Projeto Final de Curso – Eng. ML

<WILSON FALCÃO>
<https://github.com/wilsonfalcao/eng-machine-learn-entrega>

Agenda do Trabalho

O aluno deve preencher essa apresentação com os resultados da sua implementação do modelo. Os códigos devem ser disponibilizados em repositório próprio, público, para inspeção.

Essa apresentação é padronizada para que os alunos possam incluir os seus resultados, com figuras, tabelas e descrições sobre o projeto de curso. Os resultados aqui descritos serão confrontados com os códigos disponibilizados.

Roteiro

- Objetivo da modelagem
- Arquitetura da solução
  - Diagrama
  - Bibliotecas
  - Artefatos e Métricas
- Pipeline de processamento dos dados
  - Descrição dos dados
  - Análise Exploratória
  - Seleção base de teste
- Pipeline de Treinamento do Modelo
  - Validação Cruzada
  - Regressão Logística
  - Árvore de Decisão
  - Seleção, finalização e registro
- Aplicação do Modelo
  - Model as a Service localmente
  - Interface para aplicação na base de produção
  - Monitoramento do modelo

Objetivo da modelagem

Em homenagem ao jogador da NBA Kobe Bryant (falecido em 2020), foram disponibilizados os dados de 20 anos de arremessos, bem sucedidos ou não, e informações correlacionadas.

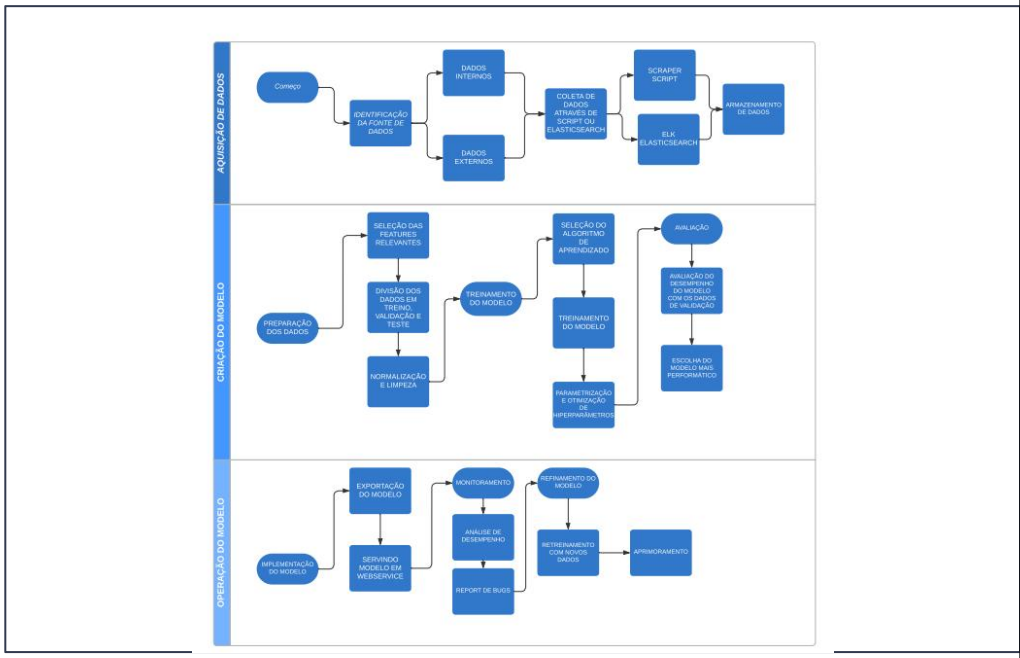
O objetivo desse estudo é aplicar técnicas de inteligência artificial para prever se um arremesso será convertido em pontos ou não.

