# Kobe Bryant Shot Selection

**Diagrama do projeto**

Planejamento do Projeto

- Definir escopo

- Estabelecer metas comerciais

- Planejar recursos

Aquisição e Preparação de Dados

- Coleta de dados

- Limpeza e processamento

- Engenharia de recursos

- Divisão de dados

Implantação do Modelo

- Coleta de dados

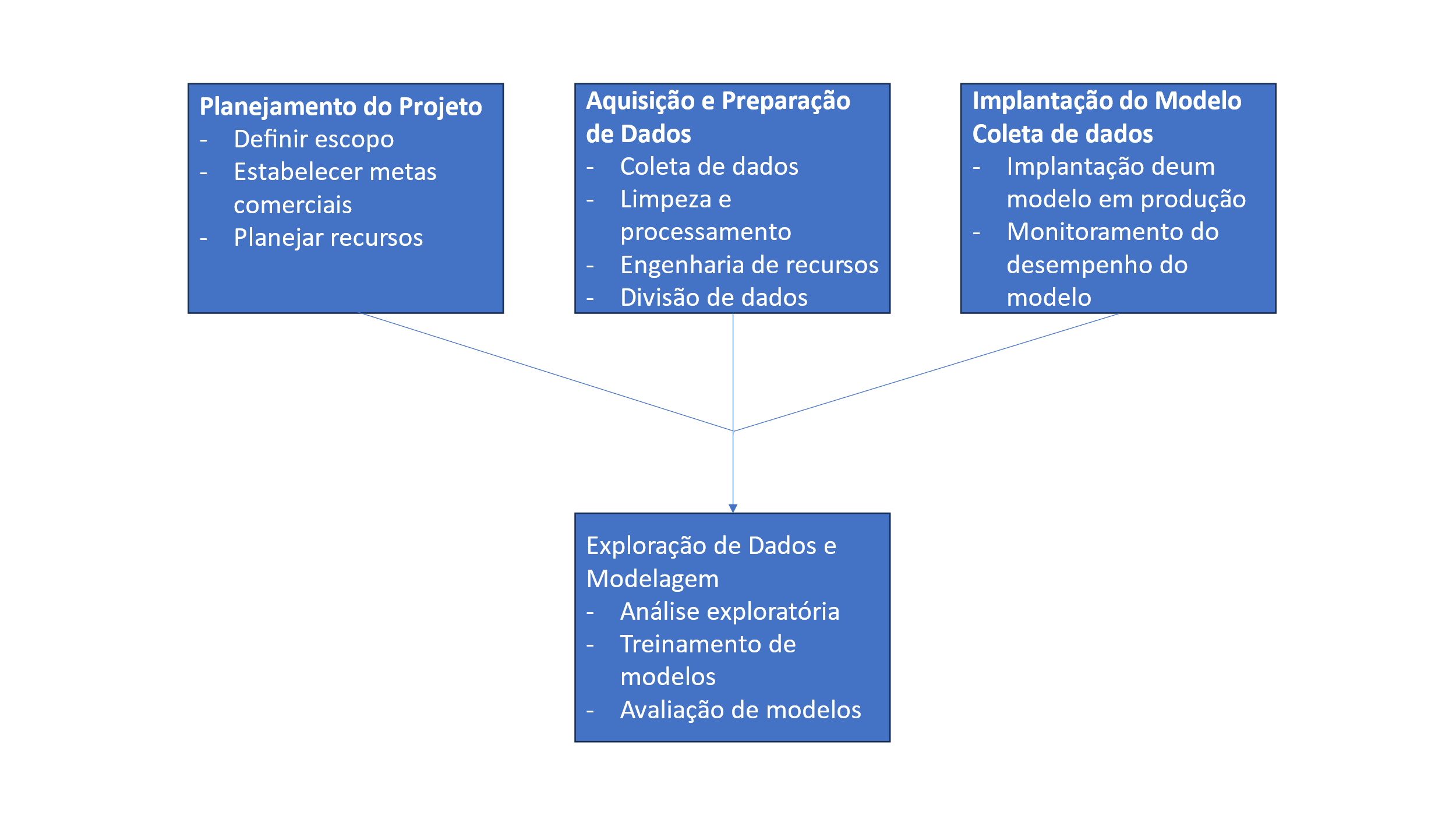
- Implantação deum modelo em produção

- Monitoramento do desempenho do modelo

Exploração de Dados e Modelagem - Análise exploratória

- Treinamento de modelos

- Avaliação de modelos



**Importância dos Pipelines de Desenvolvimento e Produção**

Pipelines de desenvolvimento e produção são como linhas de montagem em uma fábrica, mas para modelos de machine learning. Eles são importantes porque tornam o processo de construção e implantação de modelos mais eficiente e confiável.

Imagine que você está construindo um modelo para prever se Kobe Bryant acertou ou errou um arremesso de basquete. Se você não tiver um pipeline, cada etapa do processo - desde a coleta de dados até o treinamento do modelo e a implantação do modelo em produção - seria feita manualmente. Isso pode levar muito tempo e ser propenso a erros.

Com um pipeline, você pode automatizar muitas dessas etapas. Por exemplo, você pode configurar o pipeline para baixar automaticamente os dados, pré-processá-los para remover valores ausentes e formatá-los corretamente, treinar vários modelos e selecionar o melhor, e implantar o modelo em produção. Isso economiza tempo e reduz a chance de cometer erros.

Além disso, os pipelines garantem consistência e reprodutibilidade. Isso significa que você pode repetir o mesmo processo várias vezes e obter os mesmos resultados. Isso é importante porque você pode precisar atualizar seu modelo periodicamente com novos dados ou ajustar os parâmetros do modelo para melhorar o desempenho.

