Módulo #13

Finalmente chegamos ao último módulo do curso de Python Iniciante. Você já sabe bastante coisa sobre programação e existe uma área da computação chamada machine learning que está em alta no mercado.

Se caso você precise trabalhar como programador nesta área, vamos te ensinar como fazer este tipo de implementação em Python. Com isso você vai praticar tudo que aprendeu sobre programação e, ainda, adquirir novas habilidades.

Machine Learning ou Aprendizado de Máquina como falamos no português é uma subárea da inteligência artificial que visa simular a capacidade cognitiva humana do aprendizado. Com isso, podemos hoje criar modelos que aprendem a realizar diversas tarefas e aprendem padrões sobre grandes coleções de dados, como por exemplo: modelos que fazem reconhecimento facial, que aprendem a predizer se um determinado tumor é maligno ou benigno, que classificam imagens, que prediz qual seria o valor ideal da venda de um imóvel, faz predições para o mercado financeiro, entre diversas outras aplicações.

O processo de aprendizado dos modelos se da a partir da associação de diversos dados sobre o problema em questão, de modo que os padrões sobre os dados(observações) são percebidos para conseguir chegar na resposta esperada (target). Por exemplo:

Observações	**Targets**
Dados passado sobre mercado financeiro	Preço das ações futuras
Características de imagens de tumores malignos e benignos	Dada uma nova imagem de um tumor predizer se é maligno ou benigno

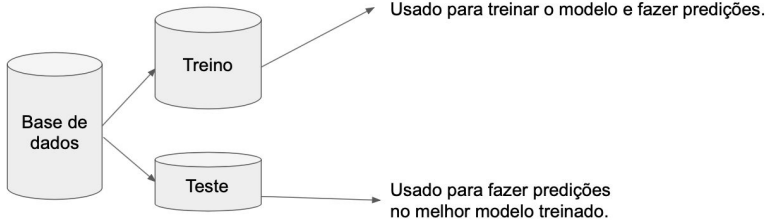
Em Python já se tem muitas bibliotecas prontas que nos auxiliam a criar nossos modelos de machine learning. Aí vai uma lista com as principais:

Tensor Flow
Keras
PyTorch
SciKit-Learn

Depois de entendermos a ideia básica de um modelo de Machine Learning, talvez uma das principais dúvidas que surge é “como saber se um modelo realmente aprendeu determinada tarefa?”

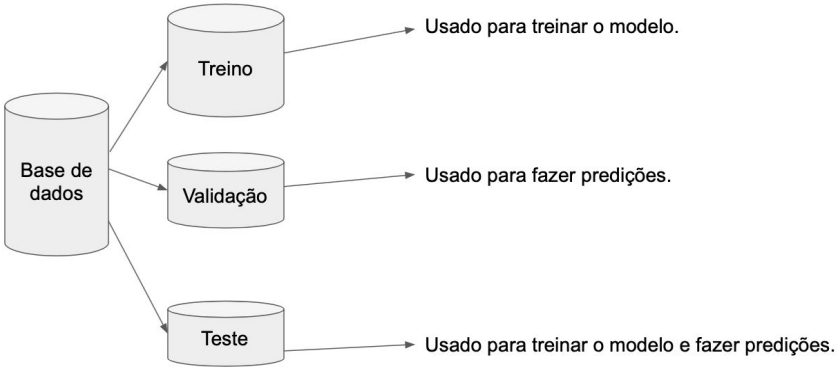
Para identificarmos o desempenho de um modelo é preciso fazer o processo de avaliação que consiste em:

Dividir a base de dados em treino e teste ou treino, teste e validação. A maior parte dos dados ficam para o treinamento, porque quanto mais exemplos diferentes o modelo conhecer, mais ele aprende. Os demais dados ficam para teste ou metade para teste e a outra metade para validação. Como mostra a imagem a seguir:



Dados de treino: são usados para treinar os modelos e fazer predições durante o treinamento. Com estas predições é possível ajustar os parâmetros do modelo, testar, usar outros algorítmos até encontrar o melhor modelo possível.

Dados de teste: são usados para realizar predições no melhor modelo encontrado para identificarmos a performance do modelo quando ele é exposto a dados que ele não conheceu no processo de treinamento.



Dados de treino: usados para treinar os modelos.
Dados de validação: usados para fazer as predições de treinamento. Estas predições são usadas para ajustar os modelo.
Dados de teste: usados, por fim, para realizar predições no melhor modelo encontrado.

