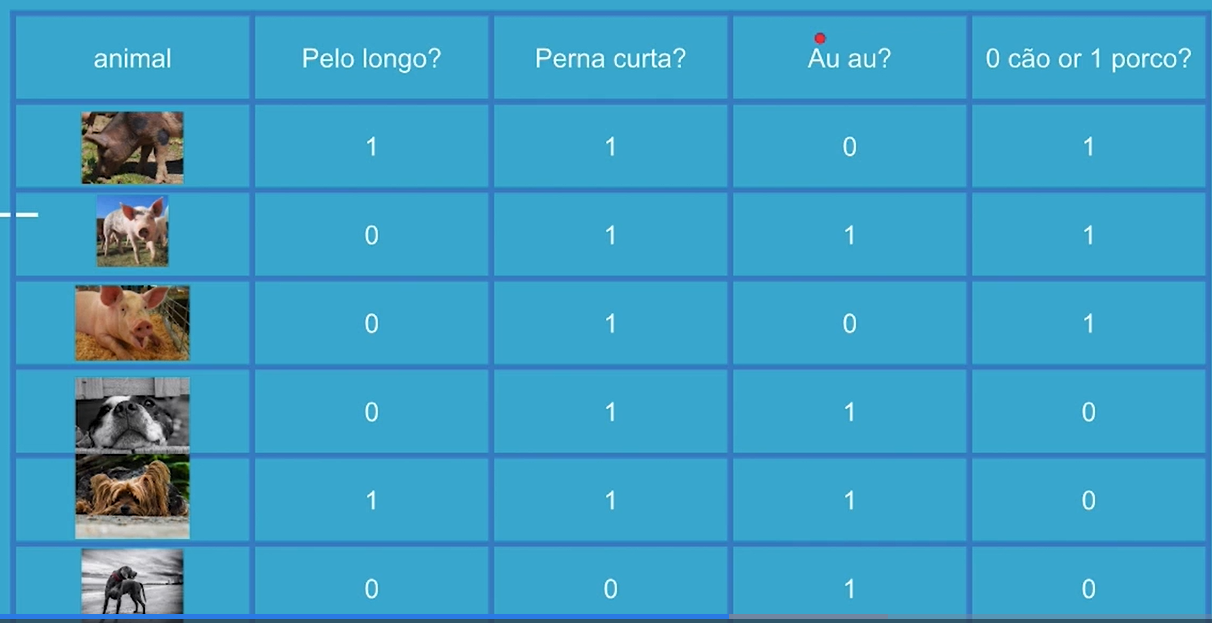
**Machine Learning Completo**

**Machine Learning**

Os notebooks dessa formação estão no seguinte link: <https://drive.google.com/drive/u/0/folders/1XG2od1spsl4OnJLrIs7pztL4c1UmvFDt>

**Classificação com SKLearn**

1. **Aula 1 – Introdução à Classificação:**
   1. Faremos uma classificação para saber se os dados que estamos utilizando é um porco ou um cachorro, por tanto, é uma classificação binária, ou seja, serão classificados em 0 e 1.
      1. Faremos a classificação a partir de características que ensinaremos para o algoritmo, isto é, se o porco tem pelo longo e perna curta, colocaremos 1, se faz se faz “au au”, 0. Essas são as features que serão utilizadas para gerar a classificação de 1 para porco e 0 para cachorro:



* + 1. Através de testes veremos se o nosso algoritmo estimador é bom ou não a partir da taxa de acertos.
  1. A biblioteca que utilizaremos para criar e treinar um modelo é a sklearn.
     1. Dentro dela temos a skleanr.svm, utilizado para importar o LinearSVC, método para criação do modelo.

from sklearn.svm import LinearSVC

* + - 1. Para criar o modelo basta atribuir o método LinearSVC em uma variável.
      2. Para treinar o modelo utilizamos o método .fit não modelo, passando como parâmetro os dados, isto é, uma lista com todos os itens com a lista de suas features dentro, e as classes, dizendo se cada item dos dados é 0 ou 1 (cachorro ou porco, no caso):

# Features: pelo longo, perna curta e faz "au au". (1 == sim, 0 == não)

porco1 = [0, 1, 0]

porco2 = [0, 1, 1]

porco3 = [1, 1, 0]

cachorro1 = [0, 1, 1]

cachorro2 = [1, 0, 1]

cachorro3 = [1, 1, 1]

# 1 => porco, 0 => cachorro

dados = [porco1, porco2, porco3, cachorro1, cachorro2, cachorro3]

classes = [1,1,1,0,0,0]

model = LinearSVC()

model.fit(dados, classes)

* + - * 1. Esse é o método de treinamento assistido, onde dizemos para o algoritmo o que cada item dos dados é.
      1. A partir disso podemos utilizar o predict() no modelo, passando como parâmetro uma lista, para que ele nos diga se é um porco ou charro. Ele irá nos retornar um Array com o número 0 ou 1, a depender da previsão que ele fez:

animal\_misterioso = [1,1,1]

model.predict([animal\_misterioso])



* + - 1. Com o modelo treinado podemos fazer o teste com outros animais misteriosos:

misterio1 = [1,1,1]

misterio2 = [1,1,0]

misterio3 = [0,1,1]

testes = [misterio1, misterio2, misterio3]

previsoes = model.predict(testes)

previsoes



* + - 1. Entretanto, sabemos que o último animal misterioso era na verdade um porco, e não um cachorro, ou seja, o nosso modelo errou. Precisamos testar a taxa de acerto dele, ou a acurácia.
    1. A sklearn.metrics é utilizada para importar o accuracy\_score, método para saber a taxa de acertos que o nosso modelo teve a partir da previsão que ele fez quando comparado com os valores reais, isto é, com os dados que sabemos qual é a resposta.

from sklearn.metrics import accuracy\_score

* + - 1. Utilizamos o accuracy\_score(), passando como parâmetro os dados reais e depois a previsão, isto é, os dados que sabemos qual é a resposta e em seguida a previsão que nosso modelo fez. Esse método retorna a porcentagem de acerto:

testes\_classes = [0,1,1]

taxa\_de\_acerto = accuracy\_score(testes\_classes, previsoes)

print("Taxa de acerto de:", taxa\_de\_acerto \* 100)



* 1. O que aprendemos:
     1. Treinar algoritmos;
     2. Definir características (features) do que desejamos classificar;
     3. Classificar em categorias;
     4. Utilizar o módulo linearSVC e accuracy\_score;
     5. Utilizar o método fit;
     6. Prever dados com a função predict;
     7. Calcular a taxa de acerto do modelo;
     8. Comparar testes e previsões;
     9. Utilizar a função soma;
     10. Padronizar variáveis.