ANA RAQUEL

PÓS TECH

DATA ANALYTICS

DEPLOY DE APLICAÇÕES

AULA 03

SUMÁRIO

O QUE VEM POR AÍ?

Muito mais do que utilizar as bibliotecas no Python para criar soluções “step by step” para modelos de Machine Learning, uma pipeline tem o objetivo de **construir fluxos de trabalho de várias etapas para otimizar os resultados**. A implementação da pipeline traz benefícios como automação, padronização, escalabilidade e reutilização dos códigos. Vamos aprender como aplicar uma pipeline para modelos de Machine Learning?

HANDS ON

Chegou a hora de criar a pipeline para colocar o modelo em uma aplicação. Antes de criar a pipeline, vamos separar as bases em **treino e teste** e, em seguida, criaremos todos os processos necessários a serem realizados na base de dados dentro de **classes e funções**.

As classes proporcionam uma forma de organizar dados e funcionalidades juntas. Elas servem, sobretudo, para **agrupar as funções de um determinado objeto**. Então teremos classes para **limpar, normalizar** e **equilibrar** a base de dados. Vamos lá?

Acesse o link do GitHub dessa aula para visualizar os notebooks utilizados!

SAIBA MAIS

CONSTRUINDO UMA PIPELINE PARA MODELOS DE MACHINE LEARNING

Você já conhece o termo “pipeline”?

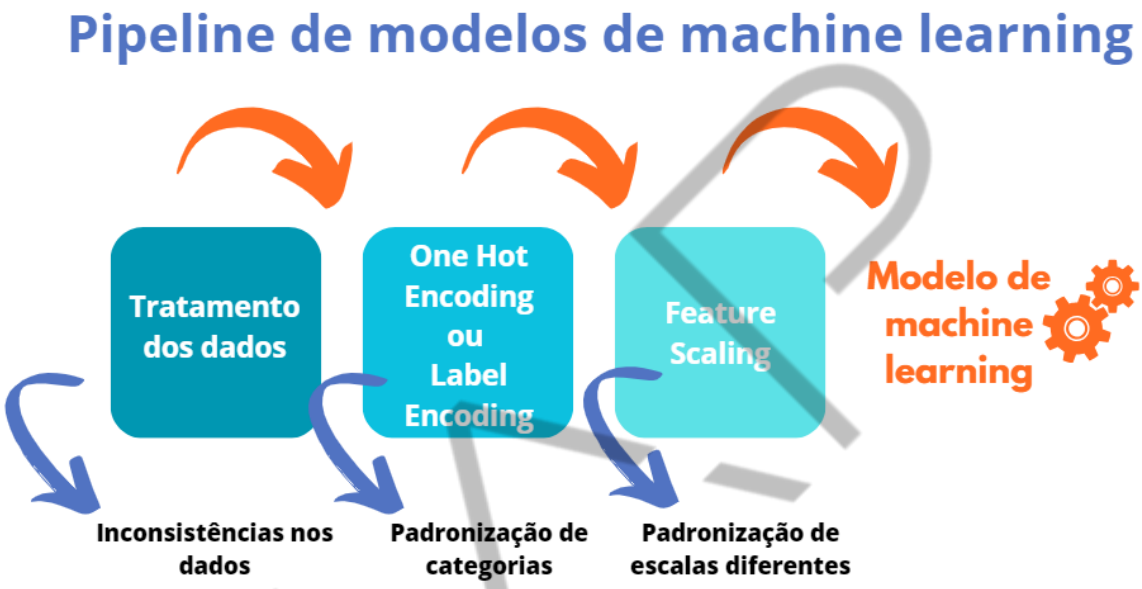

Figura 1 - Modelo de pipeline

Fonte: Elaborado pela autora (2023)

Podemos definir as pipelines como uma sequência de componentes de processamento de dados. Elas são muito comuns em sistemas de aprendizado de máquina, uma vez que existem muitos dados para manipular e muitas transformações para aplicar neles.

