JOÃO LUIZ PACHER

MBA em Ciência de Dados

Redes Neurais e Arquiteturas Profundas

Análise de Dados com Base em Processamento Massivo em Paralelo

Prova Final

Material Produzido por:

> **Profa. Dra. Cristina Dutra de Aguiar**
> **Prof. Dr. Moacir A. Ponti**

CEMEAI - ICMC/USP São Carlos

A prova final contém 1 questão, dividida em 3 itens. Por favor, procurem por Questão para encontrar a especificação da questão e por RESOLVER para encontrar a especificação do item a ser solucionado. Também é possível localizar a questão e os itens utilizando o menu de navegação.

O notebook contém a constelação de fatos da BI Solutions que deve ser utilizada para responder à questão e também todas as `bibliotecas`, `bases de dados`, `inicializações`, `instalações`, `importações`, `geração de dataFrames`, `geração de visões temporárias` e `conversão dos tipos de dados` necessárias para a realização da questão.

INSTRUÇÕES:
1) Você deve exportar esse notebook com sua solução para as questões da prova em formato .py e fazer upload no Moodle. Atenção: você não deve fazer upload de um arquivo notebook (.ipynb), mas sim um arquivo texto .py contendo os códigos python que utilizou para resolver as questões. O arquivo .py pode ser gerado através da opção:
File --> Download as --> Python (.py) disponível no Jupyter Notebook.

ou File --> Download .py no Google Colab

Caso não esteja utilizando o Jupyter, copie e cole seu código em um arquivo ASCII (Texto) salvando com a extensão .py

2) Você deve salvar esse notebook com sua solução para as questões da prova em formato .pdf e fazer upload no Moodle

3) Os arquivos devem ser nomeados com seu nome e sobrenome, sem espaços. Exemplo: moacirponti.py e moacirponti.pdf

4) É OBRIGATÓRIO conter no cabeçalho (início) do arquivo um comentário / texto com o seu nome completo

Desejamos uma boa prova!

1 Constelação de Fatos da BI Solutions

A aplicação de *data warehousing* da BI Solutions utiliza como base uma contelação de fatos, conforme descrita a seguir.

Tabelas de dimensão

- data (dataPK, dataCompleta, dataDia, dataMes, dataBimestre, dataTrimestre, dataSemestre, dataAno)
- funcionario (funcPK, funcMatricula, funcNome, funcSexo, funcDataNascimento, funcDiaNascimento, funcMesNascimento, funcAnoNascimento, funcCidade, funcEstadoNome, funcEstadoSigla, funcRegiaoNome, funcRegiaoSigla, funcPaisNome, funcPaisSigla)
- equipe (equipePK, equipeNome, filialNome, filialCidade, filialEstadoNome, filialEstadoSigla, filialRegiaoNome, filialRegiaoSigla, filialPaisNome, filialPaisSigla)
- cargo (cargoPK, cargoNome, cargoRegimeTrabalho, cargoEscolaridadeMinima, cargoNivel)
- cliente (clientePK, clienteNomeFantasia, clienteSetor, clienteCidade, clienteEstadoNome, clienteEstadoSigla, clienteRegiaoNome, clienteRegiaoSigla, clientePaisNome, clientePaisSigla)

Tabelas de fatos

- pagamento (dataPK, funcPK, equipePK, cargoPK, salario, quantidadeLancamentos)
- negociacao (dataPK, equipePK, clientePK, receita, quantidadeNegociacoes)

