Roteiro:

1 - Criação de microservices usando Flask e Containers para execução de modelos de IA:

Estrutura do projeto:
3 arquivos: teste.py, **requirements.txt**, **Dockerfile**.

Dockerfile:

```
FROM python:3-alpine
COPY . /app
WORKDIR /app
RUN pip install --trusted-host pypi.org --trusted-host files.pythonhosted.org -r requirements.txt
ENTRYPOINT ["python"]
CMD ["teste.py"]
EXPOSE 5000
```

requirements.txt:

```
Click==7.0
Flask==1.0.2
itsdangerous==1.1.0
Jinja2==2.10.1
MarkupSafe==1.1.1
Werkzeug==0.15.5
```

Teste.py (Lembrete! Python usa indentação!)

```
# Import das dependências do Flask
from flask import Flask, request, make_response, jsonify
# Inicializa a app do flask
app = Flask(__name__)
# Rota default
@app.route('/')
def index():
    return 'Hello World!'
# Função de Respostas
def results():
    # Constrói o objeto de respostas
    req = request.get_json(force=True)
    # Pega a ação do json
    action = req.get('queryResult').get('action')
    # retorna o fulfillment do response
    return {'fulfillmentText': 'This is a response from webhook.'}
# cria uma rota para o webhook
@app.route('/webhook', methods=['GET', 'POST'])
def webhook():
    # retorno da resposta
    return make_response(jsonify(results()))
# Executa a aplicação
if __name__ == '__main__':
   app.run(host='0.0.0.0', port=5000)
```

2 – Execução de Modelos:

Statements:

- A maioria dos projetos de Aprendizado de Máquinas resulta em uma classificação ou regressão geralmente utilizando abordagens de aprendizado supervisionadas;
- A extração de características dos dados é normalmente a parte crítica da maioria dos processos de aprendizado de máquinas;
- Algoritmos de aprendizado não supervisionado como o K-means, permitem agrupar os dados quando não se sabe previamente os rótulos;

Regressão:

- Definição:

Dividida em Linear e Multilinear, é utilizada para prever valores futuros, considerando valores no passado (Base histórica).

- Usado para:

Forecast, Valores de ações, previsão de risco, classificação, determinar características.

Como exemplo: usamos regressão linear para predizer o preço de uma ação no mercado de ações que é a sua variável dependente baseado em algumas variáveis iniciais como a média de interesse e a média de desemprego.

- Exemplo:

Um exemplo do dataset:

Year	Month	Interest_Rate	Unemployment_Rate	Stock_Index_Price
2017	12	2.75	5.3	1464
2017	11	2.5	5.3	1394
2017	10	2.5	5.3	1357
2017	9	2.5	5.3	1293
2017	8	2.5	5.4	1256
2017	7	2.5	5.6	1254
2017	6	2.5	5.5	1234
2017	5	2.25	5.5	1195
2017	4	2.25	5.5	1159
2017	3	2.25	5.6	1167
2017	2	2	5.7	1130
2017	1	2	5.9	1075
2016	12	2	6	1047
2016	11	1.75	5.9	965
2016	10	1.75	5.8	943
2016	9	1.75	6.1	958
2016	8	1.75	6.2	971
2016	7	1.75	6.1	949
2016	6	1.75	6.1	884
2016	5	1.75	6.1	866
2016	4	1.75	5.9	876
2016	3	1.75	6.2	822
2016	2	1.75	6.2	704
2016	1	1.75	6.1	719

No contexto acima estamos tentando prever o valore de Stock_index_price considerando Interest_rate e Unemployment_rate.