Nada como uma boa prática para nos ajudar a compreender o conteúdo. Então vamos ver um exemplo de criação e avaliação de um modelo de machine learning?

Trabalharemos com uma base de dados composta por características extraídas de tumores. Temos na base dados sobre tumores malignos (cancerígenos) e benignos (não cancerígenos). O modelo vai aprender sobre estes dados e, dada a característica de um novo tumor que não conhecemos se é maligno ou benigno, o modelo vai nos passar o diagnóstico, ou seja, vai nos informar se aquele novo tumor é cancerígeno ou não.

Vamos trabalhar com a seguinte base de dados: https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+%28diagnostic%29

Esta base também está disponível na biblioteca scikit-learn: https://scikit-learn.org/stable/datasets/toy_dataset.html#breast-cancer-dataset

Vamos então carregar o dataset:

```
#carregando o dataset que está disponível na biblioteca sklearn

from sklearn.datasets  import load_breast_cancer
dataset = load_breast_cancer()

#Veja tudo que temos na base de dados
dataset
```

Agora, vamos organizar os dados em um dataframe: No dataframe temos os dados de cada tumor e também o target de cada um deles. O target indica se o tumor é maligno ou benigno.

```
#Agora vamos organizar a base de dados em um dataframe usando a
biblioteca pandas.
import pandas as pd
df = pd.DataFrame(dataset.data, columns=[dataset.feature_names])
#add o target
df['diagnostico']=dataset.target
df
```

Vamos dividir a base em observações e target. Algumas bases já vem divididas, outras como esta não. Para isso, identifique o que é o target e o que é observação e divida. As observações devem ser referenciadas como X e os targets como Y. Por exemplo:

X = observações
Y = targets

```
X=df[dataset.feature_names]
Y=df['diagnostico']
```

Agora, divida X e Y em treino, teste e validação (pode dividir também em treino e teste)

```
#70% para treino
percentual_treino= 70
qtd_linhas= len(df)

qtd_treino= round((percentual_treino*qtd_linhas)/ 100)
restante= round(qtd_linhas-qtd_treino)

Xtreino=X[:qtd_treino]
Ytreino=Y[:qtd_treino]

Xteste=X[qtd_treino:qtd_treino+ round(restante/ 2)]
Yteste=Y[qtd_treino:qtd_treino+ round(restante/ 2)]
```

```
Xval=X[qtd_treino+round(restante/2):]
Yval=Y[qtd_treino+round(restante/2):]
```

A base de dados está pronta, então vamos utilizar agora o algoritmo SVM para treinar o modelo.

```
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
import sklearn.metrics as metrics
import numpy as np

#Criando um modelo SVM. Você pode modificar os parâmetros C e
gamma para tentar melhor a acurácia do modelo
model=SVC(C=0.99, gamma=0.0000001)

#Treinando o modelo. Use aqui suas observações de treino e os
targets de treino
model.fit(Xtreino, Ytreino)

#Fazendo predições. Use aqui sua base de teste
result = model.predict(Xteste)

#mostrando os resultados preditos pelo modelo
print('Resultado:')
print(result)

print('Resultado Esperado:')
#Os resultados esperados são os targets de teste, mostre-os a
seguir
print(Yteste)
```

```
#agora vamos calcular a acurária do modelo usando a base de
testes
print("Acurácia:", metrics.accuracy_score(result,Yteste))
```

Vamos ver a performance deste modelo usando a base de validação

```
result = model.predict(Xval)
#mostrando os resultados preditos pelo modelo
print('Resultado:')
print(result)

print('Resultado Esperado:')
#Os resultados esperados são os targets de teste, mostre-os
a seguir
print(Yval)

#agora vamos calcular a acurária do modelo usando a base de
testes
print("Acurácia:", metrics.accuracy_score(result,Yval))
```

Lembre-se

A avaliação da performance do modelo é muito importante, com ela conseguimos saber se o modelo aprendeu determinada tarefa ou não. Podemos avaliar de duas maneiras: o quanto o modelo está errando ou o quanto o modelo está acertando. Para os problemas de regressão é calculada a diferença entre os valores retornados pelo modelo e o valor esperado (o target da base de dados). Para os problemas de classificação, normalmente, é calculada quantas observações foram definidas em classes corretas ou em classes erradas.