2 Configurações

2.1 Obtenção dos Dados da BI Solutions

```
#instalando o módulo wget
%%capture
!pip install -q wget
!mkdir data

#baixando os dados das tabelas de dimensão e das tabelas de fatos
import wget

url = "https://raw.githubusercontent.com/cdaciferri/DataMartBISolutions/main/data.csv"
wget.download(url, "data/data.csv")
```

```
url = "https://raw.githubusercontent.com/cdaciferri/DataMartBISolutions/main/funcionario.c
wget.download(url, "data/funcionario.csv")

url = "https://raw.githubusercontent.com/cdaciferri/DataMartBISolutions/main/equipe.csv"
wget.download(url, "data/equipe.csv")

url = "https://raw.githubusercontent.com/cdaciferri/DataMartBISolutions/main/cargo.csv"
wget.download(url, "data/cargo.csv")

url = "https://raw.githubusercontent.com/cdaciferri/DataMartBISolutions/main/cliente.csv"
wget.download(url, "data/cliente.csv")

url = "https://raw.githubusercontent.com/cdaciferri/DataMartBISolutions/main/pagamento.csv
wget.download(url, "data/pagamento.csv")

url = "https://raw.githubusercontent.com/cdaciferri/DataMartBISolutions/main/negociacao.cs
wget.download(url, "data/negociacao.csv")
```

2.2 Instalações e Inicializações

```
#instalando Java Runtime Environment (JRE) versão 8
%%capture
!apt-get remove openjdk*
!apt-get update --fix-missing
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
```

```
#baixando Apache Spark versão 3.0.0
%%capture
!wget -q https://archive.apache.org/dist/spark/spark-3.0.0/spark-3.0.0-bin-hadoop2.7.tgz
!tar xf spark-3.0.0-bin-hadoop2.7.tgz && rm spark-3.0.0-bin-hadoop2.7.tgz
```

```
import os
#configurando a variável de ambiente JAVA_HOME
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
#configurando a variável de ambiente SPARK_HOME
os.environ["SPARK_HOME"] = "/content/spark-3.0.0-bin-hadoop2.7"
```

```
%%capture
#instalando o pacote findspark
!pip install -q findspark==1.4.2
#instalando o pacote pyspark
!pip install -q pyspark==3.0.0
```

2.3 Bibliotecas

```
import findspark
findspark.init()
```

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("pyspark-notebook").master("local[*]").getOrCreate()

from pyspark.sql.types import IntegerType
from pyspark.sql.types import FloatType
from pyspark.sql.functions import round, desc

import random
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from numpy.random import seed
from tensorflow.random import set_seed
from tensorflow import keras
from tensorflow.keras import layers
```

2.4 Geração dos DataFrames em Pandas da BI Solutions

Nesta seção são gerados os DataFrames em Pandas. Atenção aos nomes desses DataFrames.

```
pd.set_option('display.float_format', lambda x: '%.2f' % x)

cargoPandas = pd.read_csv('https://raw.githubusercontent.com/cdaciferri/DataMartBISolutior
clientePandas = pd.read_csv('https://raw.githubusercontent.com/cdaciferri/DataMartBISoluti
dataPandas = pd.read_csv('https://raw.githubusercontent.com/cdaciferri/DataMartBISolutions
equipePandas = pd.read_csv('https://raw.githubusercontent.com/cdaciferri/DataMartBISolutic
funcionarioPandas = pd.read_csv('https://raw.githubusercontent.com/cdaciferri/DataMartBISc
negociacaoPandas = pd.read_csv('https://raw.githubusercontent.com/cdaciferri/DataMartBISol
pagamentoPandas = pd.read_csv('https://raw.githubusercontent.com/cdaciferri/DataMartBISolu
```

2.5 Geração dos DataFrames em Spark da BI Solutions

Nesta seção são gerados dos DataFrames em Spark. Atenção aos nomes desses DataFrames.