Arquitetura da Solução

Arquitetura da Solução

Diagrama

< O aluno deve descrever os principais etapas de processamento da informação da solução, desde a carga da base de dados em treinamento até o serviço e consumo do modelo em produção. Descreva a importância de se implementar pipelines no desenvolvimento de modelos de ML>



Arquitetura da Solução

Bibliotecas

<Descreva como as ferramentas do curso (PyCaret, MLflow, Streamlit) elas podem ajudar nas atividades típicas de modelagem>

- Rastreamento de experimentos
- Treinamento e avaliação do modelo
- Monitoramento da saúde do modelo
- Atualização do modelo
- Provisionamento (deployment)

As ferramentas Streamlit, MLFlow, PyCaret e Scikit-Learn auxiliam na construção de pipelines de ML de diversas maneiras, abrangendo os aspectos mencionados:a. Rastreamento de Experimentos:+ MLFlow: Permite registrar e comparar parâmetros, métricas e artefatos de diferentes experimentos.+ PyCaret: Oferece um módulo de rastreamento que registra automaticamente métricas e parâmetros do modelo.+ Scikit-Learn: Suporta o módulo Yellowbrick para visualização de métricas e curvas de aprendizado.b. Funções de Treinamento:+ PyCaret: Automatiza a seleção de pré-processamento, modelo e hiperparâmetros, além de fornecer APIs para treinamento manual.+ Scikit-Learn: Oferece uma ampla variedade de algoritmos de ML e ferramentas para pré-processamento e avaliação de modelos.c. Monitoramento da Saúde do Modelo:+ MLFlow: Permite monitorar métricas de desempenho em tempo real e detectar anomalias.+ Streamlit: Integra-se com o MLFlow para visualização de métricas e dashboards interativos.d. Atualização de Modelo:+ MLFlow: Permite registrar diferentes versões do modelo e fazer rollback para versões anteriores.+ PyCaret: Oferece funções para reavaliar e atualizar modelos com novos dados.e. Provisionamento (Deployment):+ Streamlit: Permite a criação de interfaces web interativas para seus modelos.+ MLFlow: Oferece APIs para deploy de modelos em diferentes plataformas.+ PyCaret: Suporta o deploy de modelos em Flask, Heroku e Kubernetes.

Arquitetura da Solução

Artefatos e Métricas

<Enumere e descreva a função dos artefatos (plots, tabelas) e das métricas de desempenho que serão utilizados em desenvolvimento e produção>

- <shots_made_and_missed.png>: gráfico do qual é visto a amplitude da coluna target
- <Plot 2>: ……
- <dataset_kobe_prod.parquet>: Tabela de dados para tratamento e produção
- <metrics.csv>: ….

<continuação>

Processamento de Dados

Pipeline de processamento dos dados

Descrição dos dados

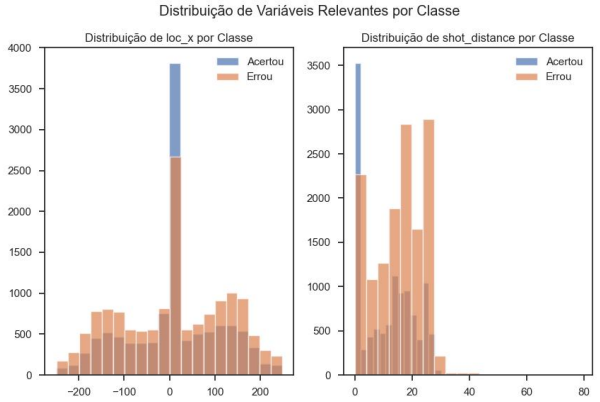
<Descreva o dataset, quantidade de linhas, colunas e dados faltantes. Para as colunas que serão utilizadas na modelagem, quais tipos de codificações de variáveis serão necessárias, quais são os valores das variáveis categóricas....>

O dataset Kobe Shot contém informações sobre as tentativas de arremesso e acertos de Kobe Bryant ao longo de sua carreira na NBA, com 24.073 linhas e 23 colunas. Existem poucos dados faltantes nas colunas de distância do arremesso e posição na quadra. Para a modelagem, é sugerida a codificação das variáveis categóricas e a normalização das variáveis numéricas. A análise dos dados faltantes pode revelar padrões, e diferentes técnicas de modelagem devem ser exploradas para encontrar a melhor solução. O dataset oferece oportunidades para análises mais complexas, como a performance de Kobe por temporada, equipe e adversário.

