**SUMÁRIO**

## O QUE VEM POR AÍ?

Nesta aula, introduziremos os conceitos básicos dos principais frameworks de machine learning que podemos utilizar quando trabalhamos com Python: Keras, Scikit-learn, TensorFlow e PyTorch. Esses frameworks são amplamente usados na comunidade de machine learning e oferecem diversas ferramentas e funcionalidades para construção, treinamento e avaliação de modelos de aprendizado de máquina.

## HANDS ON

Nesta aula prática, demonstraremos como criar e publicar uma biblioteca no PyPI. Também exploraremos a importância dessa prática para a organização e a reutilização de código e aprenderemos a criar nossos próprios módulos e bibliotecas de maneira eficiente.

## SAIBA MAIS

Nesta aula nós conheceremos algumas técnicas e frameworks que podemos utilizar em Machine Learning com Python. Iniciando pelo Scikit-Learn, ele é uma das bibliotecas mais populares para Machine Learning em Python. Conhecida por sua simplicidade e eficácia, ela oferece uma variedade de ferramentas avançadas para desenvolvedores(as) experientes.

### Pipelines para pré-processamento de dados

Pipelines são uma maneira eficiente de encadear várias etapas de pré-processamento de dados e modelagem. Isso é crucial para manter o código organizado e facilitar a reprodução de experimentos.

Vejamos um exemplo prático:

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

# Carregar um dataset exemplo
data = load_iris()
X, y = data.data, data.target

# Dividir os dados em conjuntos de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Definir o pipeline
pipeline = Pipeline([
  ('scaler', StandardScaler()),
  ('pca', PCA(n_components=3)),
```

```
    ('classifier', RandomForestClassifier())
])

# Treinar o pipeline
pipeline.fit(X_train, y_train)

# Fazer previsões
predictions = pipeline.predict(X_test)

# Mostrar previsões
print(predictions)
```

**Otimização de hiperparâmetros com grid search**

O Grid Search é uma técnica poderosa para encontrar a combinação ideal de hiperparâmetros para um modelo, o que resulta em um desempenho otimizado.

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.datasets import load_iris

# Carregar um dataset exemplo
data = load_iris()
X, y = data.data, data.target

# Dividir os dados em conjuntos de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Definir o pipeline
```

```
pipeline = Pipeline([
   ('scaler', StandardScaler()),
   ('pca', PCA(n_components=3)),
   ('classifier', RandomForestClassifier())
])

# Definir a grade de parâmetros para GridSearch
param_grid = {
   'classifier__n_estimators': [50, 100, 150],
   'classifier__max_depth': [10, 20, 30]
}

# Executar GridSearchCV
grid_search = GridSearchCV(pipeline, param_grid, cv=5)
grid_search.fit(X_train, y_train)

# Mostrar os melhores parâmetros encontrados
print(grid_search.best_params_)
```

**Keras**

Keras é uma biblioteca de deep learning de alto nível, popular por sua simplicidade e facilidade de uso. Ela foi projetada para permitir a criação rápida e eficiente de modelos de aprendizado profundo, sendo a escolha preferida de muitos indivíduos pesquisadores e desenvolvedores.

**Interface de Alto Nível**

A principal característica do Keras é sua interface de alto nível, que abstrai muitas das complexidades envolvidas no desenvolvimento de modelos de deep learning. Isso facilita o trabalho de desenvolvimento, permitindo que usuários se concentrem mais na experimentação e menos na implementação técnica. Com o Keras, você pode rapidamente construir, treinar e avaliar modelos de aprendizado profundo com poucas linhas de código.

**Versatilidade**

O Keras é incrivelmente versátil. Ele suporta múltiplos backends, como TensorFlow, Theano e Microsoft Cognitive Toolkit (CNTK). Isso significa que você pode escolher o backend que melhor se adapta às suas necessidades ou infraestrutura e ainda aproveitar a simplicidade da interface Keras.

**Conceitos básicos**

1. **Construção de modelos com camadas personalizadas**: o Keras permite a criação de modelos personalizados usando camadas customizadas, proporcionando maior flexibilidade para a construção de arquiteturas específicas. Você pode criar suas próprias camadas, funções de ativação e até mesmo modelos, ajustando-os às necessidades específicas do seu projeto.
2. **Treinamento de modelos com callbacks**: os callbacks no Keras permitem realizar ações específicas durante o treinamento do modelo, como salvamento de pesos e ajuste dinâmico de taxa de aprendizado, entre outros.

**TensorFlow**

O TensorFlow é um poderoso framework de machine learning e deep learning desenvolvido pela Google. Ele é amplamente conhecido por sua versatilidade e escalabilidade, sendo utilizado em uma variedade de cenários de machine learning, desde pesquisas acadêmicas até aplicações industriais em grande escala.

Uma das inovações mais significativas do TensorFlow 2.x é a introdução do modo de execução eager, que torna a construção de modelos mais intuitiva e flexível. Com o modo eager, os cálculos são avaliados imediatamente, facilitando a depuração e a iteração rápida durante o desenvolvimento.

**PyTorch**

O PyTorch é uma popular biblioteca de deep learning desenvolvida pelo Facebook AI Research (FAIR). Ela é amplamente reconhecida por sua flexibilidade e expressividade, tornando-se a escolha preferida de muitas pessoas pesquisadoras e desenvolvedoras.

**Flexibilidade**

Uma das principais vantagens do PyTorch é sua flexibilidade. Ele permite a construção dinâmica de grafos computacionais, o que significa que o grafo é construído em tempo de execução. Isso torna o desenvolvimento e a depuração de modelos de deep learning muito mais intuitivos, permitindo a modificação do fluxo de dados e a adição de operações de forma mais natural e menos restritiva.

**Expressividade**

A expressividade do PyTorch facilita a implementação de modelos complexos. Com uma interface de alto nível que se integra perfeitamente com o Python, especialistas em desenvolvimento podem aproveitar todas as funcionalidades da linguagem, como loops e condições, para criar modelos mais complexos e específicos. Essa integração com o Python permite utilizar o amplo ecossistema de bibliotecas e ferramentas disponíveis na linguagem, potencializando a capacidade de criar soluções inovadoras e eficientes.

## O QUE VOCÊ VIU NESTA AULA?

Nesta aula, introduzimos os conceitos básicos dos principais frameworks de machine learning que podemos utilizar quando trabalhamos com o Python: Keras, Scikit-learn, TensorFlow e PyTorch. Esses frameworks são amplamente usados na comunidade de machine learning e oferecem diversas ferramentas e funcionalidades para construção, treinamento e avaliação de modelos de aprendizado de máquina.

# REFERÊNCIAS

BROWNLEE, J. **Deep Learning with Keras**: Implementing deep learning models and neural networks with the power of Python. [s.l.]: Machine Learning Mastery, 2017.

GULLI, A.; KAPOOR, A.; PAL, S. **Deep Learning with TensorFlow 2 and Keras**. [s.l.]: Packt Publishing, 2019.

PEDREGOSA, F. et al. **Scikit-learn**: Machine Learning in Python. [s.l.]: Journal of Machine Learning Research, 2011.

RAO, K.; R. and M. KELLEHER. **Deep Learning with PyTorch**. [s.l.]: Manning Publications, 2018.

## PALAVRAS-CHAVE

**Palavras-chave**: Keras. Pytorch. TensorFlow.