

Rachel Reuters - Engenharia de Machine Learning [25E1_3]

Link pra o git: [posGraduacaoIA/EngenhariaML/pdblackmamba](https://github.com/posGraduacaoIA/EngenhariaML/pdblackmamba) at main · [rachelreuters/posGraduacaoIA](https://github.com/rachelreuters/posGraduacaoIA)

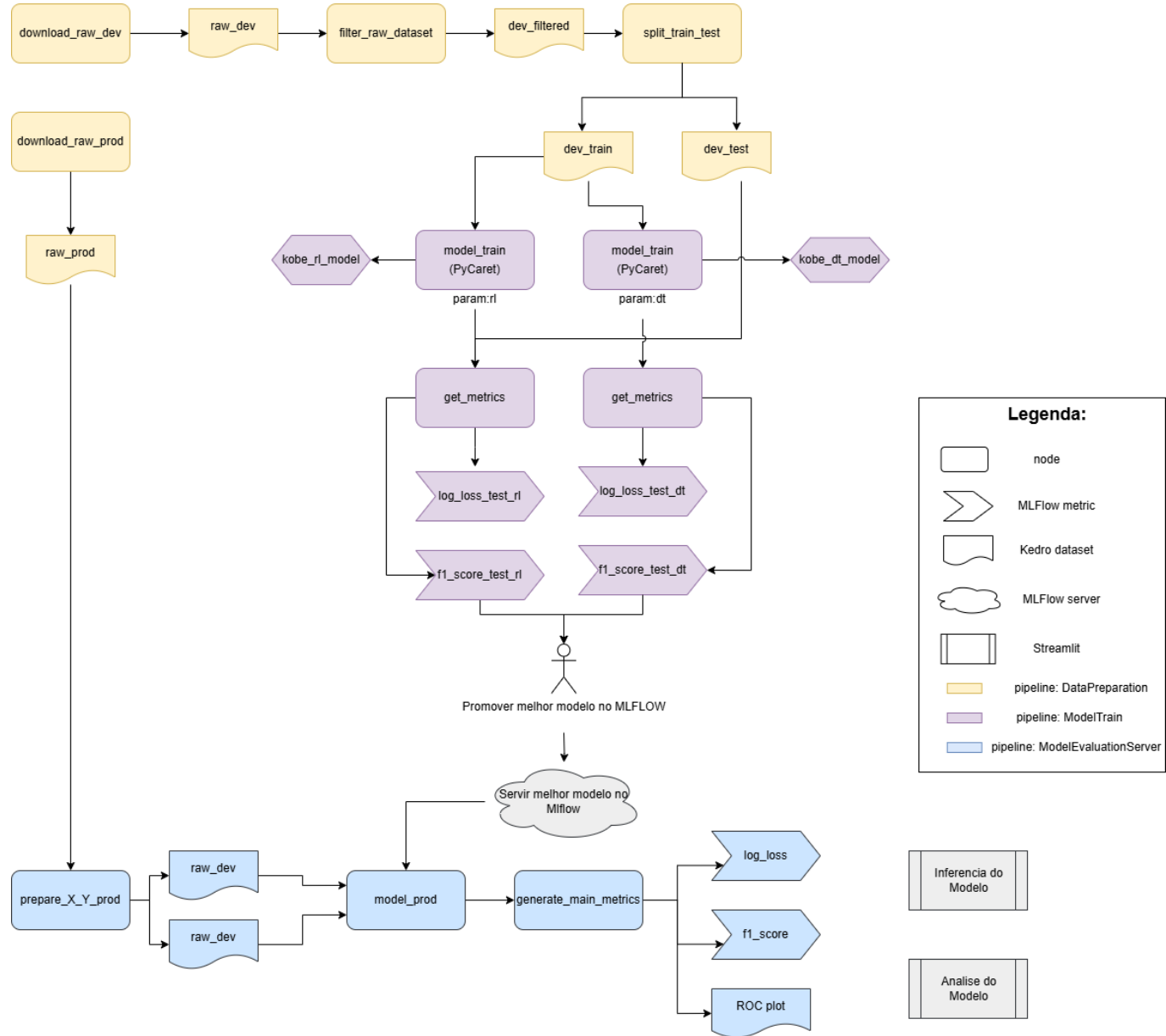
- [Introdução](#)
- [Diagrama](#)
- [Estrutura dos arquivos](#)
- [Descrição dos artefatos](#)
- [Respostas do Projeto de Disciplina e Rubricas explicadas :](#)
 - [PD:](#)
 - [Rubricas:](#)
- [HOW TO:](#)
 - [Requisitos:](#)
 - [Comandos:](#)

Introdução

Projeto de Engenharia de Machine Learning aplicado ao esporte, com foco no uso de modelos para prever resultados de partidas de basquete, mais especificamente para avaliar se o famoso Kobe Bryant converteu ou não a cesta. O projeto utiliza o framework **Kedro**, sendo estruturado em pipelines em que cada pipeline contem um grupo de nós. A fim de gerenciar o ciclo de vida dos modelos, está sendo utilizado o **MIFlow**, facilitando a organização e rastreamento dos experimentos, o que é crucial para melhorar a reprodutibilidade e escalabilidade dos projetos de Machine Learning. Para realizar um teste de inferência e também exibir um dashboard com as análises do modelo, foi utilizado o **Streamlit**. O repositório organiza os dados de forma estruturada, incluindo artefatos como datasets de treino e teste, e resultados da avaliação do modelo em produção. A abordagem inclui análises estatísticas detalhadas e geração de relatórios sobre a performance do modelo, visando otimizar as previsões no contexto de um servidor de produção.