Em resumo, os pipelines de desenvolvimento e produção são essenciais para construir modelos de machine learning de forma eficiente, confiável e escalável.

**Ferramentas para Construção de Pipelines**

No projeto do Kobe Bryant, usamos várias ferramentas para nos ajudar a construir nossos pipelines de desenvolvimento e produção. Aqui está como cada uma delas nos ajudou:

* **Streamlit**: Usamos o Streamlit para construir um dashboard interativo onde podemos visualizar e interagir com os resultados dos nossos modelos de ML de uma maneira amigável. Por exemplo, criamos um dashboard para visualizar as previsões do modelo sobre se o Kobe acertou ou errou a cesta.
* **MLFlow**: O MLFlow nos ajudou a rastrear nossos experimentos de ML. Isso significa que pudemos acompanhar as diferentes versões dos nossos modelos, os parâmetros usados para treiná-los e as métricas de desempenho associadas a cada modelo. Isso nos permitiu comparar facilmente diferentes modelos e ver como eles se saíram ao longo do tempo.
* **PyCaret**: O PyCaret é uma biblioteca de ML que nos ajudou a treinar nossos modelos de ML com muito menos código. Ele automatizou muitas das etapas que normalmente teríamos que fazer manualmente, como a seleção de modelos e a otimização de hiperparâmetros. Isso nos permitiu experimentar rapidamente diferentes modelos e encontrar o melhor para nossos dados.
* **Scikit-Learn**: O Scikit-Learn é uma biblioteca popular de ML que usamos para treinar nossos modelos. Ele nos forneceu uma ampla gama de algoritmos de ML prontos para uso, bem como ferramentas para pré-processamento de dados e avaliação de modelos. Usamos o Scikit-Learn para treinar modelos de regressão logística e classificação para prever se o Kobe acertou ou errou a cesta.

Deixaremos esse item pra responder no final: “Aborde aspectos como rastreamento de experimentos, funções de treinamento, monitoramento da saúde do modelo, atualização de modelo e provisionamento (deployment).”

