Keras é uma biblioteca aberta de Deep Learning implementada utilizando a base do TensorFlow para diversas linguagens/plataformas, como Python e R, como foco na sua facilidade para utilização. Ela permite modelar e treinar modelos de redes neurais com poucas linhas de código.

Nesse exemplo, vamos utilizar o Keras para criar um modelo capaz de classificar imagens em carros ou aviões

1. **Carregando os dados**
2. **Criar o modelo utilizando Keras**

- Agora que nossos dados foram carregados e ajustados entre entradas e saídas, podemos definir nosso modelo utilizando a biblioteca Keras.

- Modelos Keras são definidos como uma sequência de camadas.

- Esse projeto utilizará 5 camadas, as quais as duas primeiras utilizarão a função de ativação [ReLU](https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/) e a função Softmax na última. Podemos especificar o número de neurônios no primeiro argumento, e a função de ativação com o parâmetro **activation**.

###### 3. Compilando o modelo

Com nosso modelo definido, precisamos compilá-lo. A compilação ocorre utilizando a biblioteca TensorFlow, onde a melhor forma de representar a rede para treinar e fazer predições utilizando o hardware disponível é selecionada.

Ao compilar, precisamos especificar algumas propriedades, como a função de perda, otimizador e a métrica que será utilizada para avaliar o modelo. Foge ao escopo do tutorial apresentar esses conceitos, mas vamos utilizar a função de perda de Entropia Cruzada Binária, o otimizador Adam (que utiliza o gradiente descendente) e acurácia como métrica.

###### 4. Treinando o modelo

Como nosso modelo definido e compilado, precisamos treiná-lo, ou seja, executar o modelo utilizando nossos dados. Para treinar o modelo, basta chamar a função fit() para o modelo.

O treinamento ocorre através de épocas, e cada época é dividida em lotes, onde uma época é uma passagem por todas as linhas do conjunto de testes, e um lote é composto de uma ou mais amostras (quantidade definida pelo usuário) consideradas pelo modelo antes que seja feita a atualização dos seus pesos.  Aqui, vamos executar 10 épocas com lotes de tamanho 10 (um número considerado pequeno).

**5. Avaliando o modelo**

Agora que estamos com nosso modelo treinado, precisamos avaliá-lo. Essa avaliação vai dizer o quão bem o modelo foi construído utilizando o conjunto de dados de treinamento. Pode-se separar os dados entre dados de treinamento e de teste para avaliar o desempenho do modelo com novos dado.

Para avaliar como o modelo se comportou para os dados de treinamento, basta passar os mesmos dados de entrada e saída para a função **evaluate()**. Essa função retorna uma lista com a perda e a acurácia do modelo para o conjunto de dados.