Diagrama

Seguindo as nomenclaturas do Kedro (pipelines e nós):



Estrutua dos arquivos

```
.viz
conf
├── base
├── local
data
├── 01_raw
├── 02_intermediate
├── 03_primary
├── 04_feature
├── 05_model_input
├── 06_models
├── 07_model_output
├── 08_reporting
│   ├── data_input_analysis
│   ├── model_prod_report
│   └── model_train_report
docs
├── source
mlruns
notebooks
├── .ipynb_checkpoints
├── mlruns
│   ├── .trash
│   ├── 0
│   └── models
src
├── pdblackmamba
│   ├── pipelines
│   │   ├── DataPreparation
│   │   │   └── __pycache__
│   │   ├── ModelEvaluationPickle
│   │   │   └── __pycache__
│   │   ├── ModelEvaluationServer
│   │   │   └── __pycache__
│   │   ├── ModelTrain
│   │   │   └── __pycache__
│   │   └── __pycache__
│   └── __pycache__
streamlit
├── pages
```

Descrição dos artefatos

- 01_raw

- dataset_kobe_dev.parquet : Arquivo original para teste e treino do modelo
- dataset_kobe_prod.parquet : Arquivo original para avaliação do modelo em produção
- 03_primary
 - data_filtered.parquet: Dataset de dev filtrado para treino e teste do modelo, nesse dataset foram removidos os nulos e selecionado as que irão ser utilizadas como teste e treino. Colunas : 'lat','lon','minutes_remaining','period','playoffs','shot_distance', 'shot_made_flag'
- 05_model_input
 - base_test.parquet : Dataset gerado depois de separar 80/20 o dataset de dev filtrado para testar o modelo.
 - base_train.parquet: Dataset gerado depois de separar 80/20 o dataset de dev filtrado para treinar o modelo.
 - x_prod.pkl : Arquivo pickle que representa o dado de producao ja filtrado e que sera utilizado como input do modelo para avaliacao em producao.
 - y_prod.pkl: Arquivo pickle que representa o dado de producao ja filtrado e que sera utilizado para avaliar os resultados do modelo comparando com os outputs de predicao
- 07_model_output
 - predict_prod_server.parquet: Resultado das metricas do modelo depois da avaliacao dos dados de producao comparando a predicao com os Y original.
 - y_pred_prod.pkl: Resultado da predicao do modelo utilizando os dados de producao como entrada.
 - y_prob_prod.pkl: Resultado das probabilidades do modelo utilizando os dados de producao como entrada.
- 08_reporting
 - data_input_analysis
 - dev_test_correlation.png: Correlacao dos dados de input do grupo de teste do dataset de dev.
 - dev_test_distribution.png: Distribuicao dos dados de input do grupo de teste do dataset de dev.
 - dev_test_lat_lon_shot_original.png: Plot da latitude e longitude com o shot_made_flag (Y) do grupo de teste do dataset de dev.
 - dev_train_correlation.png: Correlacao dos dados de input do grupo de treino do dataset de dev.
 - dev_train_distribution.png: Distribuicao dos dados de input do grupo de treino do dataset de dev.
 - dev_train_lat_lon_shot_original.png: Plot da latitude e longitude com o shot_made_flag (Y) do grupo de treino do dataset de dev.
 - dev_train_test_balance.png: Balanceamento do Y (shot_made_flag) comparando os dados de trein e teste do dataset de dev.
 - model_prod_report
 - balance_Y_prod.png: Balanceamento do Y para o dataset de producao
 - lat_lon_shot_prod.png: Plot da latitude e longitude X modelo acertou ou errou
 - prod_metrics.png: Metricas do modelo com os dados de producao.
 - roc_curve_prod_server.png: ROC dos dados de producao aplicados no modelo da API
 - prod_data_distribution.png: Distribuicao dos dados de producao.
 - model_train_report

- dev_dt_lat_lon_shot_model_test.png : Plot da latitude e longitude X modelo acertou ou errou para o grupo de teste (modelo de arvore de decisao)
- dev_lr_lat_lon_shot_model_test.png : Plot da latitude e longitude X modelo acertou ou errou para o grupo de teste (modelo de regressao logistica)
- dev_model_feature_importance_dt.png : Importancia de features para o modelo de arvore de decisao
- dev_model_feature_importance_lr.png: Importancia de features para o modelo de regressao logistica
- dev_roc_dt.png: Curva roc para o modelo de arvore de decisao no grupo de teste.
- dev_roc_lr.png: Curva roc para o modelo de regressao logistica no grupo de teste.
- dt_metrics.png: Metricas finais para o modelo de arvore de decisao.
- lr_metrics.png: Metricas finais para o modelo de regressao logistica.
- Metricas no MLFLOW:
 - PipelineAplicacao
 - f1_score: f1_score no modelo para o dataset de producao
 - log_loss: log_loss no modelo para o dataset de producao
 - precision_score: Precisao no modelo para o dataset de producao
 - recall_score: Recall no modelo para o dataset de producao
 - Treinamento
 - f1_score_dt_test: f1 score para a arvore de decisao para o grupo de teste
 - f1_score_lr_test: f1 score para a regressao logistica para o grupo de teste
 - log_loss_dt_test: log_loss para a arvore de decisao para o grupo de teste
 - log_loss_lr_test: log_loss para a regressao logistica para o grupo de teste
 - PreparacaoDados
 - test_size: tamanho do dataset de teste depois de dividir em treino e teste (80/20)
 - train_size: tamanho do dataset de treino depois de dividir em treino e teste (80/20)
- Parametros no MLFLOW:
 - file_name_test_prefix
 - file_name_train_prefix
 - mlflow_experiment
 - model_dt
 - model_regLog
 - percent_test
- Pipelines (Kedro):
 - DataPreparation: Essa etapa é responsável por preparar os dados para serem usados para treinar o modelo. Retirar nulos, separar dados de treino e teste, gerar métricas relativas aos dados iniciais.
 - ModelEvaluation: Essa etapa é responsável por executar predição do modelo em dados de produção e com isso gerando métricas.
 - ModelTrain: Essa etapa é responsável por realizar treinos dos modelos, comparar modelos, gerar métricas, gerar plots dos relativos testes e treinos.
- Páginas do Streamlit:
 - main.py: Página inicial para redirecionar para as demais páginas.
 - Analise.py : Reune todos os resultados e análises dos modelos.
 - Inferencia.py: Input de dados de formulário para teste do modelo servido em produção pelo MLFlow.

