Meu projeto de Deep Learning com Oracle Data Science

Erik Coelho Gama

23.03.2021

Oracle do Brasil
TDC


TDC
ONLINE
Sala
ORACLE
Inovação
com dados
em nuvem
MEU PROJETO DE DEEP
LEARNING COM ORACLE
DATA SCIENCE

Índice

Conteúdo

Introdução 4
Configurando o Ambiente 4
Criando Compartimento 4
Criando Grupo de Usuário 4
Criando Grupo Dinâmico 5
Criando Políticas para um Grupo de Usuário e Grupo Dinâmico. 5
Policy para o Grupo de Usuários 5
Policy para Grupo Dinâmico 5
Criando VCN Virtual Cloud Networking 5
Criando Projeto & Notebook no Data Science 6
Criando Projeto Data Science 6
Criando notebook data science 6
Acessando o Jupyter Notebook,Treinando e Deployando Modelo. 6
Criando Imagem Docker e Function 11
Criando Function no OCI 12
Criando API Gateway 13
Preenchendo informações básicas. 14
Preenchendo API politicas de Request 14
Políticas de Registro em Log da API. 14
Preenchendo Rotas 14
Enviando Request de predição 15
Resumo 16

Introdução

Olá!

Este documento foi construído durante o acontecimento do evento **TDC TheDevConf** 2020 Recife Online, e tem como objetivo ajudar as pessoas que assistiram a palestra de data science na pratica!

Todo processo que foi apresentado durante a apresentação no **TDC** estará disponível neste documento.

Boa sorte na aplicação!

Erik Gama.

Configurando o Ambiente

O primeiro passo é realizar algumas configurações administrativas para conseguirmos criar os nossos recursos na cloud, e principalmente nossa magnifica instância do Data Science.

Nesta etapa de configuração de ambiente criaremos alguns recursos em cloud como, nosso **compartimento**, **virtual cloud networking**, **grupo de usuário, grupo dinâmico**, **políticas, oracle function, api gateway** e a querida instância do **data science**.

Criando Compartimento

Para criar o compartimento navegue por.

Menu -> Identity -> Compartments -> Create Compartment.

Forneça o **nome** e **descrição** do seu compartimento e deixe o compartimento **root** selecionado, após isso clique em Criar Compartimento**.**

Criando Grupo de Usuário

Para criar o grupo de usuário navegue por.

Menu -> Groups -> Create Group.

Forneça o **nome** e **descrição** do grupo e clique em Criar.

Clique no nome do grupo que acabamos de criar, em seguida adicione um usuário a esse grupo.

Criando Grupo Dinâmico

Para criar o grupo de dinâmico navegue por.

Menu -> Identity -> Create Dynamic Groups.

Forneça o **nome** e **descrição** do grupo.

Vamos adicionar uma **rule** de acesso ao **datasciencenotebook**, para isso será necessário o **OCID** do seu compartment, volte para tela do compartment e copie o **OCID.**

Em seguida, substitue o campo **Colar-OCID** com o ocid do seu compartment

```
ALL {resource.type = 'datasciencenotebooksession', resource.compartment.id = 'Colar-OCID'}
```

Criando Políticas para Grupo de Usuário e Grupo Dinâmico.

Para criar políticas navegue por.

Menu -> Identity -> Policies.

Forneça **nome** e **descrição** da policy.

Dentro da mesma tela forneça as **policies** que irá permitir a utilização de recursos cloud em nosso compartimento.

A sintax de uma policy necessita de um **group**, **verb**, **resource** e **compartment**.

Policy para Grupo de Usuários

*allow group **group-name** to manage data-science-family in compartment **nome-compartment***
*allow group **group-name** to use virtual-network-family in compartment **nome-compartment***
*allow service data science to use virtual-network-family in compartment **nome-compartment***
*allow group **group-name** to manage repos in compartment **nome-compartment***

Policy para Grupo Dinâmico

*allow dynamic-group **nome-do-grupo-dinamico** to manage data-science-family in **compartment nome-comparment***

Criando VCN Virtual Cloud Networking

Para criar a virtual networking navegue por.

Menu -> Networking -> Overview.

Na tela de Overview procure por **Create a VCN with Internet Connectivity** e clique em **Start VCN Wizard.**

Dentro da tela de criação da **VCN**, fornaça o **nome** e selecione o **compartimento** criado anteriormente.

Não precisa alterar nenhum tipo de configuração, basta apenas clicar em próximo. Observe que algumas configurações já foram criadas automaticamente, isso irá nos ajudar.

Clique em **Criar** para finalizar sua **VCN**.

Criando Projeto & Notebook no Data Science

Criando Projeto Data Science

Para criar o projeto de data science navegue por.

