

Introdução ao TensorFlow e Keras

Tiago Ajala Mielnik

Semana Integrada da Computação - 2023

ASSUNTOS ABORDADOS

- ❑ APRENDIZADO DE MÁQUINA
- ❑ REDES NEURAIS E DEEP LEARNING
- ❑ PREVISÃO DE SÉRIES TEMPORAIS
- ❑ CRIPTOATIVOS
- ❑ TENSORFLOW E KERAS
- ❑ DESENVOLVIMENTO DE MODELOS DE PREVISÃO DE PREÇOS DE CRIPTOATIVOS

CRONOGRAMA DAS ATIVIDADES

01

INTRODUÇÃO

02

PREPARO DO
AMBIENTE

03

PRÉ-
PROCESSAMENTO
DOS DADOS

04

CONSTRUÇÃO DOS
MODELOS

05

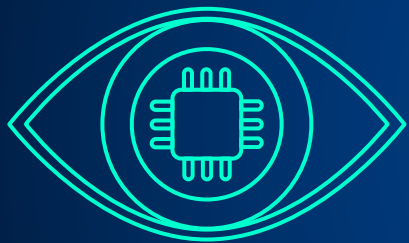
TESTES DOS
MODELOS

06

CONCLUSÃO

REQUISITOS

- ❑ PYTHON BÁSICO
- ❑ CONHECIMENTO DE ESTRUTURAS DE DADOS: ARRAYS E MATRIZES



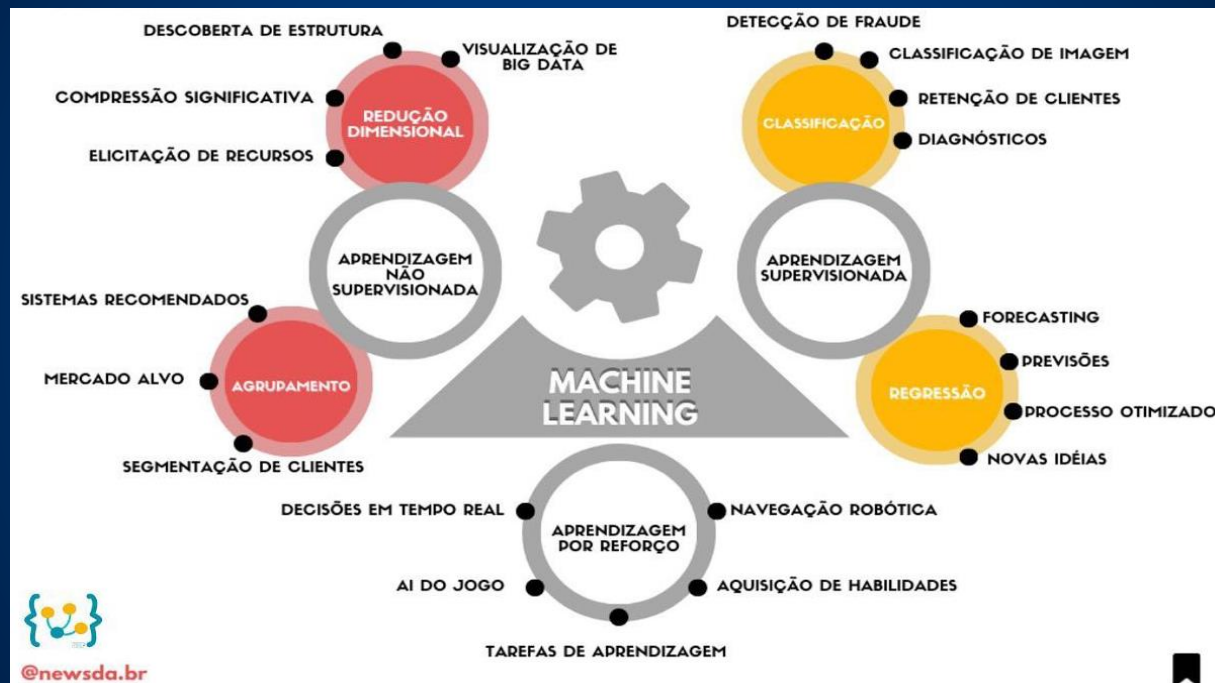
01

INTRODUÇÃO

APRENDIZADO DE MÁQUINA

- ❑ SUB-ÁREA DE INTELIGÊNCIA ARTIFICIAL QUE EXPLORA O ESTUDO E DESENVOLVIMENTO DE SISTEMAS COMPUTACIONAIS BASEADOS NO APRENDIZADO DE DADOS;
- ❑ PARADIGMA DIFERENTE DA PROGRAMAÇÃO TRADICIONAL: REGRAS/PADRÕES NÃO SÃO PROGRAMADOS, MAS SÃO APRENDIDOS.

