inteligente adapta o tratamento para os dados de treino (que contêm a coluna Survived) e os de teste (que não a contêm).

2. CONSTRUÇÃO E TREINAMENTO DA REDE NEURAL

Com os dados limpos, o próximo passo é construir e treinar o modelo.

2.1. Padronização dos Dados (Scaling)

Antes de alimentar a rede neural, os dados são padronizados com o StandardScaler. Isso coloca todas as *features* na mesma escala (média 0 e desvio padrão 1), o que ajuda o modelo a treinar de forma mais rápida e estável.

Ponto Crítico: O scaler é treinado apenas com os dados de treino (*scaler.fit_transform(X_train)*) e depois é usado para apenas transformar os dados de teste (*scaler.transform(X_test)*). Isso evita o vazamento de informações do conjunto de teste para o treinamento.

2.2. Arquitetura do Modelo

Foi construída uma rede neural sequencial usando Keras. A arquitetura é a seguinte:

- Camada de Entrada (keras.Input): Define que o modelo receberá vetores com um número de features igual ao de expanded_training_columns.
- Camadas Ocultas (Dense): Quatro camadas ocultas com 18, 36, 18 e 9 neurônios, respectivamente. Elas são responsáveis por aprender os padrões complexos nos dados.
- Camada de Saída (Dense): Uma única camada com 1 neurônio e ativação sigmoid. A função sigmoid gera uma probabilidade entre 0 e 1, ideal para classificação binária (0 para "Não Sobreviveu" e 1 para "Sobreviveu").

2.3. Compilação e Treinamento

O modelo é compilado com:

- Otimizador adam: Um algoritmo eficiente para ajustar os pesos da rede.
- Função de Perda binary_crossentropy: A métrica padrão para medir o erro em problemas de classificação binária.

O treinamento é realizado por 50 épocas (epochs=50), onde o modelo analisa o conjunto de dados 50 vezes para aprender os padrões.

3. PREDIÇÃO E GERAÇÃO DO ARQUIVO DE SUBMISSÃO

Após o treinamento, o modelo está pronto para fazer previsões nos dados de teste.

1. Carregamento e Tratamento dos Dados de Teste: O arquivo test.csv é carregado e passa pela mesma função treat_data e pelo mesmo scaler que foram usados no treino. Isso garante total consistência.
2. Predição: O método model.predict() gera as probabilidades de sobrevivência.

3. Formatação Final: As probabilidades são convertidas para 0 ou 1. Um novo DataFrame é criado contendo o PassengerId e a predição final Survived. Este DataFrame é então salvo como submission.csv.

```
output = pd.DataFrame({'PassengerId': test_df.PassengerId, 'Survived':
predictions.ravel()})
output.to_csv('submission.csv', index=False)
```

Este processo garante um pipeline robusto e completo, desde a leitura dos dados brutos até a criação do arquivo final para a competição.

4. SUBMISSÃO NO KAGGLE