Usando

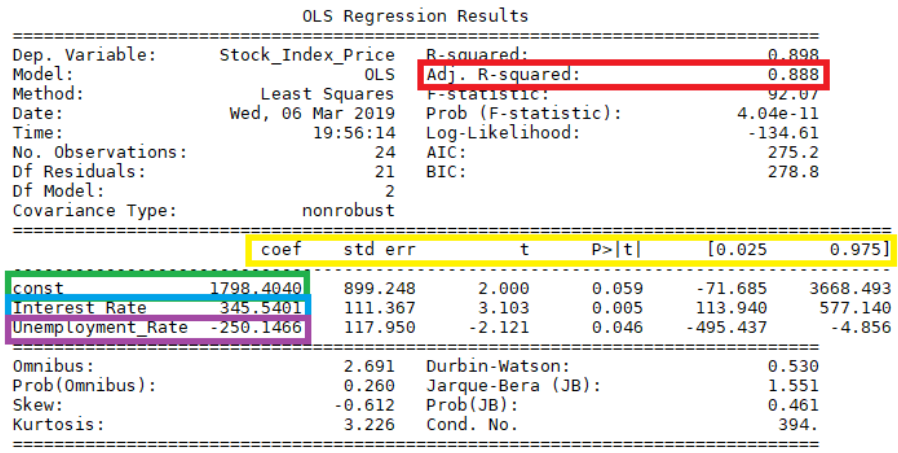
OBS: Iremos utiliza o statsmodel, que é um modulo do python que prove classes e funções para estimar diferentes modelos estatísticos. (https://www.statsmodels.org/stable/index.html)

O seu arquivo requirements.txt irá ter o pandas e o statsmodels.

```
from pandas import DataFrame

import statsmodels.api as sm



Stock_Market = {'Year':
[2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,2016,2
016,2016,2016,2016,2016,2016,2016,2016,2016,2016],

                'Month': [12,
11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],

                'Interest_Rate':
[2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75,1.75,1.75,1
.75,1.75,1.75,1.75,1.75,1.75],

                'Unemployment_Rate':
[5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.
1,5.9,6.2,6.2,6.1],

                'Stock_Index_Price':
[1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,1047,965,943,958
,971,949,884,866,876,822,704,719]

                }



df =
DataFrame(Stock_Market,columns=['Year','Month','Interest_Rate','Unemployment_
Rate','Stock_Index_Price'])
```

```
X = df[['Interest_Rate','Unemployment_Rate']]

Y = df['Stock_Index_Price']



X = sm.add_constant(X)



model = sm.OLS(Y, X).fit()

predictions = model.predict(X)



print_model = model.summary()

print(print_model)
```

Executando o código o resultado é próximo a:



OLS Regression Results

Dep. Variable:	Stock_Index_Price	R-squared:	0.898
Model:	OLS	Adj. R-squared:	0.888
Method:	Least Squares	F-statistic:	92.07
Date:	Wed, 06 Mar 2019	Prob (F-statistic):	4.04e-11
Time:	19:56:14	Log-Likelihood:	-134.61
No. Observations:	24	AIC:	275.2
Df Residuals:	21	BIC:	278.8
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1798.4040	899.248	2.000	0.059	-71.685	3668.493
Interest_Rate	345.5401	111.367	3.103	0.005	113.940	577.140
Unemployment_Rate	-250.1466	117.950	-2.121	0.046	-495.437	-4.856

Omnibus:	2.691	Durbin-Watson:	0.530
Prob(Omnibus):	0.260	Jarque-Bera (JB):	1.551
Skew:	-0.612	Prob(JB):	0.461
Kurtosis:	3.226	Cond. No.	394.

Adjusted. R-squared é o valor do fit do modelo vai de 0 a 1, **const coefficient** calculo entre Interest_Rate e Unemployment_Rate, **Interest_Rate coefficient** e **Unemployment_Rate coeficiente** são os cálculos estatísticos para as duas colunas.

Portanto para um cenário de:

Interest Rate(x1) = 2.75

Unemployment Rate(x2) = 5.3

Temos:

Stock_Index_Price = (const coef) + (Interest_Rate coef)*X_1 + (Unemployment_Rate coef)*X_2

Stock_Index_Price = (1798.4040) + (345.5401)*X_1 + (-250.1466)*X_2

O valor para janeiro de 2018 é:

Stock_Index_Price = (1798.4040) + (345.5401)*(2.75) + (-250.1466)*(5.3) = 1422.86

Para o mesmo modelo vamos rodar a regressão linear múltipla que difere na análise de 2 ou mais variáveis que devem ser lineares usamos aqui o sckit.learn, matplotlib:

Organizando os dados:

```
from pandas import DataFrame



Stock_Market = {'Year':
[2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,2016,2
016,2016,2016,2016,2016,2016,2016,2016,2016],

                'Month': [12,
11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],

                'Interest_Rate':
[2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75,1.75,1.75,1
.75,1.75,1.75,1.75,1.75,1.75],

                'Unemployment_Rate':
[5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.
1,5.9,6.2,6.2,6.1],

                'Stock_Index_Price':
[1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,1047,965,943,958
,971,949,884,866,876,822,704,719]

                }

df =
DataFrame(Stock_Market,columns=['Year','Month','Interest_Rate','Unemployment_
Rate','Stock_Index_Price'])



print (df)
```

Checando a linearidade:

```
from pandas import DataFrame

import matplotlib.pyplot as plt
```

```
Stock_Market = {'Year':
[2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,2016,2
016,2016,2016,2016,2016,2016,2016,2016,2016],


                'Month': [12,
11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],


                'Interest_Rate':
[2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75,1.75,1.75,1
.75,1.75,1.75,1.75,1.75,1.75],


                'Unemployment_Rate':
[5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.
1,5.9,6.2,6.2,6.1],


                'Stock_Index_Price':
[1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,1047,965,943,958
,971,949,884,866,876,822,704,719]


                }




df =
DataFrame(Stock_Market,columns=['Year','Month','Interest_Rate','Unemployment_
Rate','Stock_Index_Price'])




plt.scatter(df['Interest_Rate'], df['Stock_Index_Price'], color='red')


plt.title('Stock Index Price Vs Interest Rate', fontsize=14)


plt.xlabel('Interest Rate', fontsize=14)


plt.ylabel('Stock Index Price', fontsize=14)


plt.grid(True)


plt.show()




plt.scatter(df['Unemployment_Rate'], df['Stock_Index_Price'], color='green')


plt.title('Stock Index Price Vs Unemployment Rate', fontsize=14)
```

```
plt.xlabel('Unemployment Rate', fontsize=14)

plt.ylabel('Stock Index Price', fontsize=14)

plt.grid(True)

plt.show()
```

Intercept:

1798.4039776258546

Coefficients:

[345.54008701 -250.14657137]

Predicted Stock Index Price com o sckitlearn:

[1422.86238865]

Regressão Logística

- Definição:

Permite estimar a probabilidade associada à ocorrência de um evento considerando variáveis.

- Usado para:

Ex: não existe uma probabilidade de 125% a 35% de você engravidar logo deve ser utilizada a regressão logística para casos binários.

- Dataset:

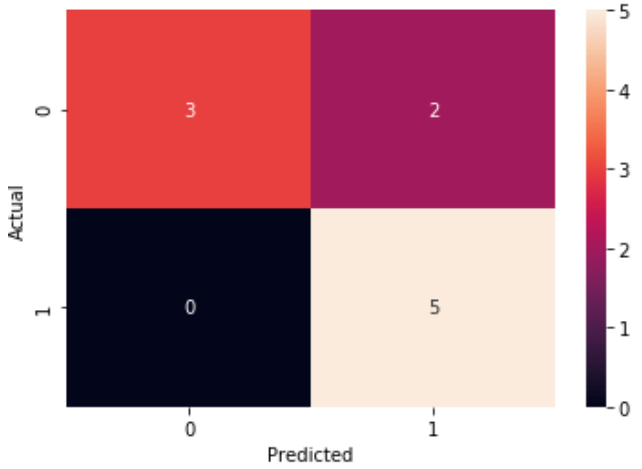
gmat	gpa	work_experience	admitted
780	4	3	1
750	3.9	4	1
690	3.3	3	1
710	3.7	5	1
680	3.9	4	1
730	3.7	6	1
690	2.3	1	0
720	3.3	4	1
740	3.3	5	1
690	1.7	1	0
610	2.7	3	0
690	3.7	5	1
710	3.7	6	1
680	3.3	4	1
770	3.3	3	1
610	3	1	0
580	2.7	4	0
650	3.7	6	1
540	2.7	2	0
590	2.3	3	0
620	3.3	2	0
600	2	1	0
550	2.3	4	0
550	2.7	1	0
570	3	2	0
670	3.3	6	1
660	3.7	4	1
580	2.3	2	0
650	3.7	6	1
660	3.3	5	1
640	3	1	0
620	2.7	2	0
660	4	4	1
660	3.3	6	1
680	3.3	5	1
650	2.3	1	0
670	2.7	2	0
580	3.3	1	0
590	1.7	4	0
690	3.7	5	1