Pipeline de processamento dos dados

Análise Exploratória

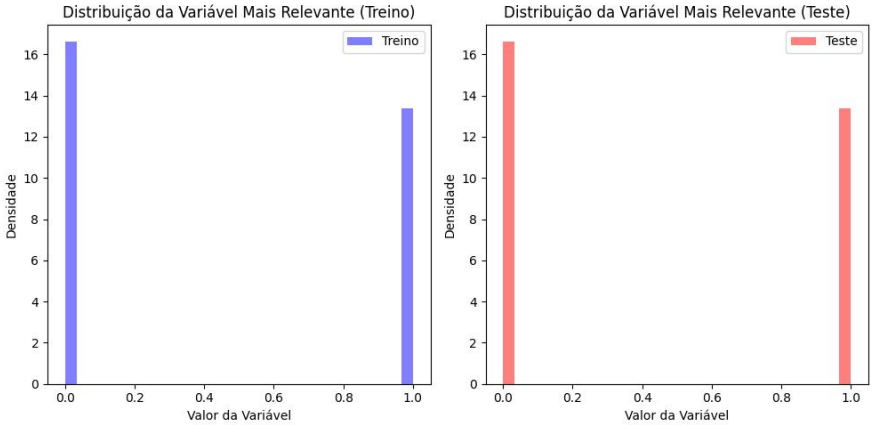
Variáveis:
loc_x: Posição X do arremesso na quadra
loc_y: Posição Y do arremesso na quadra
lat: Latitude do local do arremesso
lon: Longitude do local do arremesso
shot_distance: Distância do arremesso até a cesta
shot_made_flag: Indica se o arremesso foi convertido (1) ou não (0)
Tipo de dados:
loc_x, loc_y: Numéricas
lat, lon: Numéricas
shot_distance: Numérica
shot_made_flag: Categórica (binária)
Medidas de Tendência Central:
Podemos calcular a média, mediana e moda para cada variável numérica. Isso nos dará uma ideia de qual valor é mais frequente e como os dados se distribuem.
Medidas de Dispersão:
Podemos calcular o desvio padrão, variância e amplitude total para cada variável numérica. Isso nos ajudará a entender a variabilidade dos dados e identificar outliers.
Distribuição das Variáveis:
Podemos criar histogramas e gráficos de densidade de probabilidade para visualizar a distribuição das variáveis numéricas. Isso nos mostrará se os dados estão normalmente distribuídos ou se apresentam assimetria.
Relação entre as Variáveis:
Podemos criar gráficos de dispersão e matrices de correlação para analisar a relação entre as variáveis numéricas. Isso nos ajudará a identificar padrões e correlações entre as variáveis, que podem ser úteis para prever a classificação.



Pipeline de processamento dos dados

Seleção base de teste

Ao testar um modelo, é fundamental avaliar diversas métricas de desempenho, como precisão, recall, F1-score e AUC-ROC, para selecionar o melhor modelo. Isso envolve comparar o desempenho em diferentes algoritmos de aprendizado de máquina. Os dados são divididos em conjuntos de treinamento (80%) e teste (20%), e os modelos são treinados no conjunto de treinamento e avaliados no conjunto de teste para evitar overfitting. A validação cruzada é utilizada para garantir robustez nos resultados. Antes de testar, é importante verificar se as distribuições das variáveis relevantes são semelhantes nos conjuntos de treinamento e teste. Em suma, ao testar modelos, é crucial considerar métricas de avaliação, estratégias de treinamento e consistência das distribuições das variáveis