Os componentes rodam normalmente de forma assíncrona. Cada componente puxa uma grande quantidade de dados, os processa e envia o resultado para outro armazenador de dados. Então, o próximo componente no pipeline puxa esses dados e envia sua própria saída, e assim por diante.

Quando vamos colocar um modelo em produção, precisamos criar funções e classes no Python para tornar o código da pipeline sustentável e fácil de ser implementado para receber novos dados e, assim, lidar com todo o fluxo de processamento necessário nesses novos dados.

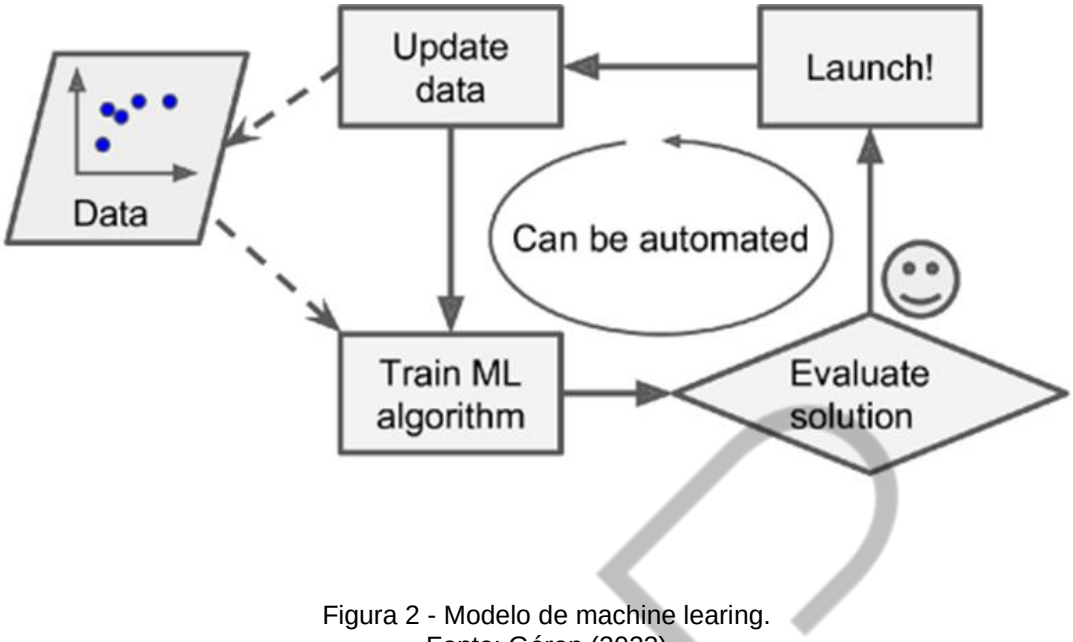

Figura 2 - Modelo de machine learing.
Fonte: Géron (2023)

Basicamente, quando criamos uma pipeline para aprendizado de máquina, precisamos pensar em alguns passos importantes que devem ser “padrões” para toda nova entrada de dados no seu modelo. Quando criamos todos os cuidados de tratamento de dados e feature scaling antes de executar o modelo, **o objetivo é reproduzir todo esse fluxo para toda nova entrada de informação**.

Imagine que vamos prever a concessão de crédito, por exemplo. Quando a pessoa informar a idade e renda, essas variáveis terão que passar pelo processo de normalização antes de entrar no modelo. É aí que as pipelines entram em ação!

