Inteligência Artificial e Cases de Negócio

PIPELINE

Prof. Fábio Buiati
buiati@gmail.com

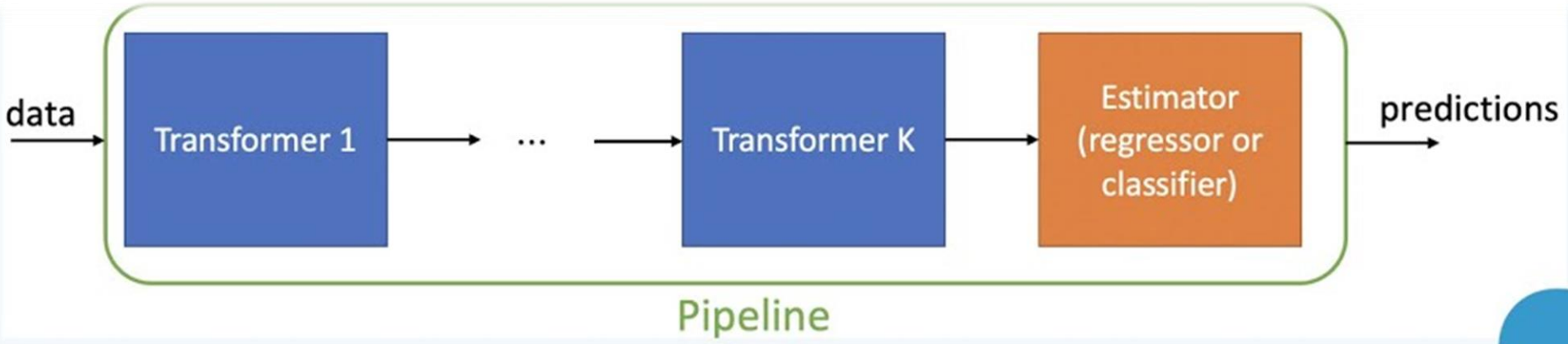
Melhorando a legibilidade e manuntenção dos seus códigos de machine learning

Introdução

A classe Pipeline é uma funcionalidade do Scikit-Learn que ajuda criar códigos que possuam um padrão que possa ser facilmente entendido e compartilhando entre times de cientistas e engenheiros de dados.

Códigos de machine-learning são melhores escritos através de pipelines, que em essência fazem o output de uma dada transformação nos dados se tornarem o input para uma outra transformação que será aplicada nos dados.

Biblioteca sklearn




```
from sklearn.pipeline import Pipeline
# creating pipeline and fitting it on data
pipe = Pipeline([('transformer',PolynomialFeatures(degree=2)),
                 ('estimator',LinearRegression())
                ])
```

Biblioteca sklearn – passo a passo

1. PASSO

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

2. PASSO

O segundo passo é instanciar a classe Pipeline com cada passo do pré-processamento do modelo como um elemento de uma tupla. O primeiro elemento da tupla é um nome que identifique o transformador ou estimador utilizado e o segundo elemento é o transformador ou estimador que irá aplicar uma transformação nos dados ou treinar um algoritmo nos dados, respectivamente.

```
model = Pipeline(steps=[
    ('one-hot encoder', OneHotEncoder()),
    ('imputer', SimpleImputer(strategy='mean')),
    ('tree', DecisionTreeClassifier(max_depth=3, random_state=0))
])
```

Biblioteca sklearn

3. PASSO

O próprio pipeline contém o método fit. Assim, quando chamamos o método fit, todos os passos definidos no Pipeline serão executados na ordem em que aparecem.

```
# criando o modelo usando pipeline
model = Pipeline(steps=[
    ('one-hot encoder', OneHotEncoder()),
    ('imputer', SimpleImputer(strategy='mean')),
    ('tree', DecisionTreeClassifier(max_depth=3, random_state=0))
])

# treinando o modelo
model.fit(X_train, y_train)
train_score = model.score(X_train, y_train)

# avaliando o modelo
test_score = model.score(X_test, y_test)
```

Exemplo

Pré-processamento com ColumnTransformer

Basta você criar um pipeline de pré-processamento para cada variável ou conjunto de variáveis e no fim usar o ColumnTransformer para compor todos os pré-processamentos, indicando na construção do modelo em quais variáveis cada pré-processamento irá atuar.

Ajuda no pré-processamento de diferentes tipos de variáveis, com algumas variáveis numéricas tendo valores faltantes imputados pela mediana, outras pela média, um conjunto de variáveis categóricas com alta cardinalidade que precisa de um pré-processamento diferente de variáveis categóricas que apresentam baixa cardinalidade.

```
# pipeline para pré-processamento das variáveis Age e Fare
num_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median'))
])

# pipeline para pré-processamento das variáveis Sex e Embarked
cat_transformer = Pipeline(steps=[
    ('one-hot encoder', OneHotEncoder())
])

# Compondo os pré-processadores
preprocessor = ColumnTransformer(transformers=[
    ('num', num_transformer, ['Age', 'Fare']),
    ('cat', cat_transformer, ['Sex', 'Embarked'])
])

# criando o modelo usando pipeline
model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('tree', DecisionTreeClassifier(max_depth=3, random_state=0))
])
```

Conclusão

As classes Pipeline e ColumnTransformer são o que há de supra sumo no scikit-learn para te ajudar a escrever um código limpo e de fácil manutenção, que vai te ajudar na hora de mandar aquele código pra ser colocado em produção pro time de engenharia.

Além disso, ele estabelece um padrão que facilita a leitura por qualquer cientista de dados que tenha domínio dessas ferramentas.

Caso Prático

Fábio Buiati
buiati@gmail.com