Pacotes: Pandas, sklearn, seaborn

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn import metrics

import seaborn as sn



candidates = {'gmat':
[780,750,690,710,680,730,690,720,740,690,610,690,710,680,770,610,580,650,540,
590,620,600,550,550,570,670,660,580,650,660,640,620,660,660,680,650,670,580,5
90,690],

              'gpa':
[4,3.9,3.3,3.7,3.9,3.7,2.3,3.3,3.3,3.3,1.7,2.7,3.7,3.7,3.3,3.3,3,2.7,3.7,2.7,2.3,
3.3,2,2.3,2.7,3,3.3,3.7,2.3,3.7,3.3,3,2.7,4,3.3,3.3,2.3,2.7,3.3,1.7,3.7],

              'work_experience':
[3,4,3,5,4,6,1,4,5,1,3,5,6,4,3,1,4,6,2,3,2,1,4,1,2,6,4,2,6,5,1,2,4,6,5,1,2,1,
4,5],
```

```
                'admitted':
[1,1,1,1,1,1,0,1,1,0,0,1,1,1,1,1,0,0,1,0,0,0,0,0,0,0,1,1,0,1,1,0,0,1,1,1,0,0,0,
0,1]


                }



df = pd.DataFrame(candidates,columns= ['gmat',
'gpa','work_experience','admitted'])



#print (df)



X = df[['gmat', 'gpa','work_experience']]

y = df['admitted']



X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.25,random_state=0)



logistic_regression= LogisticRegression()

logistic_regression.fit(X_train,y_train)

y_pred=logistic_regression.predict(X_test)



confusion_matrix = pd.crosstab(y_test, y_pred, rownames=['Actual'],
colnames=['Predicted'])

sn.heatmap(confusion_matrix, annot=True)
```

```
print('Accuracy: ',metrics.accuracy_score(y_test, y_pred))
```



- **TP** = True Positives = 5
- **TN** = True Negatives = 3
- **FP** = False Positives = 2
- **FN** = False Negatives = 0

Fazendo o print de X e Y

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression



candidates = {'gmat':
[780,750,690,710,680,730,690,720,740,690,610,690,710,680,770,610,580,650,540,
590,620,600,550,550,570,670,660,580,650,660,640,620,660,660,680,650,670,580,5
90,690],

              'gpa':
[4,3.9,3.3,3.7,3.9,3.7,2.3,3.3,3.3,3.3,1.7,2.7,3.7,3.7,3.3,3.3,3,2.7,3.7,2.7,2.3,
3.3,2,2.3,2.7,3,3.3,3.7,2.3,3.7,3.3,3,2.7,4,3.3,3.3,2.3,2.7,3.3,1.7,3.7],

              'work_experience':
[3,4,3,5,4,6,1,4,5,1,3,5,6,4,3,1,4,6,2,3,2,1,4,1,2,6,4,2,6,5,1,2,4,6,5,1,2,1,
4,5],
```

```
                'admitted':
[1,1,1,1,1,1,0,1,1,0,0,1,1,1,1,1,0,0,1,0,0,0,0,0,0,0,1,1,0,1,1,0,0,1,1,1,0,0,0,
0,1]


                }




df = pd.DataFrame(candidates,columns= ['gmat',
'gpa','work_experience','admitted'])




X = df[['gmat', 'gpa','work_experience']]


y = df['admitted']




X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.25,random_state=0)  #train is based on 75%
of the dataset, test is based on 25% of dataset




logistic_regression= LogisticRegression()


logistic_regression.fit(X_train,y_train)


y_pred=logistic_regression.predict(X_test)




print (X_test) #test dataset (without the actual outcome)


print (y_pred) #predicted values
```

	gmat	gpa	work_experience
22	550	2.3	4
20	620	3.3	2
25	670	3.3	6
4	680	3.9	4
10	610	2.7	3
15	610	3.0	1
28	650	3.7	6
11	690	3.7	5
18	540	2.7	2
29	660	3.3	5

Voltando no dataset temos resultados corretos para 8 de 10 ou seja 80% de accuracy.