```
#criando os DataFrames em Spark
cargo = spark.read.csv(path="data/cargo.csv", header=True, sep=",")
cliente = spark.read.csv(path="data/cliente.csv", header=True, sep=",")
data = spark.read.csv(path="data/data.csv", header=True, sep=",")
equipe = spark.read.csv(path="data/equipe.csv", header=True, sep=",")
funcionario = spark.read.csv(path="data/funcionario.csv", header=True, sep=",")
negociacao = spark.read.csv(path="data/negociacao.csv", header=True, sep=",")
pagamento = spark.read.csv(path="data/pagamento.csv", header=True, sep=",")

#convertendo os dados necessários para o tipo de dado inteiro
colunas_cargo = ["cargoPK"]
colunas_cliente = ["clientePK"]
colunas_data = ["dataPK", "dataDia", "dataMes", "dataBimestre", "dataTrimestre", "dataSeme
```

```
colunas_equipe = ["equipePK"]
colunas_funcionario = ["funcPK", "funcDiaNascimento", "funcMesNascimento", "funcAnoNascime
colunas_negociacao = ["equipePK", "clientePK", "dataPK", "quantidadeNegociacoes"]
colunas_pagamento = ["funcPK", "equipePK", "dataPK", "cargoPK", "quantidadeLancamentos"]

for coluna in colunas_cargo:
  cargo = cargo.withColumn(coluna, cargo[coluna].cast(IntegerType()))

for coluna in colunas_cliente:
  cliente = cliente.withColumn(coluna, cliente[coluna].cast(IntegerType()))

for coluna in colunas_data:
  data = data.withColumn(coluna, data[coluna].cast(IntegerType()))

for coluna in colunas_equipe:
  equipe = equipe.withColumn(coluna, equipe[coluna].cast(IntegerType()))

for coluna in colunas_funcionario:
  funcionario = funcionario.withColumn(coluna, funcionario[coluna].cast(IntegerType()))

for coluna in colunas_negociacao:
  negociacao = negociacao.withColumn(coluna, negociacao[coluna].cast(IntegerType()))

for coluna in colunas_pagamento:
  pagamento = pagamento.withColumn(coluna, pagamento[coluna].cast(IntegerType()))

#convertendo os dados necessários para o tipo de dado float
colunas_negociacao = ["receita"]
colunas_pagamento = ["salario"]

for coluna in colunas_negociacao:
  negociacao = negociacao.withColumn(coluna, negociacao[coluna].cast(FloatType()))

for coluna in colunas_pagamento:
  pagamento = pagamento.withColumn(coluna, pagamento[coluna].cast(FloatType()))


#criando as visões temporárias
cargo.createOrReplaceTempView("cargo")
cliente.createOrReplaceTempView("cliente")
data.createOrReplaceTempView("data")
equipe.createOrReplaceTempView("equipe")
funcionario.createOrReplaceTempView("funcionario")
negociacao.createOrReplaceTempView("negociacao")
pagamento.createOrReplaceTempView("pagamento")
```

3 Questão

A empresa BI Solutions está realizando uma investigação baseada no projeto e treinamento de uma rede neural, usando como base seus dados históricos mantidos na Constelação de Fatos. O modelo resultante deve ser usado para obter uma predição, a qual é voltada à análise da investigação esperada.

IMPORTANTE: Leia a questão com muita atenção, desde que vários passos da questão já se encontram implementados. Os locais nos quais os comandos de resposta para os itens da questão devem ser especificados são identificados em comentários.

3.1 Investigação dos Dados Históricos

A primeira parte para solucionar a investigação consiste na obtenção de dados históricos de interesse. Isso deve feito por meio da especificação de uma consulta OLAP, segundo as instruções detalhadas a seguir.

3.1.1 Instruções para a Especificação da Consulta OLAP

- A **consulta OLAP** pode ser especificada usando qualquer uma das três opções a seguir (escolha SOMENTE UMA forma)
  - Usando **Pandas** (conceitos apresentados na Aula 05).
  - Usando a **linguagem SQL** (conceitos apresentados na Aula 07).
  - Usando os métodos de **pyspark.sql** (conceitos apresentados na Aula 08).
- Na listagem das respostas:
  - As **colunas** solicitadas devem ser exibidas exatamente na mesma ordem que a definida.
  - As **linhas** retornadas como respostas devem ser exibidas exatamente na mesma ordem que a definida.
  - Os **nomes das colunas** renomeadas devem seguir estritamente os nomes definidos.

3.1.2 Consulta OLAP (RESOLVER)

Liste, para cada código da equipe, setor do cliente e nome do estado do cliente, a soma das receitas. Arredonde a soma das receitas para até duas casas decimais. Devem ser exibidas as colunas na ordem e com os nomes especificados a seguir: "CODIGOEQUIPE", "SETORCLIENTE", "NOMEESTADOCLIENTE", "TOTALRECEITA". Ordene as linhas exibidas primeiro pelo total de receita em ordem ascendente, depois pelo código da equipe em ordem ascendente, depois pelo setor do cliente em ordem ascendente, depois pelo nome do estado em ordem ascendente.