Treinamento do Modelo

Pipeline de Treinamento do Modelo

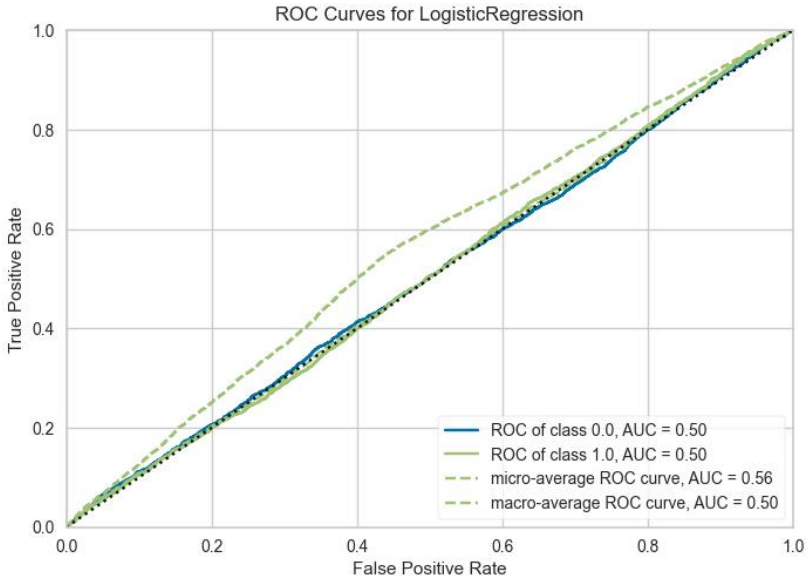
Regressão Logística - Validação Cruzada

Imagine um teste para garantir que seu modelo de IA não minta. Divida seus dados em partes, treine o modelo com todas menos uma e use essa parte para testar. Repita até que todas as partes sejam testadas. Essa é a validação cruzada!

Benefícios:

Menos Viés: Resultados mais confiáveis, sem enganar com dados específicos.
Melhor Escolha de Parâmetros: Encontre a configuração ideal para o seu modelo.
Curvas Reveladoras:

Curva de Validação: Mostre qual configuração de parâmetro faz o modelo brilhar.
Curva de Aprendizado: Evite que seu modelo se perca em dados específicos ou fique muito complexo.



Pipeline de processamento dos dados

Regressão Logística - Classificação

Precisão: Acertou em 55,61% das vezes, o que significa que para cada 100 previsões, 55,61 estavam corretas.

Curva ROC: A curva ROC indica que o modelo não consegue distinguir muito bem entre as classes (positivo e negativo).

Sensibilidade: O modelo identificou apenas 6,50% dos casos positivos reais. Ou seja, para cada 100 casos positivos reais, o modelo identificou apenas 6,50 como tal.

Precisão: Dos casos que o modelo classificou como positivos, 29,65% realmente eram positivos. Ou seja, para cada 100 casos que o modelo disse serem positivos, 29,65% realmente eram.

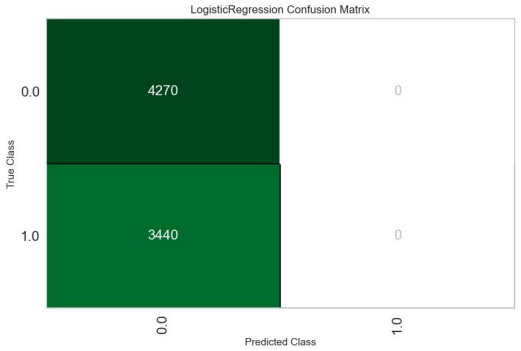
F1-Score: Uma medida que leva em conta a precisão e a sensibilidade, o F1-Score ficou em 8,89%.

Kappa: O coeficiente Kappa, que considera o acaso, ficou em 1,76%, indicando que o modelo teve um desempenho um pouco melhor que o acaso.

MCC: O coeficiente de correlação de Matthews (MCC), que também leva em conta as quatro células da matriz de confusão, ficou em 2,17%, indicando que o modelo teve um bom desempenho na classificação das classes.

