SCC0270 - Neural Networks and Deep Learning
Primeiro Projeto Prático

Esse projeto tem como objetivo fazer um estudo de uma base de fraudes em compras com cartões de crédito e tentar criar um modelo que ajude a mitigar tal problema. Os dados de treinamento estão disponíveis no arquivo **`train.csv`** e os de teste no arquivo **`test.csv`**.

Utilize a biblioteca scikit-learn para criar os modelos pedidos. Você pode usar outras bibliotecas para fazer as demais análises =)

Todas as respostas devem ser justificadas com base em:

1. **Código Python mostrando a(s) análise(s) e/ou o(s) modelo(s) feitos;**
2. **O resultado da(s) análise(s) e/ou do(s) modelo(s) e**
3. **Uma explicação textual (pode ser breve) da conclusão obtida.**

Em caso de plágio (mesmo que parcial) o trabalho de todos os alunos envolvidos receberá nota ZERO.

NÃO enviar um link para o Google Colab como resposta/relatório do projeto.

Desorganização excessiva do código resultará em redução da nota do projeto. Exemplos:

- **Códigos que devem ser rodados de forma não sequencial;**
- **Projeto entregue em vários arquivos sem um README;**
- **...**

Bom projeto,
Tiago.

Questão 1 (valor 2 pontos)

a) Baseado na base de dados fornecida, qual das duas métricas de avaliação deve ser usada para medir os resultados dos modelos: Acurácia ou AUC?
b) Na base fornecida, qual seria o resultado esperado de: um modelo aleatório, um modelo que diz que tudo é fraude e um modelo que diz que nada é fraude, para cada uma das duas métricas (Acurácia e AUC)? Discuta os resultados.

Questão 2 (valor 2 pontos) - A total corretude dessa questão depende da primeira questão

Usando a classe **`MLPClassifier`** do scikit-learn crie a seguinte arquitetura de rede neural:

- 5 camadas escondidas, com 20 neurônios em cada camada;
- Função de ativação **`ReLU`** para todos os neurônios das camadas escondidas;
- **`random_state = 42`** e
- Os demais parâmetros com o valor padrão da classe **`MLPClassifier`**.

Em seguida, treine um modelo para cada um dos valores de alpha (taxa de regularização L2): `[0.0, 0.01, 0.1, 1, 10]` e compute o resultado deles na base de teste (usando a métrica escolhida na questão anterior).

Com base nesses resultados, comente sobre o efeito da taxa de regularização no resultado do modelo.

Questão 3 (valor 2 pontos) - A total corretude dessa questão depende da primeira questão

Compare os seguintes modelos:

1. Rede neural (**`MLPClassifier`**):
    a. Sem camadas intermediárias;
    b. Com uma camada intermediária de 10 neurônios e
    c. Com duas camadas intermediárias, com 5 neurônios cada.

2. K-NN (**`KNeighborsClassifier`**)
    a. Com k = 3;
    b. Com k = 5 e
    c. Com k = 7.

Todos os demais hiperparâmetros devem ser deixados com os valores padrão do scikit-learn. Essa comparação deve ser feita usando 3-fold cross-validation nos dados de treinamento. Por fim, o valor da métrica de sucesso na base de teste só deve ser computado para o melhor modelo de rede neural e o melhor K-NN. Tal protocolo deve ser seguido para que tenhamos uma estimativa mais robusta dos resultados dos modelos.

Com base nesses resultados, responda:

a) Qual técnica obteve o melhor resultado na base de teste?
b) Qual técnica demora mais para gerar as previsões nos dados de teste?

Dicas:

1. Use a classe **`GridSearchCV`** do scikit-learn para comparar os hiperparâmetros e
2. Use o magic command **`%%time`** do Jupyter Notebook para medir os tempos.

Questão 4 (valor 2 pontos)

Usando o melhor modelo obtido na questão anterior, calcule o lucro que tal modelo traria no seguinte cenário:

- As top-1% transações com maior chance de fraude (de acordo com os scores do modelo) seriam impedidas de acontecer;
- Cada fraude evitada em média evita um prejuízo (gera um lucro) de R$ 100 e
- Cada não-fraude bloqueada gera em média um prejuízo de 2 reais.

Comente os resultados.

Questão 5 (valor 2 pontos)

Use uma rede neural sem camadas intermediárias e com um neurônio de saída para encontrar variáveis não importantes para o problema estudado. Indique porque tais variáveis não são importantes.

Dica: use a classe **`SGDClassifier(loss='log', random_state=42)`** para criar uma rede com um só neurônio de saída que usa a função de ativação sigmoid. Modifique os outros hiperparâmetros conforme necessário.