Case Cientista de Dados Inter

1º Passo: Análise Prévia dos dados fornecidos

A partir de uma leitura superficial dos dados e com julgamentos preestabelecidos por experiências anteriores, podemos esperar que a persona que representa um cliente de alto valor apresentaria as seguintes características:

- Faixa etária de 30 a 60 anos
  - Um cliente que já atingiu a maturidade profissional e financeira
- Ambos os gêneros
- Gastos no cartão de crédito consistentemente alto
- Utiliza crédito ativamente mesmo que seu montante investido não seja muito elevado
- Valoriza produtos e serviços mais exclusivos (Possui Inter Gourmet ou Cela Inter)

Estes pontos ajudarão a guiar a criação do modelo e avaliar os primeiros resultados

2º Passo: Idealizar um modelo inicial que atenda o problema apresentado

Inicialmente, vamos criar um modelo de classificação mais simples que aprenda as características de um cliente de alto valor e seja capaz de identificar novos clientes com esse mesmo perfil

1º Objetivo: Classificar um cliente em duas categorias { Alto valor / Padrão

- Modelo de classificação permite uma abordagem mais direta

3º Passo: Pré-processamento

- Converter todas as informações em números e garantir que as variáveis estejam na mesma escala para um funcionamento correto do modelo
  - Descarte da variável cliente
  - Criação da variável 'segmento', que será o nosso resultado
  - Encoding de variáveis categóricas através do One-hot Encoder
  - Fazer a divisão inicial dos conjuntos de treino e teste como 80/20
  - Normalização de variáveis

O primeiro modelo foi mais simples, realizando todo o pré-processamento citado e uma regressão logística.

Dito isto, o resultado foi muito ruim, como esperado. Obtive uma acurácia de 75% que, na prática, não é útil, pois não consegue identificar os clientes de alto valor. O modelo aprendeu a chutar que todos os clientes são "Padrão", pelo desbalanceamento dos dados

4º Passo: Testar um modelo mais complexo e poderoso

Para a segunda tentativa, optei por utilizar um Random Forest, ainda sem fazer alterações na base de dados fora o pré-processamento necessário, com o objetivo de entender como um modelo mais robusto desempenharia com os mesmos dados e ter uma base melhor para meus próximos passos

O segundo modelo testado, utilizando Random Forest, teve uma acurácia menor mas se saiu melhor na classificação de clientes de alto valor. Enquanto o primeiro modelo acertou 0 de 10 nos testes, o segundo acertou 1 de 9. Com isso, decidi seguir com o segundo modelo para os próximos passos.

Outro ponto importante é que, o modelo com Random Forest descobriu que o fator mais importante é o dinheiro investido e o segundo mais importante é a idade.

5º passo: Implementar estratégias de manipulação dos dados e técnicas de treinamento

- Validação cruzada
  - Melhorar o treinamento do modelo, visto que no 1º ele apenas aprendeu a chutar padrão pelo desbalanceamento da base de dados
- Ajustar hiperparâmetros
  - Encontrar a melhor combinação utilizando GridSearchCV

Após a otimização do modelo por parte do GridSearchCV e a melhoria do treinamento do modelo por parte da validação cruzada, não houve melhoria significativa.

Como estas estratégias não surtiram efeito, podemos concluir que o gargalo do modelo não está em seu ajuste. Assim, o próximo passo é focar nos dados de entrada e trabalhar com o csv disponibilizado

6º passo: Manipulação dos dados de entrada

Aplicar a técnica SMOTE para lidar com o desbalanceamento do dataset

A acurácia do modelo ficou mais baixa, porém o modelo se tornou mais eficaz.

Até agora, foi o modelo que mais acertou a classificação de clientes de alto valor. A acurácia diminuiu pois, com a técnica utilizada no dataset, o modelo reconheceu alguns clientes padrões como alto valor, o que, para o contexto do case, é muito mais vantajoso do que ter uma lista menor com menos clientes de alto valor e nenhum padrão.

Como o desempenho ainda não agrada, irei testar outros modelos que conheço e que façam sentido com o problema proposto.

7º passo: Utilizar Gradient Boosting no lugar do Random Forest

Utilizar a biblioteca LightGBM para desenvolver o modelo no lugar do Random Forest do Scikit-Learn

Obtive os melhores resultados até o momento. O modelo passou a identificar corretamente 50% dos clientes de alto valor, enquanto mantém um bom controle sobre os erros (falsos positivos).

Recall foi de 10 para 50% e a precisão de 20 para 36%

8º passo: Procurar formas de otimizar o modelo atual

Testar estratégias de engenharia de feature e seleção de feature para melhorar os resultados do modelo

A criação de features possibilitou um salto muito grande no desempenho do modelo, chegando a 90% de recall e 100% de precisão.

As features razao_gasto_investimento e total_movimentacoes se mostraram essenciais para o bom funcionamento do modelo

Mas para o contexto do projeto, seria melhor um recall maior mesmo que custe um pouco de precisão do modelo, porque enviar promoções de clientes de alto valor para clientes padrão não é algo que prejudicaria tanto o projeto quanto deixar de enviar essas promoções para clientes que realmente são de alto valor.

Tentar aumentar o recall

- Otimizando o GridSearch diretamente para o recall da classe alto_valor
- Utilizar pesos de classe na busca de hiperparâmetros
- Testar ADASYN além de SMOTE
- Ajustar o limiar de decisão para priorizar o recall

Mesmo com as alterações o modelo continuou a apresentar um recall de 90% e a precisão de 100%, que são números bons para o projeto.

9º passo: Definir próximos passos do projeto

- Fazer uma análise mais profunda dos resultados antes de testar novas técnicas
- Realizar a validação do modelo em outros datasets

A engenharia de feature causou o vazamento de dados para o processo de treinamento do modelo, assim voltei para o modelo antes de introduzir o erro.

10º passo: Reestruturar a engenharia de feature para melhorar o modelo

Adicionei uma etapa de validação além do treinamento e teste

Criei features de proporção do gasto e investimento e um score de idade e investimento evitando um data leakage como ocorreu na primeira vez

Resultados indicaram overfitting

Ajustei o Grid de hiperparâmetros para combater o overfitting

Overfitting corrigido

Resultados melhoraram bastante, tanto na validação quanto no teste

Precisão de 90% e recall de 82% para os clientes de alto valor com um F1-score de 0,86 e métricas de classificação consistentes entre o conjunto de validação e teste.

Score AUC consistentemente altos também (entre 0,96 e 0,985), indicando um bom poder de separação entre as classes