Em resumo: O modelo de regressão logística teve um desempenho mediano, com precisão razoável, mas com baixa capacidade de distinguir entre as classes. Mais ajustes podem ser necessários para melhorar o desempenho do modelo

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.5542	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.5542	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.5570	0.0000	0.0237	0.5938	0.0455	0.0117	0.0399
3	0.5536	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.5503	0.0000	0.0224	0.4286	0.0426	-0.0019	-0.0055
5	0.5509	0.0000	0.0137	0.4074	0.0265	-0.0026	-0.0097
6	0.5670	0.0000	0.2914	0.5270	0.3753	0.0842	0.0929
7	0.5667	0.0000	0.2830	0.5267	0.3682	0.0819	0.0911
8	0.5534	0.0000	0.0162	0.4815	0.0314	0.0024	0.0088
9	0.5539	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Mean	0.5561	0.0000	0.0650	0.2965	0.0889	0.0176	0.0217
Std	0.0056	0.0000	0.1115	0.2470	0.1424	0.0330	0.0374



Pipeline de Treinamento do Modelo

Árvore de Decisão - Validação Cruzada

O modelo de árvore de decisão foi colocado à prova e os resultados revelam alguns pontos importantes:

Precisão: Acertou em 57,97% das vezes, o que significa que para cada 100 previsões, 57,97 estavam corretas.

Curva ROC: A curva ROC indica que o modelo não consegue diferenciar bem entre as classes (positivo e negativo).

Sensibilidade: O modelo identificou corretamente 53,48% dos casos positivos reais. Ou seja, para cada 100 casos positivos reais, o modelo identificou apenas 53,48 como tal.

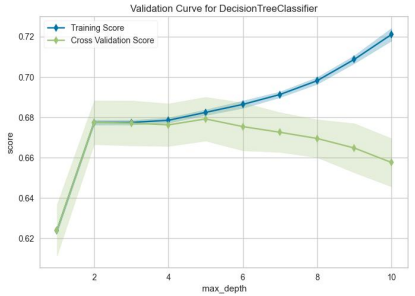
Precisão: Dos casos que o modelo classificou como positivos, 52,85% realmente eram positivos. Ou seja, para cada 100 casos que o modelo disse serem positivos, 52,85% realmente eram.

F1-Score: Uma medida que leva em conta a precisão e a sensibilidade, o F1-Score ficou em 53,15%.

Kappa: O coeficiente Kappa, que considera o acaso, ficou em 15,05%, indicando que o modelo teve um desempenho bem melhor que o acaso.

MCC: O coeficiente de correlação de Matthews (MCC), que também leva em conta as quatro células da matriz de confusão, ficou em 15,05%, similar ao Kappa, reforçando a boa performance do modelo na classificação das classes.

Em resumo: O modelo de árvore de decisão teve um bom desempenho, com precisão razoável e boa capacidade de distinguir entre as classes. Apesar da baixa capacidade de discriminação indicada pela AUC, os demais indicadores sugerem que o modelo é promissor.



Pipeline de Treinamento do Modelo

Árvore de Decisão - Classificação

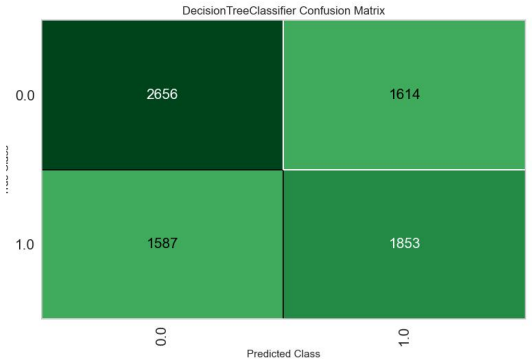
O modelo tem dificuldade em separar as classes "positivo" e "negativo". Isso significa que ele pode ter problemas para fazer previsões precisas

As métricas de desempenho, como precisão e recall, estão em um nível moderado, com uma acurácia média de 57,97%. Isso sugere que o modelo pode não ser confiável o suficiente para as previsões desejadas

O desempenho do modelo varia bastante entre os testes, o que significa que ele pode não funcionar da mesma forma em diferentes situações