TIPOS DE MACHINE LEARNING

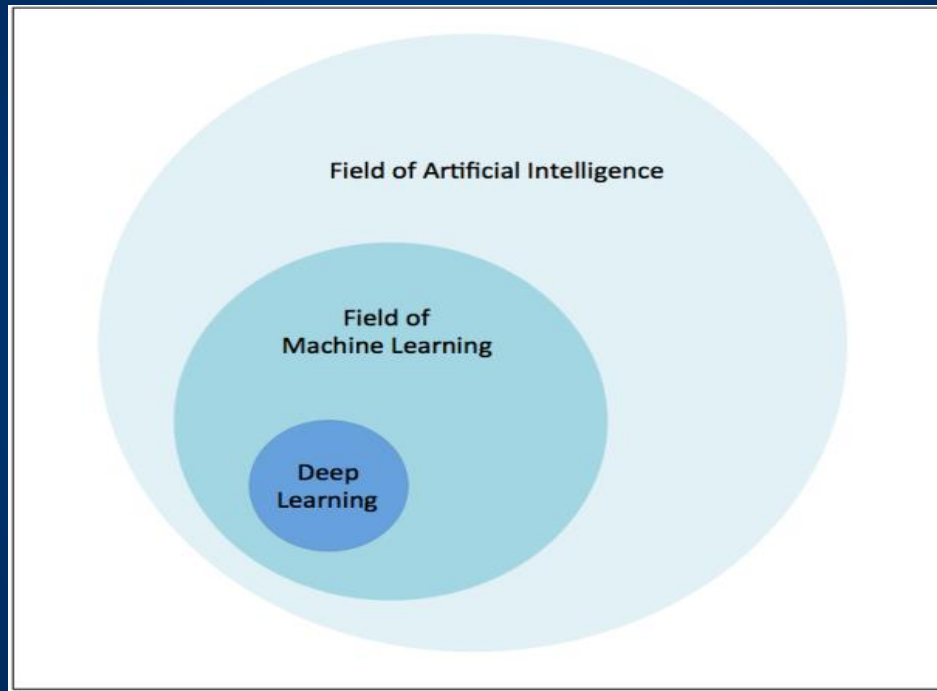


Fonte: [Facebook - Núcleo de Estudos em Web Semântica e Análise de Dados / USP](#)

DEEP LEARNING

- ❑ SUB-CONJUNTO DE MACHINE LEARNING, CARACTERIZADO PELO APRENDIZADO PROFUNDO E REPRESENTADO PELAS REDES NEURAIS ARTIFICIAIS DE MÚLTIPAS CAMADAS.

DEEP LEARNING

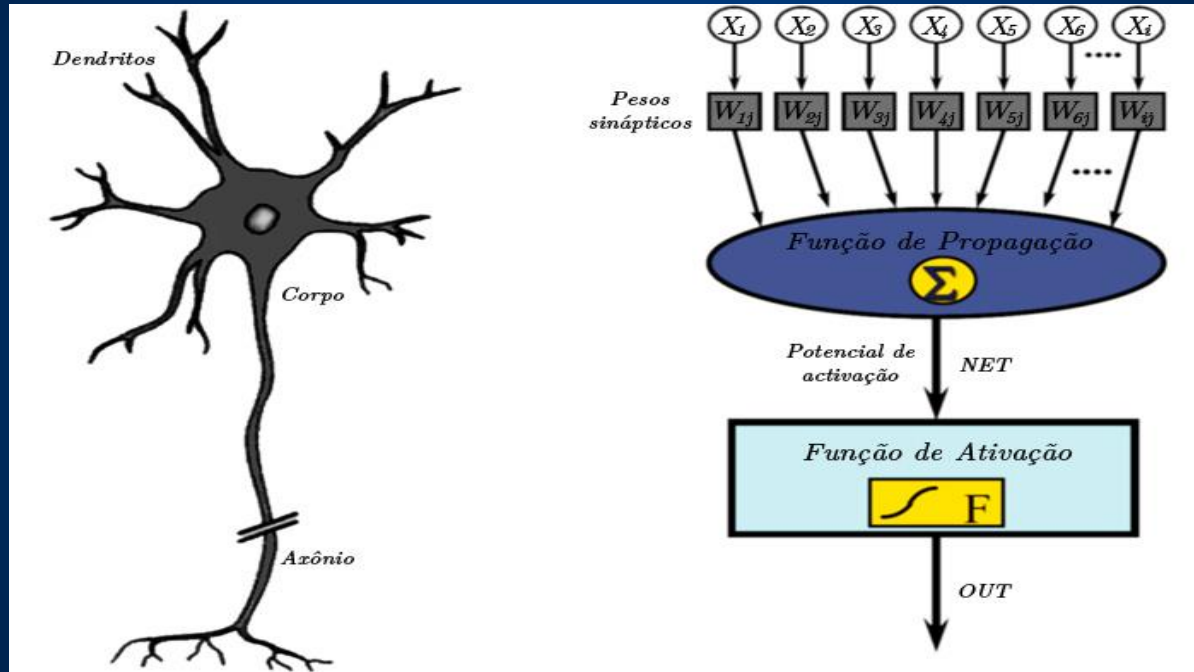


Fonte: J. Patterson and A. Gibson, 2019.