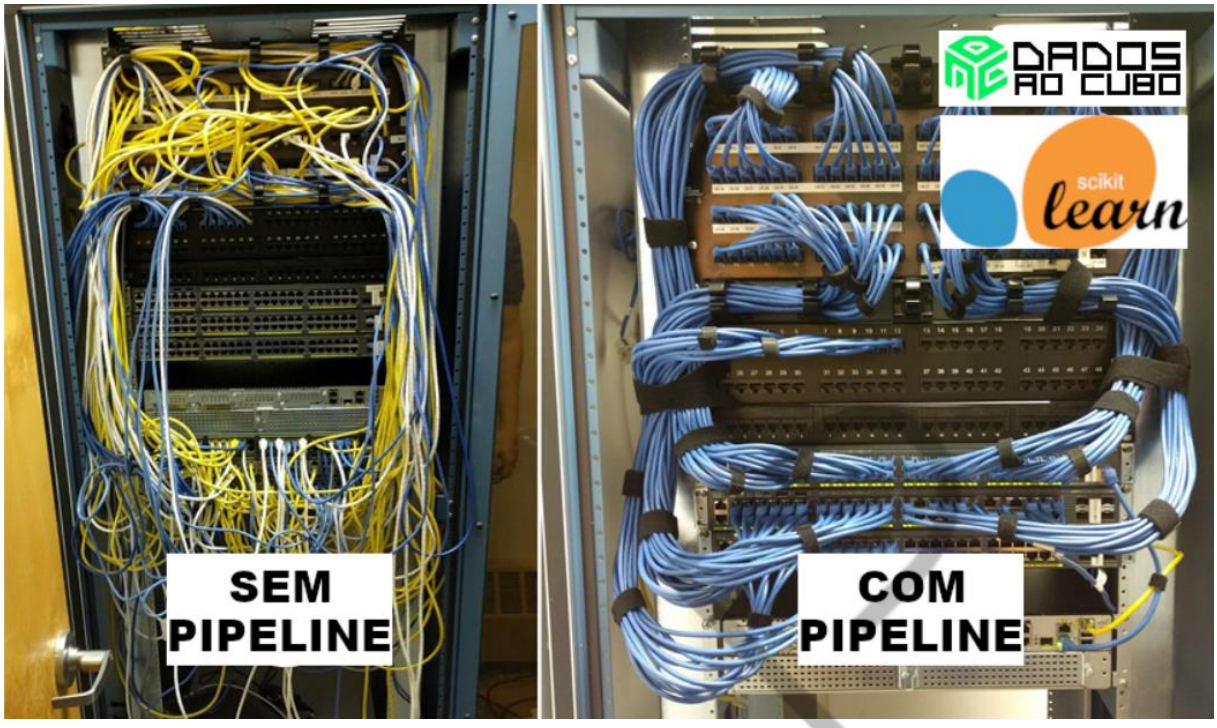

Figura 3 – Sem pipeline x com pipeline.
Fonte: Dados ao Cubo (2022)

Para preparar as nossas classes da pipeline, vamos precisar importar as classes de transformações nos dados, a **BaseEstimator** e o **TransformerMixin**.

O BaseEstimator fornece os métodos **get_params** e **set_params** que todos os estimadores Scikit-learn exigem. Já o TransformerMixin permite que os dados sejam ajustados e transformados.

```
from sklearn.base import BaseEstimator, TransformerMixin
```

Que tal dar uma olhadinha na documentação oficial do Sklearn para o BaseEstimator e o TransformerMixin?

CRIANDO CLASSES EM PYTHON

Para te mostrar como funciona a criação de uma classe, observe o exemplo comentado do código abaixo, para criar uma **classe onde estaremos removendo a coluna "ID_Cliente" do dataframe recebido pela classe**:

```
class DropFeatures(BaseEstimator, TransformerMixin):
  def __init__(self, feature_to_drop = ['ID_Cliente']):
    self.feature_to_drop = feature_to_drop

def fit(self, df):
    return self

def transform(self, df):
df.drop(self.feature_to_drop, axis=1, inplace=True)
return df

  def transform(self, df):
    df.drop(self.feature_to_drop, axis=1, inplace=True)
    return df
```

Primeiro, precisamos definir nossa classe com a palavra **class**. Então, colocamos o nome da classe, que nesse caso será **DropFeatures,** e passamos a ela o **BaseEstimator** e o **TransfromerMixin**.

```
class DropFeatures(BaseEstimator, TransformerMixin):
```

Em seguida vamos modelar a classe com o inicializador **__init__().** Esse é um método especial, que é o primeiro definido dentro da classe. Ele é especial pois sempre é executado quando criamos uma instância de um objeto. Então, automaticamente o Python invoca o __init__() quando você cria o objeto.