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.5798	0.0000	0.5486	0.5276	0.5379	0.1528	0.1529
1	0.5803	0.0000	0.5549	0.5279	0.5410	0.1549	0.1551
2	0.5737	0.0000	0.5280	0.5222	0.5251	0.1383	0.1383
3	0.5981	0.0000	0.5791	0.5471	0.5626	0.1914	0.1917
4	0.5592	0.0000	0.4807	0.5066	0.4933	0.1037	0.1038
5	0.5609	0.0000	0.5243	0.5078	0.5159	0.1143	0.1143
6	0.5853	0.0000	0.5430	0.5350	0.5389	0.1622	0.1622
7	0.5723	0.0000	0.5025	0.5213	0.5117	0.1315	0.1315
8	0.5895	0.0000	0.5337	0.5404	0.5370	0.1684	0.1684
9	0.5979	0.0000	0.5526	0.5488	0.5512	0.1870	0.1870
Mean	0.5797	0.0000	0.5348	0.5285	0.5315	0.1505	0.1505
Std	0.0129	0.0000	0.0267	0.0139	0.0193	0.0274	0.0274

max_depth



Pipeline de Treinamento do Modelo

Seleção, finalização e registro (comparação com árvore de decisão)
O GBC acerta 67,91% das vezes, enquanto o modelo de Árvore de Decisão fica em 57,97%.

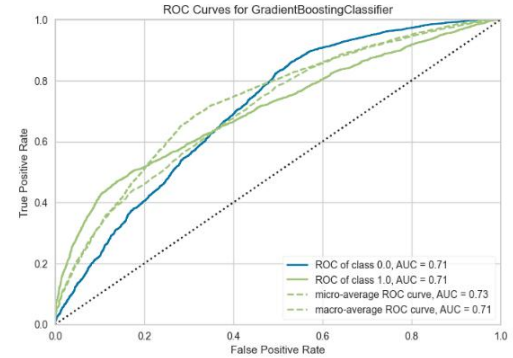
Ambos os modelos têm dificuldade em separar as classes. Mas isso não impede a gente de usar outras métricas para decidir.

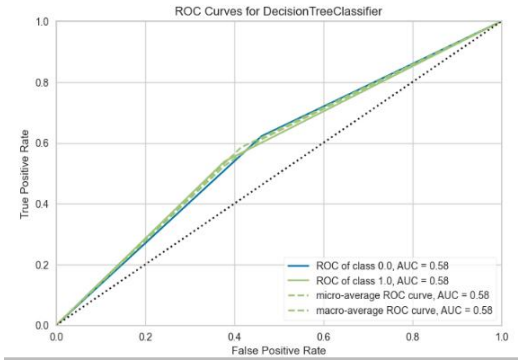
O modelo de Árvore de Decisão encontra um pouco mais os exemplos positivos (53,48% contra 46,06% do GBC)

O GBC se destaca prevendo corretamente 71,94% dos exemplos positivos, enquanto o modelo de Árvore de Decisão fica em 52,85%. Isso significa que o GBC é mais preciso no geral.

O GBC encontra um equilíbrio entre precisão e recall (56,13%) superior ao do modelo de Árvore de Decisão (53,15%)

As métricas Kappa e MCC indicam que o GBC tem uma concordância melhor entre suas previsões e os resultados reais, reforçando sua confiabilidade