REDES NEURAIS ARTIFICIAIS

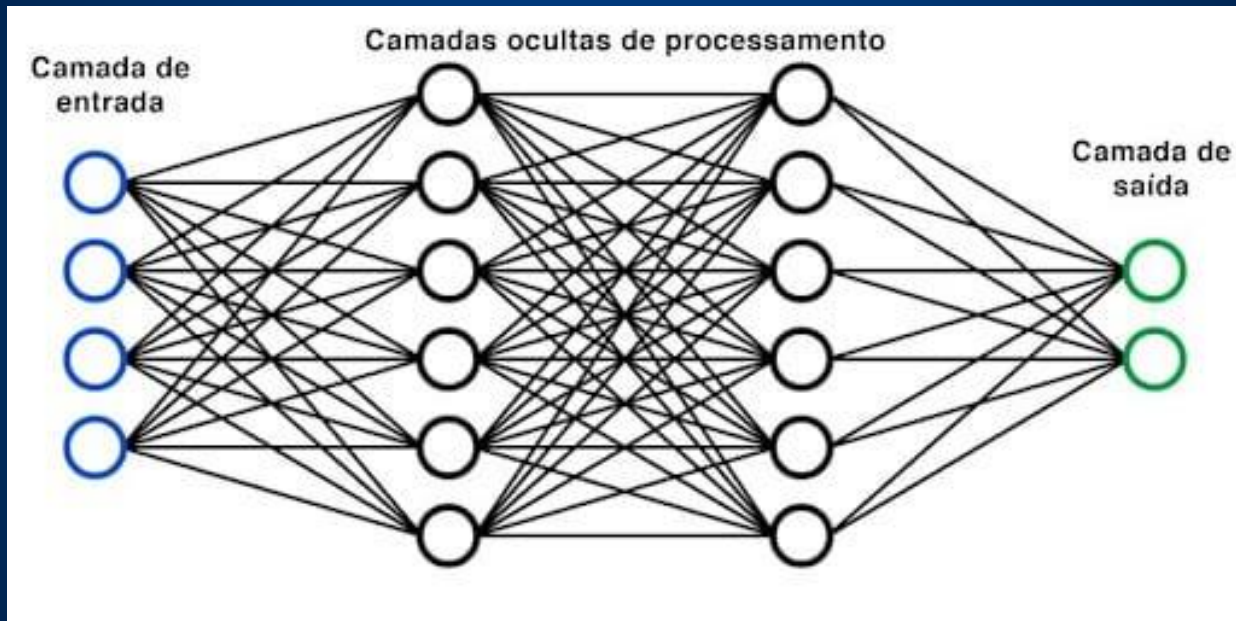
- ❑ MODELAGEM COMPUTACIONAL PROJETADA PARA SIMULAR O FUNCIONAMENTO DE UM CÉREBRO BIOLÓGICO NA EXECUÇÃO DE TAREFAS;
- ❑ POSSUEM CAPACIDADE DE GENERALIZAÇÃO, OU SEJA, CONSEGUEM APRENDER E PRODUZIR RESULTADOS ADEQUADOS PARA ENTRADAS QUE NÃO ESTAVAM PRESENTES DURANTE O SEU TREINAMENTO;
- ❑ APRENDIZADO SOBRE GRANDES QUANTIDADES DE DADOS.