Respostas do Projeto de Disciplina e Rubricas explicadas :

PD:

- Como as ferramentas Streamlit, MLFlow, PyCaret e Scikit-Learn auxiliam na construção dos pipelines descritos anteriormente?
Streamlit oferece uma interface web simples para visualizar resultados e parâmetros do modelo durante o desenvolvimento, além disso permite que usuario teste o modelo de forma intuitiva.
MLFlow é usado para rastrear experimentos, registrar modelos e gerenciar o versionamento, garantindo reprodutibilidade.
PyCaret automatiza etapas de machine learning, como pré-processamento e modelagem, permitindo construir e testar múltiplos modelos facilmente.
Scikit-Learn fornece os algoritmos fundamentais para treinamento de modelos e preparacao de dados, além disso ajuda na validação e avaliação de modelos.
- Explique como a escolha de treino e teste afetam o resultado do modelo final. Quais estratégias ajudam a minimizar os efeitos de viés de dados?
A separacao dos dados de teste e treino e fundamental e afeta diretamente a capacidade do modelo de generalizar para novos dados. Se os dados de treino não representam bem o problema, o modelo pode aprender de maneira enviesada ou limitada. Se o conjunto de teste não é representativo, a avaliação do modelo será imprecisa.
Dados de treino muito grandes e teste muito pequenos podem levar a overfitting, pois o modelo será testado em um conjunto limitado de dados. Por outro lado, treinar com poucos dados pode causar underfitting, já que o modelo não terá aprendido padrões suficientes. Um conjunto de treino ou teste mal balanceado (como uma classe com mais exemplos que outras) pode induzir viés nos resultados.
-Registre os parâmetros (% teste) e métricas (tamanho de cada base) no MIFlow
Estao registrados no item de pipeline *PreparacaoDados* do MLFLOW.

pdblackmamba >

PreparacaoDados

Overview

Model metrics

System metrics

Artifacts

Created at

2025-04-08 17:41:10

Created by

belch

Experiment ID

635427057558173878

Status

Finished

Run ID

8b24caec61c14e389ad3a7cdb2ac0caf

Duration

13.0s

Datasets used

—

Tags

kedro_command: kedro run --pipeline=data_prep...

kedro_version: 0.19.12

pipeline_name: data_preparation

project_path: C:/Users/belch/Documents/CursolA/...

runner: <kedro.runner.sequential_runner.Sequenti...

session_id: 2025-04-08T20:41:06.748Z

Source

C:\Users\belch\anaconda3\envs\kedro_py311\Scripts\kedro

Logged models

—

Registered models

—

Parameters (3)

Search parameters

Parameter	Value
file_name_test_prefix	test
file_name_train_prefix	train
percent_test	20

Metrics (2)

Search metrics

Metric	Value
test_size	4057
train_size	16228

- Experiments** pdblackmamba Provide Feedback Add Description Share

Search Experiments


pdblackmamba

Runs **Evaluation** **Experimental** **Traces** **Experimental**

Time created State: Active Datasets Sort: Created Columns Group by + New run

	Run Name	Created	Duration	F1	Log Loss	F1_score	F1_score_dt_test	F1_score_It_test	log_loss	log_loss_dt_test	log_loss_It_test	test_size	train_size	Tags	Source
<input type="checkbox"/>	PipelineAplicacao		6.7s	-	-	0.40145985...	-	-	10.1620337...	-	-	-	-	-	-
<input type="checkbox"/>	Treinamento_v		2.8s	-	-	-	-	-	-	-	-	-	-	setup	-
<input type="checkbox"/>	Logistic Regress...		410ms	0.5487	6.7691	-	-	-	-	-	-	-	-	tune_model	-
<input type="checkbox"/>	Logistic Regress...		451ms	0.5242	6.7294	-	-	-	-	-	-	-	-	create_model	-
<input type="checkbox"/>	Treinamento_dt		14.1s	-	-	-	-	-	-	-	-	-	-	setup	-
<input type="checkbox"/>	Decision Tree Cl...		420ms	0.4693	6.56	-	-	-	-	-	-	-	-	tune_model	-
<input type="checkbox"/>	Decision Tree Cl...		3.5s	0.5408	7.4498	-	-	-	-	-	-	-	-	create_model	-
<input type="checkbox"/>	treinamento		35.3s	-	-	-	0.34320867...	0.34320867...	-	8.25458946...	8.25458946...	-	-	-	-
<input type="checkbox"/>	PreparacaoDados		13.0s	-	-	-	-	-	-	-	-	4057	16228	-	-


Os modelos foram registrados no MLFLOW, e como a regressao logistica foi escolhida, esse modelo foi promovido para producao.