Menu -> Data Science -> Projects.

Dentro da tela de projetos do Data Science, selecione o compartimento que criamos anteriormente e em seguida clique em criar projeto. Forneça **nome**, **descrição** e selecione o **compartimento** recém criado.

Ao acessar o projeto recem criado, é possível visualizar os detalhes do projeto.

Na mesma página iremos criar nosso notebook Jupyter.

Criando notebook data science

Dentro da página de detalhes do projeto clique em Criar Notebook.

Uma nova tela irá aparecer, selecione o compartment criando anteriormente e forneça **Nome**, **Compute Shape**, **Block Storage**, **VCN** e **SubNet**.

Após passar todas as informações clique em Criar Sessão de Notebook.

Em poucos minutos a sua instância estará pronta e você irá poder acessar o seu Jupyter Notebook.

Acessando o Jupyter Notebook,Treinando e Deployando Modelo.

Após realizar todos os passos de configuração de ambiente, finalmente iremos acessar o Laboratório Jupyter.

Para acessar o Jupyter Notebook, basta clicar em **abrir** na página de detalhes e uma nova página será carregada.

O primeiro passo a ser executado aqui é abrir o notebook do **Getting Starded** e executar um alguns paragráfos de configuração.

O primeiro parágrafo é de importação de algumas bibliotecas e o segundo é para validarmos o acesso com a internet.

O Terceiro é para criação do **.config** com as variáveis com os valores atualizados sobre nosso ambiente, **id do notebook**, **id do projeto** e outros. Esse arquivo é necessário para validação de autenticação.

Próximo passo é criarmos o nosso notebook de treinamento.

Feche o notebook do Getting Started e clique em novo notebook e partiu treinar o modelo!

Abaixo estão todos os códigos, treinamento e deployment do modelo.

```
import numpy as np
import os
import cv2
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Flatten
from keras.callbacks import TensorBoard


dire = r''

categ = ['cats', 'dogs']

data = []

for category in CATEGORIES:
    path = os.path.join(dire, category)
    for img in os.listdir(path):
        img_path = os.path.join(path, img)
        label = CATEGORIES.index(category)
        arr = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        new_arr = cv2.resize(arr,(100, 100))
        data.append([new_arr, label])

plt.imshow(data[3][0])

random.shuffle(data)

X = []
y = []

for features, label in data:
    X.append(features)
    y.append(label)


X = np.array(X)
y = np.array(y)

X = X.reshape(-1, 100, 100, 1)

X = X/255

X.shape



model = Sequential()

model.add(Conv2D(64, (3,3), activation = 'relu'))
model.add(MaxPooling2D((2,2)))

model.add(Conv2D(64, (3,3), activation = 'relu'))
model.add(MaxPooling2D((2,2)))

model.add(Flatten())

model.add(Dense(128, input_shape=[None,100,100,1], activation = 'relu'))
```

```
import numpy as np
import os
import cv2
import random
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Flatten
from keras.callbacks import TensorBoard

dire = r''

categ = ['cats', 'dogs']

data = []

for category in categ:
    path = os.path.join(dire, category)
    for img in os.listdir(path):
        img_path = os.path.join(path, img)
        label = categ.index(category)
        arr = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        new_arr = cv2.resize(arr,(50, 50))
        data.append([new_arr, label])



random.shuffle(data)

X = []
y = []

for features, label in data:
    X.append(features)
    y.append(label)


X = np.array(X)
y = np.array(y)
X = X.reshape(-1, 50, 50, 1)


X = X/255

X.shape


model = Sequential()

model.add(Conv2D(64, (3,3), activation = 'relu'))
model.add(MaxPooling2D((2,2)))

model.add(Conv2D(64, (3,3), activation = 'relu'))
model.add(MaxPooling2D((2,2)))

model.add(Flatten())

model.add(Dense(128, input_shape=[None,100,100,1], activation = 'relu'))
```

```
model.add(Dense(2, activation = 'softmax'))

model.compile(optimizer='adam',
          loss='sparse_categorical_crossentropy',
          metrics=['accuracy'])

model.fit(X, y, epochs=5, validation_split=0.1)

X.shape

catPredict = ['cat', 'dog']


def image(path):
   img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
   new_arr = cv2.resize(img, (50, 50))
   new_arr = np.array(new_arr)
   new_arr = new_arr.reshape(-1, 50, 50, 1)
   return new_arr

prediction = model.predict([image('test/123.jpg')])
print(catPredict[prediction.argmax()])
```

Criando Imagem Docker e Function

Para disponibilização do nosso modelo para outra aplicações e serviços consumirem, é preciso criar uma function.

Nas etapas anteriores nós já criamos os arquivos necessários para a criação dessa função.