NEURÔNIO BIOLÓGICO X ARTIFICIAL



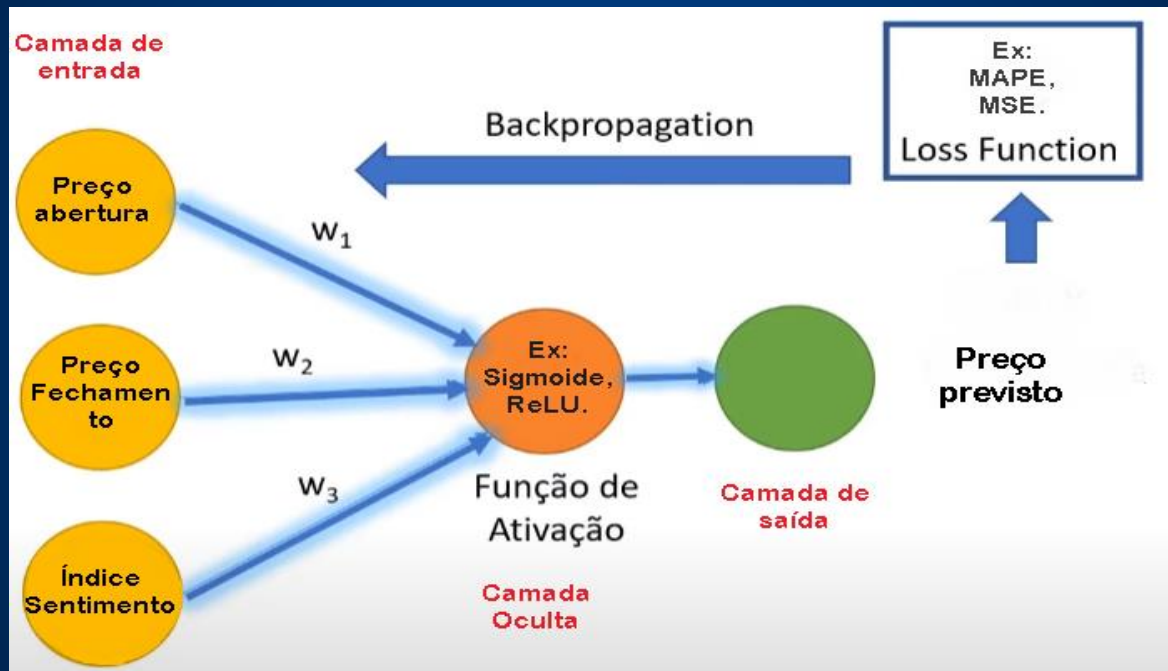
Fonte: [\(PDF\) Inteligência computacional aplicada na geração de respostas impulsivas bi-auriculares e em aurilização de salas \(researchgate.net\)](#)

REDE NEURAL MULTILAYER PERCEPTRON



Fonte: [O que são Redes Neurais? Importância e Como Funciona? \(opencadd.com.br\)](http://opencadd.com.br)

APRENDIZADO – REDES NEURAIS



Fonte: Adaptado de <https://youtu.be/mWD8wWwZpi8>

PREVISÃO DE SÉRIES TEMPORAIS

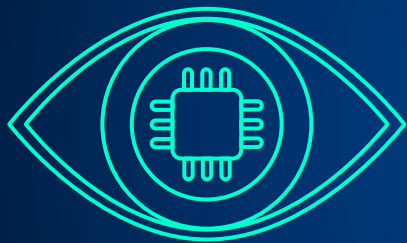
- ❑ SÉRIES TEMPORAIS: CONJUNTO DE OBSERVAÇÕES FEITAS SEQUENCIALMENTE AO DECORRER DE UM PERÍODO DE TEMPO;
- ❑ REPRESENTAM DADOS QUE MUDAM AO DECORRER DO TEMPO;
- ❑ EXEMPLOS DE PREVISÕES: PREVISÃO DO CLIMA, PREVISÃO DE PREÇOS DE ATIVOS, PREVISÃO DE VENDAS, ETC;
- ❑ OBJETIVO: EXTRAIR PADRÕES DE UM CONJUNTO DE DADOS OBSERVADOS EM UM PERÍODO DE TEMPO PASSADO, PARA PREVER COMPORTAMENTOS FUTUROS.

CRIPTOATIVOS

- ❑ ATIVOS DIGITAIS QUE UTILIZAM A CRIPTOGRAFIA E A TECNOLOGIA BLOCKCHAIN PARA MANTER SUAS TRANSAÇÕES SEGURAS E INVOLÁVEIS DE FORMA DESCENTRALIZADA;
- ❑ PRINCIPAL CLASSE DE ATIVOS PRESENTES NA WEB 3.0;
- ❑ EXEMPLOS: BITCOIN, ETHEREUM, SOLANA, DOGECOIN;

TENSORFLOW

- ❑ BIBLIOTECA *OPEN SOURCE* PARA CONSTRUÇÃO DE MODELOS DE *MACHINE LEARNING* PARA COMPUTADORES, *WEB* E DISPOSITIVOS MÓVEIS;
- ❑ POSSUI UMA API DE ALTO NÍVEL (KERAS) QUE PERMITE CONSTRUIR MODELOS DE *MACHINE LEARNING* DE FORMA ÁGIL E SIMPLES;
- ❑ BIBLIOTECAS SIMILARES: SCIKIT-LEARN, PYTORCH.