[Experiments](#)
[Models](#)

[GitHub](#)
[Docs](#)

Registered Models

Name 	Latest version	Aliased versions	Created by	Last modified	Tags
kobe_dt_model	Version 1			2025-04-04 18:51:10	—
kobe_lr_model	Version 1			2025-04-04 18:51:21	—
kobe_lr_model_prod	Version 1			2025-04-04 18:52:31	—

-Em seguida, para testar o modelo servido através da API, é possível executar a pipeline *ModelEvaluationServer* (o comando pode ser encontrado no setor HOW TO mais abaixo). Essa pipeline utiliza os dados de produção (`dataset_kobe_prod`), filtra as colunas e remove os nulos (assim como foi realizado no grupo de dados de dev) para testar o modelo servido através de API pelo Mlflow. Os resultados obtidos das métricas foram salvos tanto num arquivo parquet (`predict_prod_server.parquet`)

quanto registrados no MLFLOW, pelo item de pipeline do Mlflow *PipelineAplicacao*.

pdblackmanba

PipelineAplicacao

Overview

Model metrics

System metrics

Artifacts

Created by

belch

Experiment ID

635427057558173878

Status

Finished

Run ID

ec69cd692c4745e8847fa24c045f7a95

Duration

6.7s

Datasets used

—

Tags

kedro_command: kedro run --pipeline=model_eva...

kedro_version: 0.19.12

pipeline_name: model_evaluation_server

project_path: C:/Users/belch/Documents/CursolA/...

runner: <kedro.runner.sequential_runner.Sequential...

session_id: 2025-04-08T20:43:46.522Z

Source

C:\Users\belch\anaconda3\envs\kedro_py311\Scripts\kedro

Logged models

—

Registered models

—

Parameters (0)

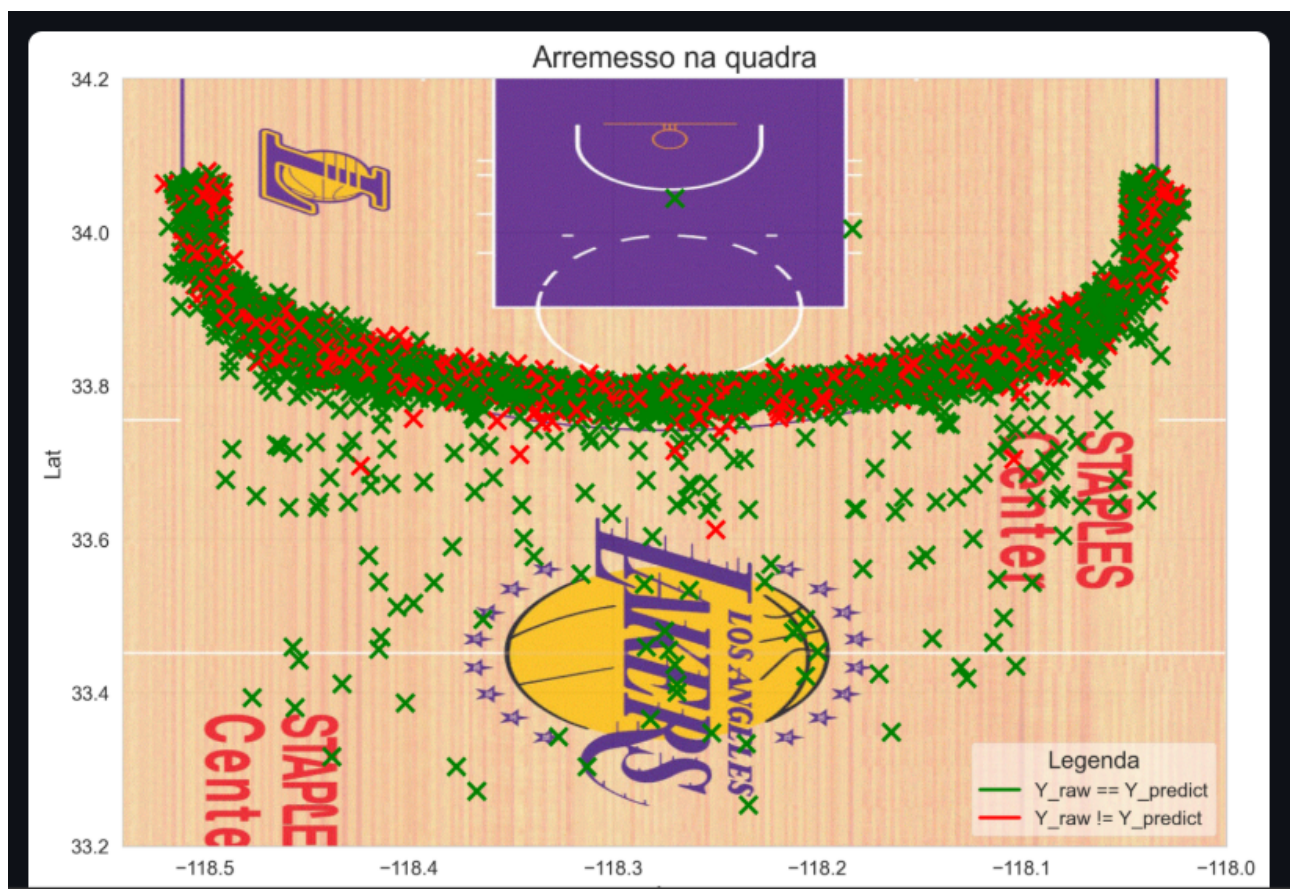
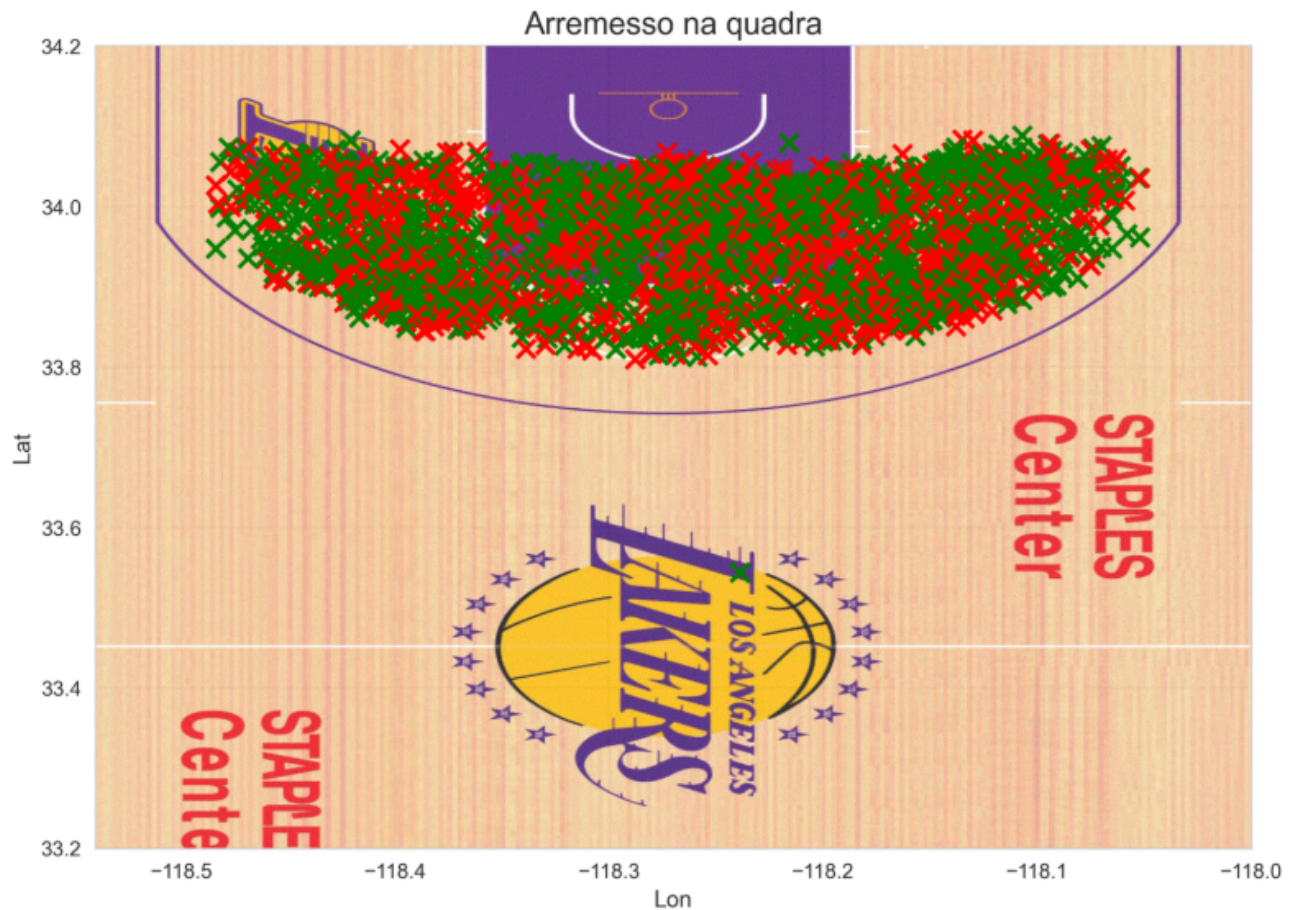
No parameters recorded