Aplicação do Modelo

Pipeline de Aplicação do Modelo

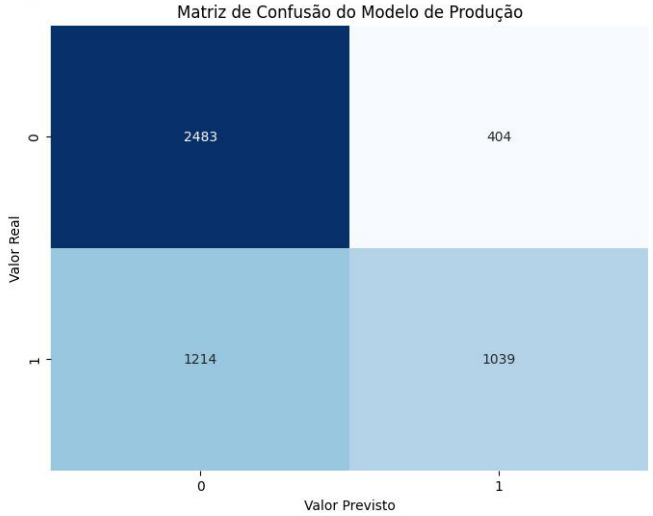
Deployment

```
print(model)
```

```
Pipeline(memory=Memory(location=None),
         steps=[('numerical_imputer',
                 TransformerWrapper(include=['game_event_id', 'game_id',
                                             'loc_x', 'loc_y', 'season',
                                             'seconds_remaining',
                                             'shot_distance', 'shot_type',
                                             'shot_zone_area',
                                             'shot_zone_basic', 'team_id',
                                             'team_name', 'matchup', 'opponent',
                                             'shot_id'],
                                    transformer=SimpleImputer())),
                ('categorical_imputer',
                 TransformerWrap...
                ('onehot_encoding',
                 TransformerWrapper(include=['combined_shot_type'],
                                    transformer=OneHotEncoder(cols=['combined_shot_type'],
                                                              handle_missing='return_nan',
                                                              use_cat_names=True))),
                ('rest_encoding',
                 TransformerWrapper(include=['action_type'],
                                    transformer=TargetEncoder(cols=[],
                                                              handle_missing='return_nan'))),
                ('trained_model',
                 GradientBoostingClassifier(random_state=3707))])
```

	precision	recall	f1-score	support
0.0	0.67	0.86	0.75	2887
1.0	0.72	0.46	0.56	2253
accuracy			0.69	5140
macro avg	0.70	0.66	0.66	5140
weighted avg	0.69	0.69	0.67	5140

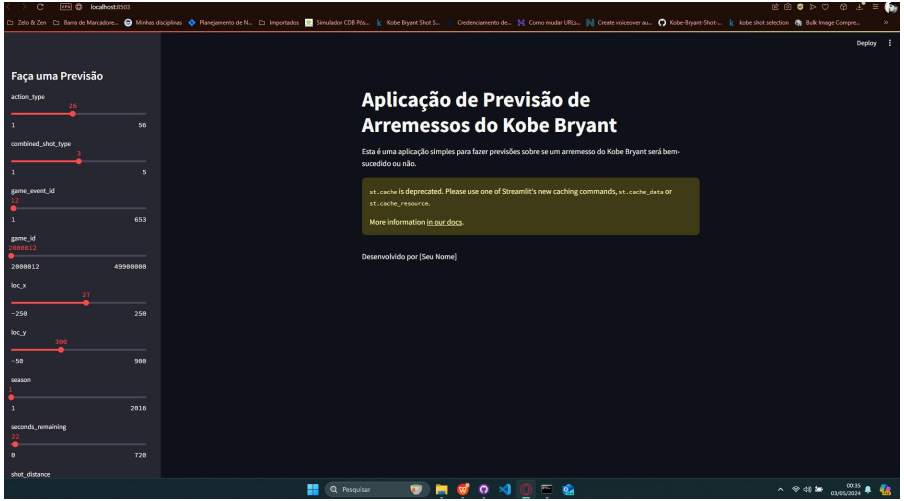
AUC: 0.7233598092125482



Pipeline de Aplicação do Modelo

Interface Monitoramento

O Streamlit é uma excelente lib para desenvolver modelos onde o projeto visa a escalabilidade e reuso para diversos tipos de teste.



Pipeline de Aplicação do Modelo

Retreinamento

<Descreva as estratégias reativa e preditiva de retreinamento para o modelo em operação>

Estratégia reativa:

Mudança na Distribuição dos Dados: Caso a distribuição dos dados na base de operação mude de forma significativamente, o modelo perde sua capacidade de generalização. Monitorar regularmente a distribuição dos dadose retrainando o modelo quando necessário.

Decaimento de Desempenho: Caso as métricas de desempenho do modelo diminuir, isso pode indicar que o modelo está perdendo sua eficácia devido a mudanças nos padrões dos dados ou em outros fatore. O retreinamento é uma boa alternativa.

Estratégia preditiva:

Retreinamento: O modelo pode ser re-treinado em intervalos regulares, independentemente de mudanças. Isso garante que o modelo esteja sempre atualizado.