02

PREPARO DO AMBIENTE

BIBLIOTECAS

- ❑ tensorflow
- ❑ numpy
- ❑ yfinance
- ❑ pandas
- ❑ matplotlib
- ❑ ipympl
- ❑ scikit-learn

HELLO-WORLD DAS REDES NEURAIS

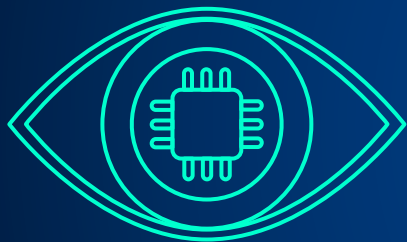
- ❑ PRÁTICAR NO JUPYTER-LAB;
- ❑ CONSTRUIR UM MODELO QUE ENCONTRE O PADRÃO NOS DADOS ROTULADOS ABAIXO E FAÇA AS PREVISÕES PARA NOVOS DADOS DE ENTRADA:

$X = [-1.0, 0.0, 1.0, 2.0, 3.0, 4.0]$

$Y = [-3.0, -1.0, 1.0, 3.0, 5.0, 7.0]$

X: REPRESENTA OS DADOS DE ENTRADA DO MODELO .

Y: REPRESENTA OS VALORES DE SAÍDA DO MODELO.



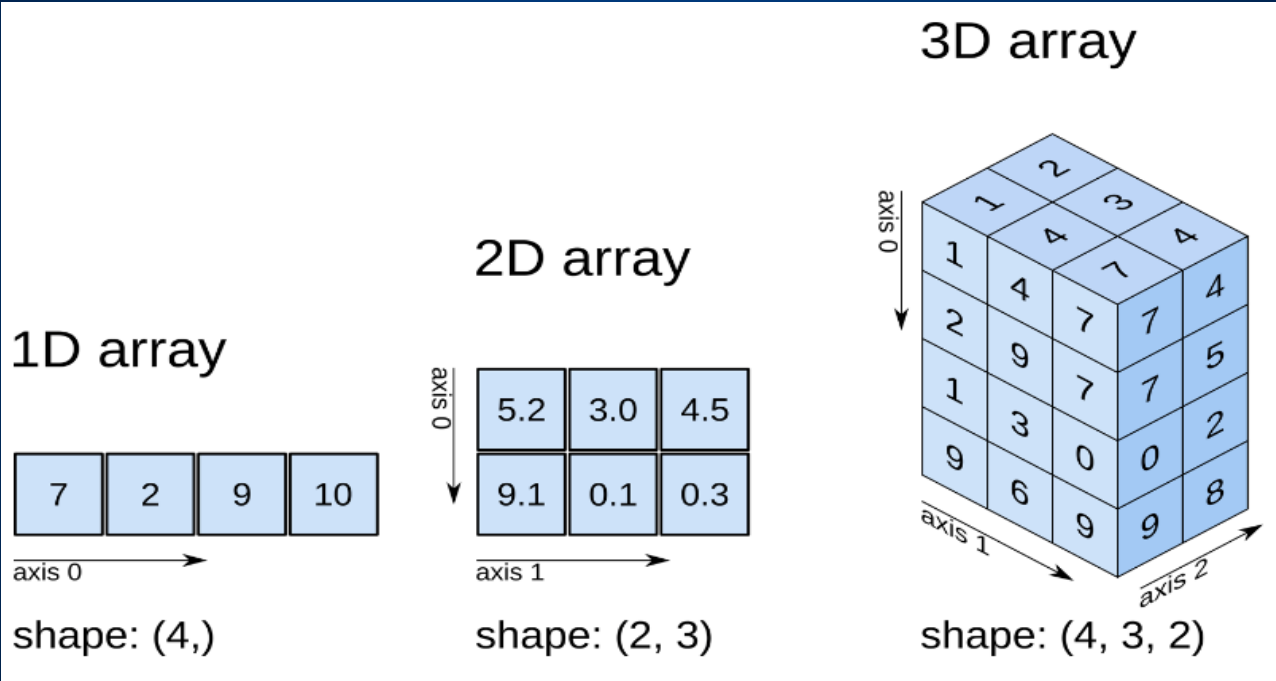
03

PRÉ- PROCESSAMENTO DOS DADOS

REDIMENSIONAMENTO DOS DADOS

- ❑ OS MODELOS PRECISAM QUE OS DADOS ESTEJAM ESTRUTURADOS NO FORMATO DE ARRAYS/MATRIZES;
- ❑ SHAPE: FORMATO (DIMENSÃO) DOS DADOS EM ARRAYS/MATRIZES;
- ❑ RESHAPE: TÉCNICA PARA REDIMENSIONAR OS DADOS EM ARRAYS/MATRIZES;
- ❑ AO CONSTRUIR OS MODELOS, NA CAMADA DE ENTRADA É PRECISO ESPECIFICAR O SHAPE DA MATRIZ DOS DADOS DE ENTRADA E FORNCÊ-LOS NO FORMATO ESPERADO PELA REDE NEURAL.