Metrics (4)

Search metrics

Metric	Value
f1_score	0.40145985401459855
log_loss	10.162803724645473
precision_score	0.3353658536585366
recall_score	0.5

- O modelo é aderente a essa nova base? O que mudou entre uma base e outra? Justifique.
Podemos ver que, em produção, a maioria dos arremessos tem uma distância entre 20 e 30, e esses valores quase não são observados no dataset de treino do modelo. Outra coisa é sobre a latitude, que teve maior grau de importância no treinamento do modelo. Enquanto que, nos dados de treino, os valores se concentraram entre 34 e 34.1, os de produção estão concentrados na faixa de 33.8, o que pode ter ocasionado a previsão de todos os dados como sendo 0 (errou a cesta).
As imagens a seguir mostram os arremessos de dev e de prod, e podemos observar essa divergência citada acima. (essas análises podem ser vistas no Streamlit).



- Descreva como podemos monitorar a saúde do modelo no cenário com e sem a disponibilidade da variável resposta para o modelo em operação.
Com a disponibilidade da variável de resposta, pode utilizar as métricas mais comuns como F1, precisão, recall (dependendo da figura de mérito do problema em questão). Além disso tem a curva ROC, e matriz de confusão que também devem apoiar no monitoramento.

Sem a disponibilidade da variável de resposta, o foco maior seria em observar distribuição dos dados, identificação de outliers, e balanceamento das classes.

Em ambos os casos é importante observar o padrão de comportamento dos resultados.

- Descreva as estratégias reativa e preditiva de retreinamento para o modelo em operação.
Na abordagem reativa, o modelo é retreinado após detectar problemas em desempenho. É uma estratégia baseada em eventos e depende de sinais específicos que indicam a necessidade de ajuste. Ocorre quando mudanças significativas nos dados são detectadas, como data drift ou concept drift. As vantagens de um modelo reativo é a simplicidade de implementação e otimização de recursos (já que treina o modelo apenas na necessidade), a principal desvantagem é o atraso entre a degradação e sua correção, podendo gerar problemas para os usuários ou acidentes (dependendo do problema). Na estratégia preditiva, o modelo é retreinado de forma programada com base nas análises preditivas que identificam quando o desempenho pode deteriorar (essas análises podem ser por exemplo padrões históricos de performance do modelo). Vantagens desse método incluem períodos de baixa performance e a garantia de maior estabilidade no ambiente produtivo. Como desvantagem ocorre um maior custo computacional e também requer um sistema robusto de previsão para determinar intervalos de retreinamento eficientes.