O inicializador __init__() sempre receberá **self** e também passaremos outros atributos. O self serve para indicar que nós estamos nos referenciando a alguma coisa do próprio objeto (sejam eles atributos ou métodos). Em seguida, passamos quais colunas serão retiradas e armazenamos isso em "**feature_to_drop**."

```
  def __init__(self, feature_to_drop = ['ID_Cliente']):
```

Após dar dois pontos, na próxima linha digitamos o **self.feature_to_drop** e damos um sinal de igual ao **feature_to_drop**.

```
    self.feature_to_drop = feature_to_drop
```

O próximo passo é criar a função **fit**, que recebe o **self** e o **DataFrame df**, retornando o **self**.

```
  def fit(self, df):
    return self
```

Nossa última função é a **transform**, que também recebe o **self** e o **DataFrame df**. Adicionamos dois pontos e, na linha seguinte, faremos o drop. Aqui, precisamos lembrar de adicionar o axis=1 e o inplace=True. Finalmente, o DataFrame será retornado.

```
  def transform(self, df):
    df.drop(self.feature_to_drop, axis=1, inplace=True)
    return df
```

CONSTRUINDO A PIPELINE

Depois de construir todas as classes necessárias para o seu modelo de Machine Learning ser implementado, será necessário importar a biblioteca **Pipeline** do **Sklearn**. Essa biblioteca permite que você **crie todo passo a passo para realizar o tratamento dos dados e combinar várias etapas de pré-processamento de dados e modelagem em um único objeto,** antes de executar o modelo.

```
from sklearn.pipeline import Pipeline
```

Depois de importar a pipeline, será preciso colocar todas as classes criadas no seu modelo para dentro dessa pipeline.

No case utilizado em aula, a função da pipeline vai receber o **DataFrame**, que será **df_treino** e o **df_teste.** As bases de treino e teste precisam passar por todas as etapas da pipeline. Então, vamos passar para a pipeline a relação de todos os nomes das classes criadas anteriormente e suas respectivas classes. Você aprendeu, nas videoaulas, todas as classes necessárias para a construção da pipeline desse case, tal como a **DropFeatures(),** que você viu no começo dessa aula, e as classes **OneHotEncodingNames(), OrdinalFeature(), MinMax() e Oversample()**.

Em seguida, criamos uma variável chamada **df_pipeline,** que vai receber a **pipeline.fit_transform,** e passamos o **DataFrame**.

Finalmente, pedimos para que a função retorne o df_pipeline com todos os dados já tratados por todas as classes contidas na pipeline:

```
def pipeline(df):

  pipeline = Pipeline([
      ('feature_dropper', DropFeatures()),
      ('min_max_scaler', MinMax()),
      ('OneHotEncoding', OneHotEncodingNames()),
      ('Ordinal_feature', OrdinalFeature()),
      ('oversample', Oversample())
  ])
  df_pipeline = pipeline.fit_transform(df)
  return df_pipeline
```

Depois de criar todas as pipelines necessárias para tratar os dados, chegou o momento de escolher qual é o melhor modelo preditivo de Machine Learning para realizar a concessão de crédito! Na próxima aula, exploraremos esse assunto antes de colocar o modelo na aplicação.

O QUE VOCÊ VIU NESTA AULA?

Nessa aula, você aprendeu como construir uma pipeline de modelo de aprendizado de máquina e qual é a sua importância para tornar o processo mais sustentável.

Tem alguma dúvida ou quer conversar sobre o tema desta aula? Entre em contato conosco pela comunidade do Discord! Lá você pode fazer networking, receber avisos, tirar dúvidas e muito mais.

REFERÊNCIAS

ALENCAR, Valquiria. Machine Learning: Escolhendo o melhor modelo. 2023. São Paulo. 1. Notas de aula.

DOCUMENTAÇÃO SCIKIT-LEARN. **Scikit-learn.** Disponível em: <https://scikit-learn.org/stable/>. Acesso em: 04 Ago. 2023.

GÉRON, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow**. Sebastapol: O'Reilly Media, 2019.

PALAVRAS-CHAVE

Pipeline, BaseEstimator, TransfromerMixin.