Até esse ponto já temos nosso modelo treinado e "deployado" no model catalog, o próximo passo é criar uma Function para realizar predições sem estar necessariamente dentro do Jupyter Notebook.

O primeiro passo é adicionarmos algumas políticas de liberação de acesso.

Para adicionar as políticas navegue por.

Menu -> Identity -> Políticas.

No canto esquerdo inferior selecione o compartimento **(root).**

Neste momento iremos criar algumas políticas que precisam ser criadas no compartimento root.

A primeira política será para liberar acesso ao **Oracle Function** e a segunda para o **API Gateway.**

Após selecionar o compartimento **root,** clique em criar política.

Forneça o nome, descrição e novamente selecione o compartimento **root**, para criarmos a primeira política**.**

Obs: *Estamos trabalhando com o compartimento root apenas neste momento, em etapas futuras isso não irá acontecer.*

No construtor de políticas altere para construtor avançado e cole o seguinte código.

```
allow service FAAS to use virtual-network-family in tenancy
allow service FAAS to read repos in tenancy
```

Clique em criar e a política de acesso ao **Oracle Function** esta feita!

Refaça o mesmo passo-a-passo para criar a polítca de acesso ao recurso **API Gateway**.

O Código em relação ao **API Gateway** é o seguinte.

Aqui eu trago dois exemplo para liberar acesso ao serviço.

Liberar acesso para qualquer pessoa utilizar o serviço de function.

```
Allow any-user to manage functions-family in tenancy
```

Ou liberar acesso apenas para um determinado compartimento.

ALLOW any-user to use functions-family in compartment nome-compartimento where ALL {request.principal.type = 'ApiGateway', request.resource.compartment.id = '<OCID-DO-COMPARTIMENTO>'}

O segundo passo para iniciarmos a criação da function é fazer o downlod do nosso Modelo.

Na página principal localize o icone do **Cloud Shell** e clique para abrir o terminal(canto direito superior).

Vamos utilizar alguns comandos **OCI CLI** para fazer o download do nosso modelo.

Primeiro vamos copiar o OCID do nosso modelo.

Navegue para página de detalhes do projeto do data science.

Menu -> Data Science -> Project -> Models.

Procure por **OCID** e clique em **COPIAR.**

Agora que temos nosso OCID do modelo, podemos realizar o download!

Volte para a console do shell, e digite os seguintes comandos.

Criação da pasta
mkdir demo

Acessar a nova pasta
cd demo

Download do model
oci data-science model get-artifact-content --model-id ***<colar-ocid >*** *--file* ***<nome-do-arquivo>.zip***

Unzip do arquivo baixado
unzip ***<nome-do-arquivo>****.zip*

Listar os arquivos
ls

Verificar arquivo requirements.txt
cat requirements.txt

Criando Function no OCI

Agora que já temos todos os arquivos prontos, podemos iniciar o deploy da nossa function.

*Para criarmos a nossa function precisamos acessar o menu do **OCI** e procurar por **Developer Services & Functions**.*

Na página do Function, clique em Criar Aplicativo.

Forneça o **nome e** selecione a **VCN e Sub Net** e clique em Criar.

Após criar a Function, clique no mesmo para abrir a página de detalhes.

Logo em seguida clique na aba **Conceitos Básico OU Getting Starded** para inicarmos o nosso o deploy da função.

Dentro da página de **Conceitos Básico OU Getting Starde** Selecione Configurações do Cloud Shell e um tutorial com tudo que você deve fazer irá aparecer, basicamente basta você seguir esse tutorial para criar a sua imagem docker e realizar o **deploy** da function.

OBS: É de extrema importância você estar dentro da pasta que contém os arquivos relacionado ao modelo ML, passos atrás nós "dezipamos" os arquivos .py, pkl, .txt e outros.

Após realizar o deploy de sua função, você pode testar para verificar se tudo está funcionando da maneira que você queira.

Com objetivo de enviar dados para a função analisar e predizer uma saída, vamos criar alguns arquivos no formato **.json.**

Para criar o arquivo, digite o seguinte comando em seu Cloud Shell, lembrando que você acessa o cloud shell pelo icone que representa o mesmo, você pode localizar no canto direito superior.