- Streamlit

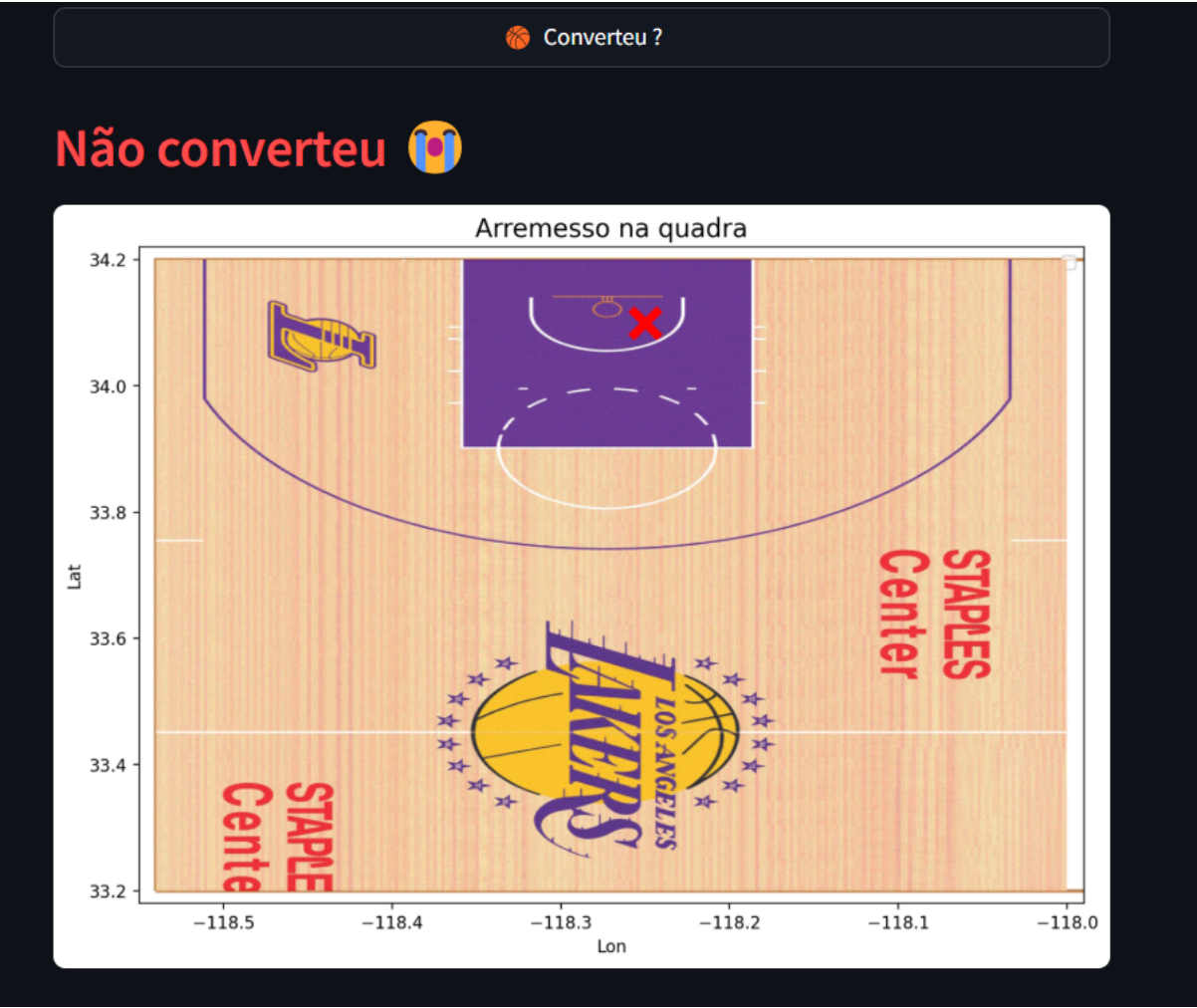
- Inferência

Nessa tela o usuário pode realizar um teste utilizando o modelo em produção.

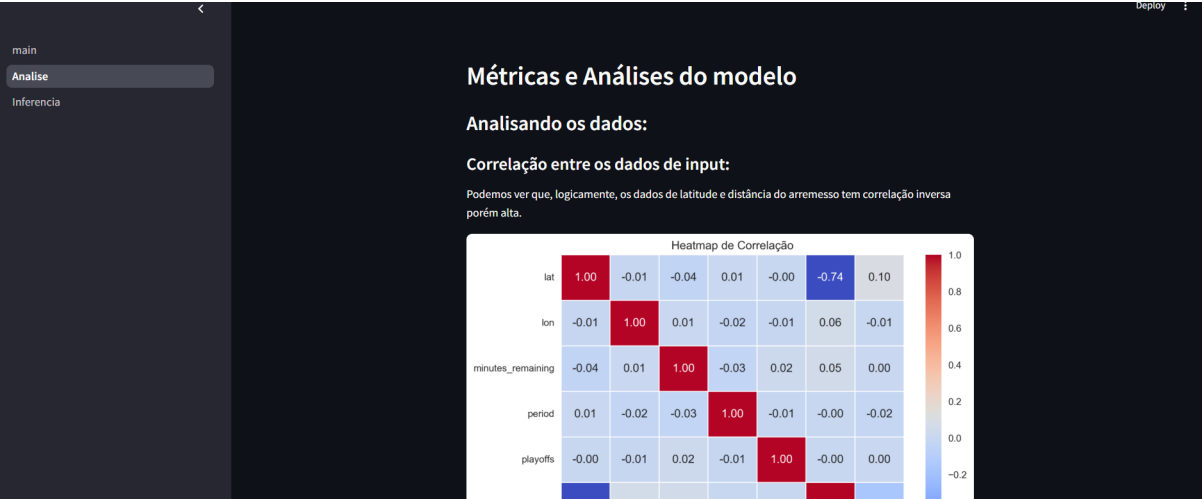
The screenshot shows a web application interface with a dark theme. At the top, the title 'BLACK MAMBA Converteu a cesta???' is displayed in large, bold, white letters. Below the title, a subtitle 'Preencha o formulário e descubra!' is shown in a slightly smaller font. The form consists of several input fields arranged in two columns. The left column contains three numeric input fields with labels: 'Posição do arremesso de latitude da quadra' (value: 33,2000), 'Posição do arremesso de longitude da quadra' (value: -118,5400), and 'Minutos faltantes para fechar o quarto' (value: 0). The right column contains two inputs: a numeric field for 'Quarto no momento do arremesso' (value: 1) and a checkbox labeled 'Etapa de playoffs?'. Below the checkbox is another numeric field for 'Distância da cesta em pés' (value: 0). At the bottom of the form, there is a large button with a basketball icon and the text 'Converteu ?'.