```
# Escreva aqui a sua resposta para a consulta OLAP
querySQL = """
SELECT equipePK AS CODIGOEQUIPE,
       clienteSetor AS SETORCLIENTE,
       clienteEstadoNome AS NOMEESTADOCLIENTE,
       ROUND(SUM(receita), 2) As TOTALRECEITA
FROM negociacao JOIN cliente ON negociacao.clientePK = cliente.clientePK
```

```
GROUP BY equipePK, clienteSetor, clienteEstadoNome
ORDER BY TOTALRECEITA ASC, equipePK ASC, clienteSetor ASC, clienteEstadoNome ASC
"""
spark.sql(querySQL).show(25,truncate=False)
```

```
+------------+------------------+------------------+------------+
|CODIGOEQUIPE|SETORCLIENTE      |NOMEESTADOCLIENTE |TOTALRECEITA|
+------------+------------------+------------------+------------+
|3           |SAUDE             |CEARA             |11966.6     |
|4           |CREDITO           |MATO GROSSO DO SUL|22399.35    |
|3           |CREDITO           |AMAZONAS          |22440.2     |
|8           |BEBIDAS E ALIMENTOS|CEARA             |24046.1     |
|5           |SAUDE             |SANTA CATARINA    |26919.95    |
|6           |SAUDE             |RIO GRANDE DO SUL |32943.35    |
|3           |TECNOLOGIA        |MATO GROSSO DO SUL|33314.45    |
|7           |BEBIDAS E ALIMENTOS|PERNAMBUCO        |37847.2     |
|4           |CREDITO           |SAO PAULO         |40457.8     |
|5           |BEBIDAS E ALIMENTOS|CEARA             |42554.95    |
|4           |VESTUARIO         |MINAS GERAIS      |42796.1     |
|5           |CREDITO           |MATO GROSSO DO SUL|43237.35    |
|5           |SAUDE             |CEARA             |43899.4     |
|3           |BEBIDAS E ALIMENTOS|CEARA             |44198.6     |
|4           |BEBIDAS E ALIMENTOS|PARANA            |45200.0     |
|5           |CREDITO           |AMAZONAS          |45336.6     |
|5           |VESTUARIO         |SAO PAULO         |48322.5     |
|3           |BEBIDAS E ALIMENTOS|PERNAMBUCO        |49501.2     |
|3           |VESTUARIO         |MINAS GERAIS      |49724.7     |
|8           |SAUDE             |RIO GRANDE DO SUL |52748.1     |
|4           |TECNOLOGIA        |AMAZONAS          |59102.4     |
|3           |BEBIDAS E ALIMENTOS|PARANA            |60093.05    |
|5           |CREDITO           |SAO PAULO         |61647.05    |
|4           |BEBIDAS E ALIMENTOS|PERNAMBUCO        |63210.1     |
|3           |CREDITO           |SAO PAULO         |65127.9     |
+------------+------------------+------------------+------------+
only showing top 25 rows
```

```
# Caso tenha feito a sua resposta usando a linguagem SQL
# querySQL = """ comando SQL """
# Transforme o resultado da consulta, o qual encontra-se em "querySQL",
# em um dataFrame em Pandas descomentando o comando a seguir:

df = spark.sql(querySQL).toPandas()