SHAPE EM ARRAYS



Fonte: [python - Numpy's "shape" function returns a 1D value for a 2D array - Stack Overflow](#)

COLETA DE DADOS

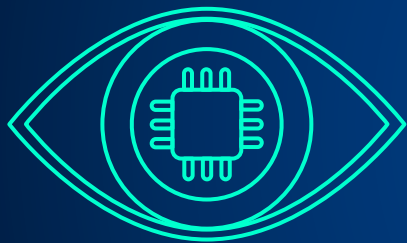
- ❑ UTILIZAR A BIBLIOTECA YFINANCE PARA COLETAR OS DADOS DE PREÇOS DO CRIPTOATIVO BITCOIN;
- ❑ OS DADOS DA BIBLIOTECA YFINANCE SÃO RETORNADOS NO FORMATO DE UM DATAFRAME DE DUAS DIMENSÕES;
- ❑ UTILIZAR MÉTODOS .INFO, .HEAD, .TAIL PARA VISUALIZAR AS INFORMAÇÕES DO DATAFRAME E AMOSTRAS DOS DADOS.

DIVISÃO DOS DADOS

- ❑ DIVIDIR OS DADOS EM DOIS CONJUNTOS, UM PARA TREINO (90%) E OUTRO PARA TESTES (10%). UTILIZAR O MÉTODO `TRAIN_TEST_SPLIT` DA BIBLIOTECA `SCIKIT-LEARN`;
- ❑ O OBJETIVO É TREINAR O MODELO COM UM CONJUNTO DE DADOS, E TESTÁ-LO COM OUTROS DADOS QUE ELE NÃO TEVE ACESSO NO TREINAMENTO;

NORMALIZAÇÃO DOS DADOS

- ❑ TRANSFORMA AS DIFERENTES ESCALAS DE VALORES DAS *FEATURES* EM UM INTERVALO ESPECÍFICO DEFINIDO ENTRE 0 E 1 OU ENTRE -1 E 1. UTILIZAR A CLASSE *MINMAXSCALER* DA BIBLIOTECA *SCIKIT-LEARN*;
- ❑ BOA PRÁTICA PARA REALIZAR O TREINAMENTO DE REDES NEURAIS, POIS ESTAS TRABALHAM INTERNAMENTE COM PEQUENOS VALORES EM SEUS PARÂMETROS;
- ❑ APÓS O TREINAMENTO, É PRECISO FAZER A DESNORMALIZAÇÃO DOS DADOS E VOLTÁ-LOS PARA A ESCALA ORIGINAL DE VALORES.



04

CONSTRUÇÃO DOS MODELOS

CONSTRUÇÃO: MODELO ANN (MLP)

- ❑ UTILIZAR API KERAS DO TENSORFLOW;
- ❑ UTILIZAR UMA CAMADA DENSA COM 64 NEURÔNIOS;
- ❑ FUNÇÃO DE ATIVAÇÃO: TANGENTE HIPERBÓLICA (TANH);
- ❑ INPUT_SHAPE: MATRIZ DE DADOS COM 5 FEATURES (OPEN, HIGH, LOW, VOLUME, CLOSE);
- ❑ UTILIZAR UMA CAMADA DE SAÍDA DO TIPO DENSA COM 1 NEURÔNIO (SAÍDA É A PREVISÃO PARA O PRÓXIMO DIA);
- ❑ UTILIZAR O ALGORITMO OTIMIZADOR ADAM;
- ❑ COMPILAR O MODELO COM A FUNÇÃO DE PERDA MAE (MEAN ABSOLUTE ERROR)

CONSTRUÇÃO: MODELO ANN (MLP)

```
model = keras.Sequential(name='ANN_ONE-STEP-FORECAST')
model.add(layers.Dense(units=64, activation='tanh', input_shape=(x_train.shape[1],))) # Funções de ativação: sigmoid, tanh, relu,
model.add(layers.Dense(1))

# Define o algoritmo otimizador dos pesos da rede neural e a taxa de aprendizado
opt = keras.optimizers.Adam() # learning_rate=0.001 por padrão

# Compila o modelo definindo a função de perda para cálculo do erro e o algoritmo otimizador.
model.compile(loss='mae', optimizer=opt) # Funções de perda para testar: mae, mse, msle, huber, log_cosh
```

Fonte: Elaboração própria.

TREINAMENTO: MODELO ANN (MLP)

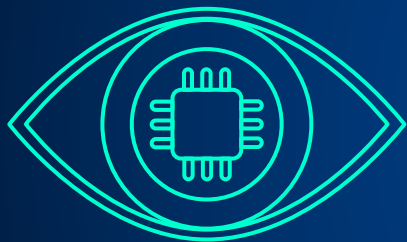
- ❑ TREINAR O MODELO POR MEIO DO MÉTODO FIT;
- ❑ INFORMAR OS DADOS ROTULADOS (TREINAMENTO E TESTES);
- ❑ TREINAR POR 100 ÉPOCAS;
- ❑ DEFINIR O BATCH_SIZE COM UM VALOR DE 32;
- ❑ DEFINIR O PARÂMETRO SHUFFLE = TRUE

TREINAMENTO: MODELO ANN (MLP)

```
%%time  
  
# Treina o modelo  
history = model.fit(x_train, y_train, epochs=100, batch_size=32, verbose=1, shuffle=True, validation_data=(x_test, y_test))
```

Fonte: Elaboração própria.