Vamos criar 2 Arquivos.

vim data.json
Aperte a tecla **i** para poder comentar o arquivo.
Cole o seguinte texto.

```
{"input":[[5.8, 2.8, 5.1, 2.4]]}
```

vim data2.json
Aperte a tecla **i** para poder comentar o arquivo.
Cole o seguinte texto.

```
{"input":[[5.8, 2.8, 5.1, 2.4],[5.8, 2.8, 5.1, 2.4],[5.8, 2.8, 5.1, 2.4],[5.8, 2.8, 5.1, 2.4],
        [5.8, 2.8, 5.1, 2.4],[5.8, 2.8, 5.1, 2.4],[5.8, 2.8, 5.1, 2.4],
        [5.8, 2.8, 5.1, 2.4],[5.8, 2.8, 5.1, 2.4],[5.8, 2.8, 5.1, 2.4],
        [5.8, 2.8, 5.1, 2.4]]}
```

Após criar os arquivos, podemos realizar as predições utilizando o fn invoke.

Segue os comandos para invocação.

Comando de invocação da função
cat ***<nome-do-arquivo>*** *| fn invoke* ***<nome-da-funcao>*** *model_api --content-type application/json*

Lembrando que você pode visualizar a imagem docker criada navegando por,

Menu -> Serviços ao Desenvolvedor -> Registro(OCIR) .

Partiu próxima etapa!

Criando API Gateway

Na etapa anterior criamos nossa imagem e função, com isso já é possível realizar predições via

Fn function invoke.

O objetivo nesta etapa é criar uma **API** disponibilizar nossa Function para ser consumida via **API**.

Para criarmos uma API selecione o **Menu**, depois procure por ***Developer Services &* API Gateway.**

Selecione o compartimento e forneça o **Nome**, **Tipo**, **VCN** e **Subnet**.

Clique na API Gateway recém criado.

Na tela do **API Gateway** clique em implementações para iniciarmos o deploy da API.

Clique em **Criar Deployment**.

No processo de implementação será preciso completar 2 etapas, **Informações básica** e **rotas**.

Preenchendo informações básicas.

Forneça o **Nome**, **prefixo do caminho** e selecione o compartimento utilizado em todos.

Preenchendo API politicas de Request.

Você consegue aplicar algumas políticas em relação a cada request recebido, porém, neste tutorial não vamos aplicar nenhuma política.

Políticas de Registro em Log da API.

Deixe informações selecionado.

E clique em próximo.

Preenchendo Rotas

Vamos configurar apenas a nossa rota do **predict**.

Digite seu caminho.

Em métodos, selecione **Post**.

Em **Tipo**, selecione **Oracle Function**.

Selecione o compartimento que possui o **Oracle Function**.

Selecione a Function criada anteriomente.

Clique em fazer Deploy.

Enviando Request de predição

Ao completar o deploy, uma mensagem será plotada na tela no canto direito superior, junto a isto um end-point será disponibilizado para que possamos utiliza-lo.

Para pegar o end-point, clique na API implementada e procure por end-point.

Com o end point em mãos já podemos enviar um request POST com os dados para o modelo realizar os predicts.

Podemos realizar o request por diversos caminhos, neste caso irei usar o **CURL** e iremos consumir os arquivos **.json** criados anteriormente.

Se por acaso você não criou os arquivos, esse é o passo a passo de criação.

Para criar o arquivo, digite o seguinte comando em seu Cloud Shell.

vim data.json
Aperte a tecla **i** para poder comentar o arquivo.
Cole o seguinte texto.

```
{"input":[[5.8, 2.8, 5.1, 2.4]]}
```

vim data2.json
Aperte a tecla **i** para poder comentar o arquivo.
Cole o seguinte texto.

```
{"input":[[5.8, 2.8, 5.1, 2.4],[5.8, 2.8, 5.1, 2.4],[5.8, 2.8, 5.1, 2.4],[5.8, 2.8, 5.1, 2.4],
        [5.8, 2.8, 5.1, 2.4],[5.8, 2.8, 5.1, 2.4],[5.8, 2.8, 5.1, 2.4],
        [5.8, 2.8, 5.1, 2.4],[5.8, 2.8, 5.1, 2.4],[5.8, 2.8, 5.1, 2.4],
        [5.8, 2.8, 5.1, 2.4]]}
```

Agora que já temos nosso arquivo .json pronto, vamos enviar um request HTTP com esses dados.

Navegue em seu API Gateway até a página de detalhes e copie o seu end-point.

Substitui no código abaixo o **<COLAR-END-POINT>**

```
curl -k -X  POST <COLAR-END-POINT>/demo/predict -d @data.json --header "Content-type: application/json"
```

Após executar esse comando, enviaremos os dados para a função e em poucos segundos teremos nossa resposta com a predição.

Resumo

Se você chegou até aqui, parabéns!

Passamos por todas as etapas do nosso tutorial, criamos o compartimento, nossos grupos, aplicamos todas as políticas de permissão, criamos o projeto de data science, o oracle function + API gateway e conseguimos realizar a predição via API.

Agradeço o acompanhamento e viva o **UNIVERSO DE DADOS**!