O resultado da inferência do modelo pode ser observado logo abaixo, após o usuário clicar no

botao de "Converteu?"



- Análise
Nessa tela pode ser visto varios graficos e analises dos dados, do treino do modelo, dos testes e as métricas.



Rubricas:

1.1 O aluno categorizou corretamente os dados?

R: Tanto os dados de dev quanto de prod foram filtrados com as colunas (features) citadas no pd , além disso os dados nulos foram devidamente removidos (como pede no pd). Além disso, para treino e teste do modelo, foi separado em 80, 20 os grupos de treino e teste de forma estratificada.

1.2 O aluno integrou a leitura dos dados corretamente à sua solução?

R: Sim, o primeiro passo do pipeline de DataPreparation é realizar o download dos dados do github conforme link citado no PD.

1.3 O aluno aplicou o modelo em produção (servindo como API ou como solução embarcada)?

R: Sim, através do MLFLOW com o comando `mlflow models serve ^ -m models:/kobe_lr_model_prod/latest ^ --env-manager=local ^ --port 5001`

1.4 O aluno indicou se o modelo é aderente a nova base de dados?

R: O modelo é aderente, porém não é eficiente, já que principalmente com relação a latitude, que é a feature com maior grau de importancia para o modelo, teve padrão diferente do dataset de treino e de teste do modelo. O próximo passo para melhoria desse modelo em produção poderia ser retreinar novamente o modelo utilizando um conjunto de dados mais abrangente.

2.1 O aluno criou um repositório git com a estrutura de projeto baseado no Framework TDSP da Microsoft?

R: Utilizei o Kedro conforme instruído na aula.

2.2 O aluno criou um diagrama que mostra todas as etapas necessárias para a criação de modelos?

R: Sim, verificar Diagrama no início desse documento.

2.3 O aluno treinou um modelo de regressão usando PyCaret e MLflow?

R: Sim, é possível observar essa lógica no nó *model_train* no pipeline de nome *ModelTrain*. (o parametro para esse nó para treinar com árvore de decisão é o *regLog*)

2.4 O aluno calculou o Log Loss para o modelo de regressão e registrou no mlflow?

R: Sim, é possível observar essa lógica no nó *get_metrics* no pipeline de nome *ModelTrain*.

pdblackmamba

Treinamento

OverviewModel metricsSystem metricsArtifacts

Created at

2025-04-08 17:41:33

Created by

belch

Experiment ID

635427057558173878

Status

Finished

Run ID

6f736d8d7c6c4d29a441f7a082ff2635

Duration

35.3s

Datasets used

—

Tags

kedro_command: kedro run --pipeline=model_trainkedro_version: 0.19.12pipeline_name: model_train

project_path: C:/Users/belch/Documents/CursorA/...runner: <kedro.runner.sequential_runner.Sequential...

session_id: 2025-04-08T20:41:35.225Z

Source

C:\Users\belch\anaconda3\envs\kedro_py311\Scripts\kedro

Logged models

sklearn

Registered models

kobe_lr_model_prod v1 +2

Parameters (3)

Search parameters

Parameter	Value
mlflow_experiment	pdblackmamba
model_dt	dt
model_regLog	lr

Metrics (4)

Search metrics

Metric	Value
f1_score_dt_test	0.3432086773514651
f1_score_lr_test	0.3432086773514651
log_loss_dt_test	8.254589461015943
log_loss_lr_test	8.254589461015943

2.5 O aluno treinou um modelo de árvore de decisao usando PyCaret e MLflow?

R: Sim, é possível observar essa lógica no nó `model_train` no pipeline de nome `ModelTrain`. (o parametro para esse nó para treinar com árvore de decisão é o `dt`)

2.6 O aluno calculou o Log Loss e F1 Score para o modelo de árvore de decisão e registrou no mlflow?

R: Sim, é possível observar essa lógica no nó `get_metrics` no pipeline de nome `ModelTrain`.

pdblackmamba

Treinamento

OverviewModel metricsSystem metricsArtifacts

Created at

2025-04-08 17:41:33

Created by

belch

Experiment ID

635427057558173878

Status

Finished

Run ID

6f736d8d7c6c4d29a441f7a082ff2635

Duration

35.3s

Datasets used

—

Tags

kedro_command: kedro run --pipeline=model_trainkedro_version: 0.19.12pipeline_name: model_train

project_path: C:/Users/belch/Documents/CursorA/...runner: <kedro.runner.sequential_runner.Sequential...

session_id: 2025-04-08T20:41:35.225Z

Source

C:\Users\belch\anaconda3\envs\kedro_py311\Scripts\kedro

Logged models

sklearn

Registered models

kobe_lr_model_prod v1 +2

Parameters (3)

Search parameters

Parameter	Value
mlflow_experiment	pdblackmamba
model_dt	dt
model_regLog	lr

Metrics (4)

Search metrics

Metric	Value
f1_score_dt_test	0.3432086773514651
f1_score_lr_test	0.3432086773514651
log_loss_dt_test	8.254589461015943
log_loss_lr_test	8.254589461015943

3.1 O aluno indicou o objetivo e descreveu detalhadamente cada artefato criado no projeto?

R: No tópico Descrição dos artefatos presente nesse documento.