# Caso tenha feito a sua resposta usando os métodos de pyspark.sql
# dataFramepyspark = resposta em pyspark sem finalizar com o método .show()
# Transforme o resultado da consulta, o qual encontra-se
# em "dataFramepyspark", em um dataFrame em Pandas
# descomentando o comando a seguir:

# df = dataFramepyspark.toPandas()


# Em qualquer caso, exiba algumas linhas do dataFrame de nome df,
# o qual é um dataFrame em Pandas, descomentando o comando a seguir:
```

```
df
```

	CODIGOEQUIPE	SETORCLIENTE	NOMEESTADOCLIENTE	TOTALRECEITA
0	3	SAUDE	CEARA	11966.60
1	4	CREDITO	MATO GROSSO DO SUL	22399.35
2	3	CREDITO	AMAZONAS	22440.20
3	8	BEBIDAS E ALIMENTOS	CEARA	24046.10
4	5	SAUDE	SANTA CATARINA	26919.95
...	...	...	...	...
198	9	BEBIDAS E ALIMENTOS	SAO PAULO	2122377.28
199	9	VESTUARIO	RIO DE JANEIRO	2243123.71
200	10	BEBIDAS E ALIMENTOS	SAO PAULO	2751213.15
201	10	TECNOLOGIA	SAO PAULO	3326962.29
202	9	TECNOLOGIA	SAO PAULO	4007617.34

203 rows × 4 columns

```
# Setor mais frequente dentre os 8 primeiros que aparecem na resposta da consulta OLAP
df[0:8]['SETORCLIENTE'].value_counts().idxmax()
```

```
'SAUDE'
```

Note que o dataFrame em Pandas, chamado `df`, é o dataFrame a ser utilizado para o treinamento da rede neural.

3.2 Treinamento da Rede Neural

A segunda parte para solucionar a investigação consiste no treinamento da rede neural usando como base os dados históricos obtidos no item 3.1.2, segundo as instruções detalhadas a seguir.

3.2.1 Preparação dos Dados

A preparação dos dados já encontra-se pronta, sendo necessário apenas executar as células.

```
# ordena dados
df = df.sort_values(by='CODIGOEQUIPE')

# obtem dummy-variables / one-hot-encoding
setor = pd.get_dummies(df['SETORCLIENTE'])
estado = pd.get_dummies(df['NOMEESTADOCLIENTE'])
equipe = pd.get_dummies(df['CODIGOEQUIPE'])
```

```
# cria dataframe para treinamento
input_data = pd.concat([equipe, setor, estado, df['TOTALRECEITA']], axis=1, sort=False)


# separa dados de treinamento e 1 para teste
n_test = 1

x_train = np.array(input_data.iloc[:-n_test,:-1])
x_test = np.array(input_data.iloc[-n_test:,:-1])

y_train = np.array(input_data.iloc[:-n_test,-1])
y_test = np.array(input_data.iloc[-n_test,-1])

print("Tamanho conjunto de treinamento: ", x_train.shape)
print("Tamanho conjunto de testes: ", x_test.shape)
```

```
Tamanho conjunto de treinamento:  (202, 25)
Tamanho conjunto de testes:  (1, 25)
```

```
# valor maximo para normalizar receita
y_max = np.max(y_train)
print('Max train', y_max)
```

```
Max train 4007617.34
```

```
# dados para regressao normalizados
y_train = np.array(input_data.iloc[:-n_test,-1])/y_max
y_test = np.array(input_data.iloc[-n_test:,-1])/y_max
```

3.2.2 Projeto e Treinamento de Rede Neural (RESOLVER)

Utilize o código a seguir e, em seguida, compile e treine a rede neural utilizando:

- função de custo de erro médio quadrático,
- learning rate inicial de 0.005 e decaimento dado pela função `scheduler` provida,
- 30 épocas,
- batchsize de tamanho 10.

