Tarefa Final - Parte 2

Nome

Ivan Prado da Costa

Reproduza um código da Web utilizando SQL com Python (pyspark). Submeta o código no github. Deixe seus comentários sobre o desenvolvimento aqui.

RESPOSTA:

Consultei algumas páginas explicando sobre a conexão com banco de dados SQL utilizando Python.

Vi alguns códigos de exemplo e pensei em adaptar para extrair algumas informações de um csv retirado do site Kaggle (considerando que o conteúdo dessa base de dados esteja em um banco de dados SQL).

https://www.kaggle.com/heesoo37/120-years-of-olympic-history-athletes-and-results

Trata-se de uma base com dados básicos de atletas olímpicos que disputaram os jogos desde 1896 (Atenas) até 2016 (Rio de Janeiro).

Eu gostaria de adaptar o código para listar, por exemplo, os atletas brasileiros que constam na tabela original.

//Criar a conexão com o banco de dados com informações fictícias e selecionar algumas colunas da tabela

//Selecionar informações das colunas "Nome" e "Ano" para atletas do time do Brasil ordenando por ano crescente

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)
```

```
mycursor = mydb.cursor()

mycursor.execute("SELECT Name, Year FROM athlete_events WHERE NOC = 'BRA' ORDER BY Year")

myresult = mycursor.fetchall()

for x in myresult:
  print(x)
```

Reproduza um código da Web utilizando Stream com Python (pyspark). Submeta o código no github. Deixe seus comentários sobre o desenvolvimento aqui.

RESPOSTA:

Encontrei uma página chamada Faust, uma biblioteca para construir aplicações de streaming em Python.

https://faust.readthedocs.io/en/latest/

Achei interessante um exemplo que serve para contagem de page views em uma URL. Isso pode ser muito útil para aplicar em uma empresa de pequeno porte que não tem estrutura nem orçamento para ferramentas de marketing digital. Ou seja, de forma gratuita é possível levantar algumas estatísticas simples como o número de visualizações/visitas em uma página web.

```
# data sent to 'clicks' topic sharded by URL key.
# e.g. key="http://example.com" value="1"
click_topic = app.topic('clicks', key_type=str, value_type=int)

# default value for missing URL will be 0 with `default=int`
counts = app.Table('click_counts', default=int)

@app.agent(click_topic)
async def count_click(clicks):
   async for url, count in clicks.items():
      counts[url] += count
```

Reproduza um código da Web utilizando Machine Learning com Python (pyspark). Submeta o código no github. Deixe seus comentários sobre o desenvolvimento aqui. faca um gráfico mostrando a analise do resultado, envie junto no github.

RESPOSTA:

Li algumas páginas que falam sobre como começar a utilizar machine learning com python. Várias delas citam a base de dados de flores de íris.

O arquivo contém quatro colunas de medidas das observações de flores além da espécie da flor. Portanto a ideia é que o algoritmo "aprenda" e identifique padrões para que aponte a espécie da flor a partir das medidas. Inicialmente é importante que o modelo passe por uma fase de treino, aplicando o método sobre uma amostra do próprio conjunto de dados.

Pensei em utilizar um outro csv do site Kaggle com informações (características) de jogadores registrados no jogo virtual de futebol FIFA 19.

https://www.kaggle.com/karangadiya/fifa19

Na minha opinião seria interessante relacionar as colunas de altura (height) + as características que são destaque nos goleiros (GKDiving, GKHandling, GKKicking, por exemplo) para definir se aquele atleta é de fato um goleiro. E é possível confirmar pela colula de posição (position). No caso, a posição de goleiro é representada pela sigla GK. A base possui mais de 18 mil linhas e é viável separar uma amostra para treino/teste.

Pelo exemplo abaixo, retirado de um blog, o autor segue com a ideia da identificação das flores que eu pensei em adaptar para identificação de goleiros a partir de sua altura + atributos do jogo.

https://medium.com/blog-do-zouza/seu-primeiro-projeto-de-machine-learning-em-python-passo-a-passo-78c5f7bce22d

```
# Carregar bibliotecas
import pandas
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.metrics import classification_report
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

# Carregar conjunto de dados. Aqui traria a base do jogo FIFA e as colunas que nos interessam para análise
url = “https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
names = [‘sepal-length’, ‘sepal-width’, ‘petal-length’, ‘petal-width’, ‘class’]
dataset = pandas.read_csv(url, names=names)

# Verificação do tamanho do conjunto de dados
print(dataset.shape)

# Verificação das primeiras 20 linhas
print(dataset.head(20))

# Estatísticas descritivas
print(dataset.describe())

# Distribuição de classe. Aqui trocaria por distribuição da posição do atleta. GK é uma das possibilidades.
print(dataset.groupby(‘class’).size())

# Visualização dos dados com box plot, histograma e dispersão
dataset.plot(kind=’box’, subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()

dataset.hist()
plt.show()

scatter_matrix(dataset)
```

```
plt.show()


# Criação de um conjunto de dados para validação. Aqui é necessária adaptação para a base do jogo FIFA.
array = dataset.values
X = array[:,0:4]
Y = array[:,4]
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, Y, test_size=validation_size, random_state=seed)


# Métrica de teste
seed = 7
scoring = 'accuracy'


# Criação de 6 modelos a partir de algoritmos distintos.
models = []
models.append(('LR', LogisticRegression(solver='liblinear', multi_class='ovr')))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))
# evaluate each model in turn
results = []
names = []
for name, model in models:
  kfold = model_selection.KFold(n_splits=10, random_state=seed)
  cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold, scoring=scoring)
  results.append(cv_results)
  names.append(name)
  msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
  print(msg)


#Avaliar acurácia (média e desvio) de cada modelo para escolha do mais adequado
```

#Finalmente aplicar o modelo em base de treino para medir o resultado

#No caso aplicaria em uma amostra de jogadores fornecendo como entrada as características e avaliando se o modelo é confiável para predição se aquele jogar é um goleiro ou não.

#É possível representar o resultado desses testes em formato de gráfico para melhor visualização.