3.2 O aluno cobriu todos os artefatos do diagrama proposto?

R: No tópico Descrição dos artefatos presente nesse documento.

3.3 O aluno usou o MLFlow para registrar a rodada "Preparação de Dados" com as métricas e argumentos relevantes?

R: Salvei as metricas e parametros da preparação de dados e de treino do modelo na run do Mlflow com nome *PreparacaoDados*.

3.4 O aluno removeu os dados faltantes da base?

R: Foram removidos os dados faltantes no nó *filter_raw_dataset*.

3.5 O aluno selecionou as colunas indicadas para criar o modelo?

R: Foram selecionadas as colunas no nó *filter_raw_dataset*.

3.6 O aluno indicou quais as dimensões para a base preprocessada?

R: As dimensoes da base de treino e teste (*test_size* e *train_size*) foram salvas como métrica no MLFLOW na run de nome *PreparacaoDados*.

3.7 O aluno criou arquivos para cada fase do processamento e os armazenou nas pastas indicadas?

R: Download inicial foi colocado na pasta 01_raw. Já os dados filtrados (remoção de nulos e seleção de colunas) está sendo salvo na pasta 03_primary. Após a separação de dados de treino e teste, esses arquivos estão sendo salvos no 05_model_input.

3.8 O aluno separou em duas bases, uma para treino e outra para teste?

R: Grupos de treino e teste estão sendo separados no nó *split_train_test* no pipeline de nome *DataPreparation*. Foi utilizada a técnica estratificada, garantindo que a proporção entre as classes do target seja preservada nos dois conjuntos.

3.9 O aluno criou um pipeline chamado "Treinamento" no MIFlow?

R: Foram criados 3 pipelines de treinamento, um global que seria o resultado final dos treinamentos dos modelos e os parciais que correspondem ao treinamento do Pycaret.

Experiments

Search Experiments

pdblackmamba

pdblackmamba

Provide Feedback

Add Description

Share

Runs

Evaluation

Experimental

Traces

Experimental

Q metrics.rmse < 1 and params.model = "tree"

Time created

State: Active

Datasets

Sort Created

Columns

Group by

New run

Metrics												Tags	
Run Name	Creia	Duration	F1	Log Loss	f1_score	f1_score_dt_test	f1_score_lr_test	log_loss	log_loss_dt_test	log_loss_lr_test	test_size	train_size	Source
PipelineAplicacao	6.7s	-	-	-	0.40143985...	-	-	10.1628037...	-	-	-	-	-
Treinamento_lr	2.8s	-	-	-	-	-	-	-	-	-	-	-	setup
Logistic Regress...	410ms	0.5487	6.7691	-	-	-	-	-	-	-	-	-	tune_model
Logistic Regress...	451ms	0.5242	6.7284	-	-	-	-	-	-	-	-	-	create_model
Treinamento_dt	14.1s	-	-	-	-	-	-	-	-	-	-	-	setup
Decision Tree Cl...	420ms	0.4683	6.56	-	-	-	-	-	-	-	-	-	tune_model
Decision Tree Cl...	3.5s	0.5408	7.4498	-	-	-	-	-	-	-	-	-	create_model
Treinamento	35.3s	-	-	-	0.34320867...	0.34320867...	-	8.25458946...	8.25458946...	-	-	-	-
PreparacaoDados	13.0s	-	-	-	-	-	-	-	-	-	4057	16228	-

4.1 O aluno identificou a diferença entre a base de desenvolvimento e produção?

R: Verificar resposta no tópico do PD.

4.2 O aluno descreveu como monitorar a saúde do modelo no cenário com e sem a disponibilidade da variável alvo?

R: Verificar resposta no tópico do PD.

4.3 O aluno implementou um dashboard de monitoramento da operação usando Streamlit?

R: Verificar resposta no tópico do PD.

4.4 O aluno descreveu as estratégias reativa e preditiva de retreinamento para o modelo em operação?

R: Verificar resposta no tópico do PD.

HOW TO:

Requisitos:

Versão do Python que foi criado o projeto: **Python 3.11** Sistema operacional: **Windows 11** Gerenciador de pacotes do Python: **conda**

Comandos:

Comandos para iniciar o ambiente virtual:

```
conda create --name py11_kedro python=3.11
conda activate py11_kedro
```

Instalar os pacotes:

```
pip install -r requirements.txt
```

Para preparar os dados

```
kedro run --pipeline data_preparation
```

Para treinar o modelo:

```
kedro run --pipeline model_train
```

Após o treinamento, verificar os plots e métricas resultantes dos treinamentos. Utilizando o mlflow :

```
mlflow ui
```

Com isso escolhe-se o melhor modelo (nesse caso a regressao logistica) , e serve ele utilizando o mlflow:

```
mlflow models serve ^  
  -m models:/kobe_lr_model_prod/latest ^  
  --env-manager=local ^  
  --port 5001
```

Para gerar as metricas encima dos dados de producao, acessando o modelo servido pelo Mlflow (escolhido anteriormente):

```
kedro run --pipeline model_evaluation_server
```

Para acessar o formulário em streamlit para testar o modelo em produção e verificar as inferências do modelo, entrar na pasta do streamlit e executar o comando:

```
streamlit run main.py
```