- ❑ AVALIAR O ERRO DO TREINAMENTO COM A BIBLIOTECA MATPLOTLIB.



05

TESTES DOS MODELOS

TESTES: MODELO ANN (MLP)

- ❑ OS TESTES DE PREVISÃO SÃO FEITOS A PARTIR DOS DADOS DE TESTES QUE NÃO FORAM CONHECIDOS PELO MODELO DURANTE O TREINAMENTO;
- ❑ FAZER PREVISÃO POR MEIO DO MÉTODO “PREDICT”, FORNECENDO OS DADOS DE ENTRADA DO CONJUNTO DE TESTES;
- ❑ DESNORMALIZAR OS VALORES PREVISTOS E TRANSFORMÁ-LOS PARA A ESCALA ORIGINAL;
- ❑ DESNORMALIZAR OS VALORES REAIS E TRANSFORMÁ-LOS PARA A ESCALA ORIGINAL E COMPARAR COM OS VALORES PREVISTOS.

TESTES: MODELO ANN (MLP)

```
# Função que transforma os dados normalizados pelo scaler para a escala original
def transform_orignal_scale(array_data):
    scaler_min_value = scaler.feature_range[0] # Obtém o menor valor do scaler
    scaler_max_value = scaler.feature_range[1] # Obtém o maior valor do scaler
    original_data_min = data_train[['Close']].values.min(axis=0)
    original_data_max = data_train[['Close']].values.max(axis=0)

    return (array_data - scaler_min_value) / (scaler_max_value - scaler_min_value) * (original_data_max - original_data_min) + original_data_min
```

Fonte: Elaboração própria.

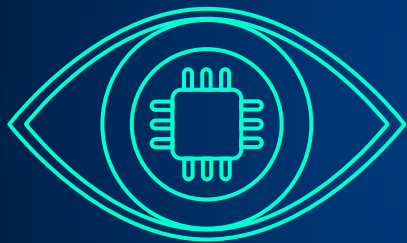
TESTES: MODELO ANN (MLP)

```
# Teste de predição a partir dos dados de testes  
predictions = model.predict(x_test)
```

```
# Desnormalização dos dados de predição (transforma para a escala original)  
predictions = transform_orignal_scale(predictions)  
print(predictions.shape)  
print(predictions[-1:])
```

```
# Desnormalização dos valores reais (transforma para a escala original)  
real_prices = transform_orignal_scale(y_test)  
print(real_prices.shape)  
print(real_prices[-1:])
```

Fonte: Elaboração própria.



06

CONCLUSÃO

VISUALIZAÇÃO DOS RESULTADOS

- ❑ COMPARAR OS VALORES PREVISTOS COM OS VALORES REAIS A PARTIR DO CONJUNTO DE DADOS DE TESTES;
- ❑ UTILIZAR A BIBLIOTECA MATPLOTLIB PARA CRIAR O GRÁFICO;
- ❑ UTILIZAR A MÉTRICA MAPE (MEAN ABSOLUTE PERCENTAGE ERROR) PARA MEDIR O ERRO DO MODELO;