**Artefatos do Projeto**

No projeto do Kobe Bryant, vários artefatos são criados para ajudar no desenvolvimento e operação do modelo de previsão de arremessos. Aqui está uma lista deles e o que eles fazem:

1. **Dados Originais (Raw Data):** São os dados brutos coletados, como os arremessos do Kobe Bryant.
2. **Dados Processados (Processed Data):** São os dados limpos e preparados para uso no modelo, removendo dados faltantes e selecionando apenas as informações relevantes.
3. **Base de Treino e Teste (Training and Testing Data):** São conjuntos de dados separados em um usado para treinar o modelo e outro para testar o modelo, garantindo que ele generalize bem para novos dados.
4. **Modelo Treinado (Trained Model):** É o modelo de previsão treinado com os dados de treino, selecionando o melhor algoritmo com base no desempenho nos dados de teste.
5. **Métricas de Avaliação (Evaluation Metrics):** São medidas usadas para avaliar o desempenho do modelo, como precisão, recall e log loss.
6. **API de Modelo (Model API):** É uma interface que permite que outros sistemas façam previsões usando o modelo treinado.
7. **Dashboard de Monitoramento (Monitoring Dashboard):** É uma ferramenta visual para acompanhar o desempenho do modelo em tempo real e identificar a necessidade de ajustes ou re-treinamento.
8. **Registro de Experimentos (Experiment Logging):** Mantém um registro de todos os experimentos realizados durante o desenvolvimento do modelo, incluindo hiperparâmetros, métricas de desempenho e resultados.

**Qual a dimensão resultante do dataset?**

Dimensão do DataFrame filtrado e processado: **(20285, 7)**

**Como a escolha de treino e teste afetam o resultado do modelo final?**

A escolha adequada do conjunto de treino e teste é crucial para avaliar corretamente o desempenho do modelo final. O conjunto de treino é utilizado para ajustar os parâmetros do modelo durante o treinamento, enquanto o conjunto de teste é utilizado para avaliar o desempenho do modelo em dados não vistos durante o treinamento.

Se a divisão entre os conjuntos de treino e teste não for adequada, isso pode levar a uma avaliação enviesada do desempenho do modelo. Por exemplo, se o conjunto de treino for muito pequeno, o modelo pode não capturar corretamente os padrões nos dados, resultando em um desempenho insatisfatório no conjunto de teste. Por outro lado, se o conjunto de treino for muito grande, pode haver overfitting, onde o modelo se ajusta demais aos dados de treino e não generaliza bem para novos dados.

Portanto, é importante realizar uma divisão adequada dos dados, geralmente utilizando técnicas como a divisão aleatória e estratificada, garantindo que o modelo seja treinado e avaliado de forma justa e representativa. A análise dos resultados nos conjuntos de treino e teste permite entender se o modelo está sofrendo de overfitting ou underfitting, auxiliando na escolha e ajuste dos algoritmos e parâmetros do modelo.

**Quais estratégias ajudam a minimizar os efeitos de viés de dados?**

Para minimizar os efeitos de viés de dados, algumas estratégias importantes incluem:

1. Divisão aleatória e estratificada dos dados: Garante que os conjuntos de treino e teste sejam representativos da distribuição dos dados originais, reduzindo o risco de enviesamento nos conjuntos de treino ou teste.
2. Cross-validation: Permite avaliar o desempenho do modelo em múltiplas divisões dos dados, fornecendo uma estimativa mais robusta do desempenho do modelo.
3. Aumento de dados (data augmentation): Aumenta a diversidade dos dados de treinamento introduzindo variações nos exemplos de treinamento, ajudando o modelo a generalizar melhor para dados não vistos.
4. Seleção cuidadosa das métricas de avaliação: Escolher métricas de avaliação adequadas que considerem o contexto do problema pode ajudar a reduzir viéses e garantir que o modelo seja avaliado de forma justa.
5. Detecção e correção de viéses nos dados: Identificar e corrigir viéses nos dados, como desequilíbrios de classe ou dados ausentes, pode ajudar a melhorar a representatividade dos conjuntos de treino e teste.