```
def deep_net(neurons, input_dim, n_layers=1, batch_norm=False, dropout_rate=0.0):

    input_data = keras.layers.Input(shape=(input_dim,))

    x = keras.layers.Activation('relu')(input_data)
    if batch_norm:
        x = keras.layers.BatchNormalization(name='bn_input')(x)
        x = keras.layers.Activation('relu')(x)
    x = keras.layers.Dense(neurons, activation='relu')(x)
    x = keras.layers.Dense(neurons, activation='relu')(x)
    x = keras.layers.Dropout(dropout_rate)(x)
    output = keras.layers.Dense(1, activation='relu')(x)
```

```
    dnn = keras.models.Model(input_data, output)

    return dnn


def scheduler(epoch, lr):
    if epoch < 5: return lr
    return lr * tf.math.exp(-0.05)

callbacklr = tf.keras.callbacks.LearningRateScheduler(scheduler)


seed(1)
set_seed(2)
dnn = deep_net(64, x_train.shape[1], batch_norm=False, dropout_rate=0.5)


# Escreva aqui a sua resposta para a compilação e treinamento de rede neural

# compilar
dnn.compile(
    optimizer=keras.optimizers.Adam(0.005), loss="mse"
)

# treinar
batch_size=10
epochs=30
history = dnn.fit(x_train, y_train,
                  batch_size=batch_size,
                  epochs=epochs,
                  verbose=1,
                  callbacks=[callbacklr])
```

```
Epoch 2/30
21/21 [==============================] - 0s 3ms/step - loss: 0.0308 - lr: 0.0050
Epoch 3/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0307 - lr: 0.0050
Epoch 4/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0307 - lr: 0.0050
Epoch 5/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0303 - lr: 0.0050
Epoch 6/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0309 - lr: 0.0048
Epoch 7/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0296 - lr: 0.0045
Epoch 8/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0307 - lr: 0.0043
Epoch 9/30
21/21 [==============================] - 0s 3ms/step - loss: 0.0287 - lr: 0.0041
Epoch 10/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0304 - lr: 0.0039
Epoch 11/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0249 - lr: 0.0037
Epoch 12/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0284 - lr: 0.0035

Epoch 13/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0191 - lr: 0.0034
Epoch 14/30
```

```
21/21 [==============================] - 0s 2ms/step - loss: 0.0321 - lr: 0.0032
Epoch 15/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0251 - lr: 0.0030
Epoch 16/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0233 - lr: 0.0029
Epoch 17/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0192 - lr: 0.0027
Epoch 18/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0162 - lr: 0.0026
Epoch 19/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0155 - lr: 0.0025
Epoch 20/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0150 - lr: 0.0024
Epoch 21/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0153 - lr: 0.0022
Epoch 22/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0094 - lr: 0.0021
Epoch 23/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0079 - lr: 0.0020
Epoch 24/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0096 - lr: 0.0019
Epoch 25/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0093 - lr: 0.0018
Epoch 26/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0081 - lr: 0.0017
Epoch 27/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0067 - lr: 0.0017
Epoch 28/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0052 - lr: 0.0016
Epoch 29/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0043 - lr: 0.0015
Epoch 30/30
21/21 [==============================] - 0s 2ms/step - loss: 0.0044 - lr: 0.0014
```

3.2.3 Predição da receita (RESOLVER)

Utilizando o modelo treinado, realize a predição da receita para a linha separada para teste, i.e. a linha em `x_test`

Não se esqueça de, após obter a predição, reescalar novamente para o intervalo original dos dados, revertendo a normalização feita na etapa de preparação dos dados.

```
# Escreva aqui a sua resposta para a predição de receita em milhões de reais
y_pred = dnn.predict(x_test)[0]
y_pred = np.round(y_pred*y_max, 2)

print('Teste:', y_test[0]*y_max)
print('Predito:', y_pred[0])
print()
print('Predição em milhões de Reais', np.round(y_pred[0]/1000000,1))
```

```
Teste: 1245572.86
Predito: 1325050.4

Predição em milhões de Reais 1.3
```

✓ 0s conclusão: 16:41