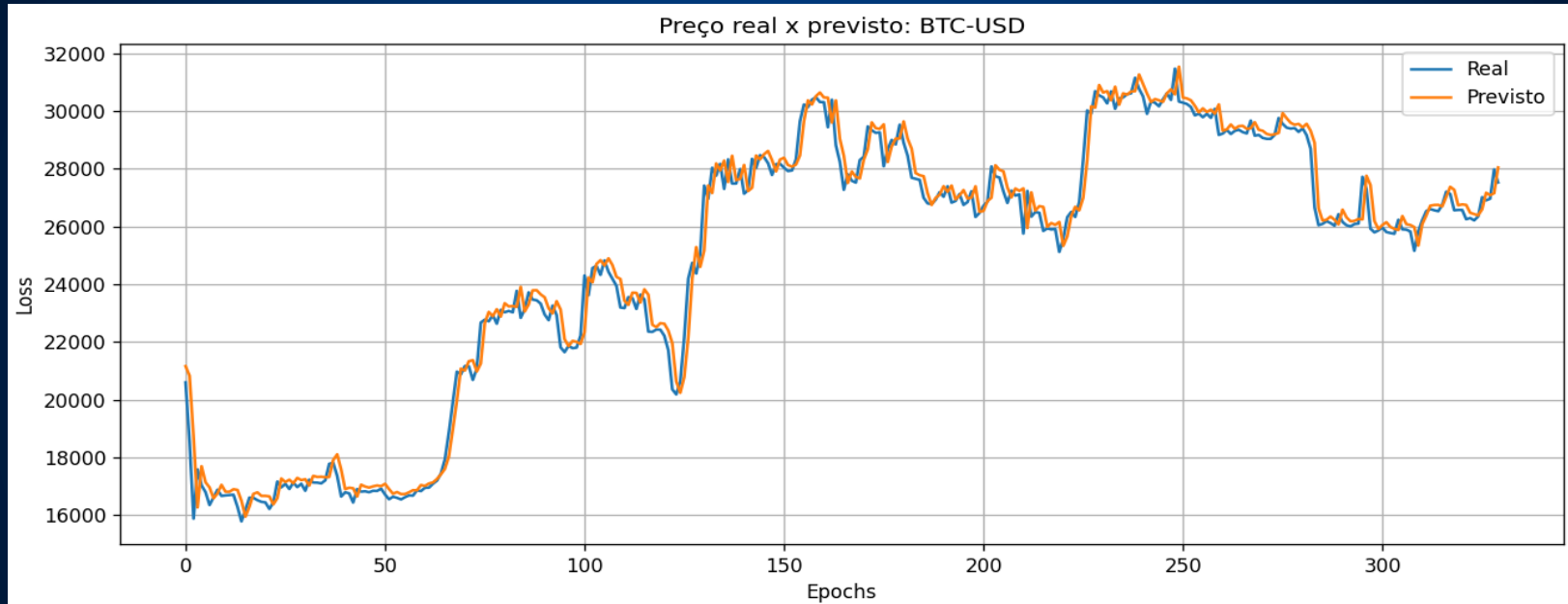
VISUALIZAÇÃO DOS RESULTADOS

```
# Plota o gráfico comparativo entre os valores reais x previstos
epochs = range(len(y_test))
plt.figure(figsize=(13, 5))
plt.plot(epochs, real_prices, label='Real')
plt.plot(epochs, predictions, label='Previsto')
plt.title(f'Preço real x previsto: {SYMBOL}')
plt.xlabel('Epochs')
plt.ylabel('Loss')
#plt.semilogy()
plt.legend()
plt.grid()

plt.show()
```

Fonte: Elaboração própria.

VISUALIZAÇÃO DOS RESULTADOS



Fonte: Elaboração própria.

AVALIAÇÃO DOS RESULTADOS: MAPE

```
# Avaliação da métrica MAPE
mape = tf.keras.metrics.MeanAbsolutePercentageError()
mape.update_state(real_prices, predictions)
mape_result = mape.result().numpy()

print(f'MAPE: {mape_result}')
```

MAPE: 1.7494179010391235

Fonte: Elaboração própria.

- ❑ REPRESENTA O ERRO MÉDIO ABSOLUTO EM PERCENTUAL DE TODAS AS PREVISÕES FEITAS A PARTIR DO CONJUNTO DE TESTES.

É POSSÍVEL MELHORAR?

- ❑ UTILIZAR ARQUITETURAS DE REDES NEURAIS MAIS ROBUSTAS PARA PROBLEMAS DE SÉRIES TEMPORAIS: RNN, LSTM E ARQUITETURAS HÍBRIDAS;
- ❑ TESTAR DIFERENTES PARÂMETROS DO MODELO: NUMERO DE NEURÔNIOS, QUANTIDADE DE CAMADAS OCULTAS, FUNÇÕES DE PERDA, ALGORITMOS OTIMIZADORES, REGULARIZADORES, BATCH_SIZE, ÉPOCAS, ETC;
- ❑ ENGENHARIA DE DADOS PARA MELHORAR A SELEÇÃO DE FEATURES E O PRÉ-PROCESSAMENTO DOS DADOS.

REFERÊNCIAS

- M. Swamynathan. *Mastering Machine Learning with Python in Six Steps: A Practical Implementation Guide to Predictive Data Analytics Using Python*. Apress, 2019.
- J. Patterson and A. Gibson. *Deep Learning: A Practitioner's Approach*. O'Reilly Media, 2017.
- C. Burniske and J. Tatar. *Cryptoassets: The Innovative Investor's Guide to Bitcoin and Beyond*. McGraw-Hill Education, 2017.
- [Previsão de séries temporais | TensorFlow Core](#)
- [Time Series Forecasting as Supervised Learning - MachineLearningMastery.com](#)
- [scikit-learn: machine learning in Python — scikit-learn 1.3.1 documentation](#)
- [Keras: Deep Learning for humans](#)
- [yfinance · PyPI](#)
- [NumPy](#)
- [pandas - Python Data Analysis Library \(pydata.org\)](#)
- [Matplotlib — Visualization with Python](#)