K-Means Clustering

- Definição:

Aprendizado não supervisionado *(Unsupervised* Learning). Clustering é a quantidade de grupos que vamos montar.

- Usado para:

Separar e encontrar grupos em volume de dados sem label.

- Dataset:

x	y
25	79
34	51
22	53
27	78
33	59
33	74
31	73
22	57
35	69
34	75
67	51
54	32
57	40

43	47
50	53
57	36
59	35
52	58
65	59
47	50
49	25
48	20
35	14
33	12
44	20
45	5
38	29
43	27
51	8
46	7

Capturar o dado usando pandas e python:

```
from pandas import DataFrame


Data = {'x':
[25,34,22,27,33,33,31,22,35,34,67,54,57,43,50,57,59,52,65,47,49,48,35,33,44,4
5,38,43,51,46],


        'y':
[79,51,53,78,59,74,73,57,69,75,51,32,40,47,53,36,35,58,59,50,25,20,14,12,20,5
,29,27,8,7]


       }


df = DataFrame(Data,columns=['x','y'])


print (df)
```

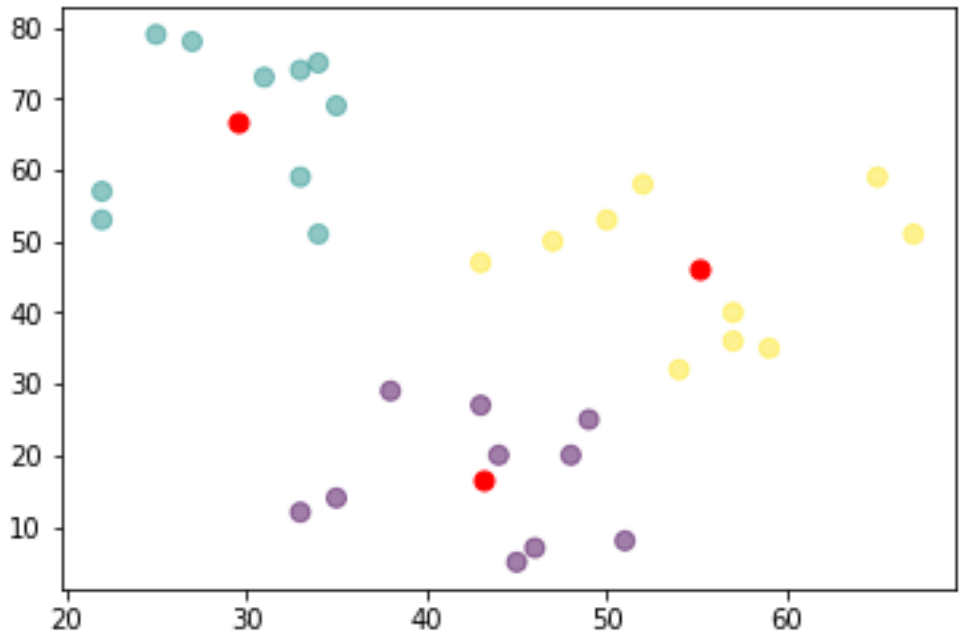
	x	y
0	25	79
1	34	51
2	22	53
3	27	78
4	33	59
5	33	74
6	31	73
7	22	57
8	35	69
9	34	75
10	67	51
11	54	32
12	57	40
13	43	47
14	50	53
15	57	36
16	59	35
17	52	58
18	65	59
19	47	50
20	49	25
21	48	20
22	35	14
23	33	12
24	44	20
25	45	5
26	38	29
27	43	27
28	51	8
29	46	7

Executando a separação

```
from pandas import DataFrame

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans
```

```
Data = {'x':
[25,34,22,27,33,33,31,22,35,34,67,54,57,43,50,57,59,52,65,47,49,48,35,33,44,4
5,38,43,51,46],


        'y':
[79,51,53,78,59,74,73,57,69,75,51,32,40,47,53,36,35,58,59,50,25,20,14,12,20,5
,29,27,8,7]


       }




df = DataFrame(Data,columns=['x','y'])




kmeans = KMeans(n_clusters=3).fit(df)

centroids = kmeans.cluster_centers_

print(centroids)




plt.scatter(df['x'], df['y'], c= kmeans.labels_.astype(float), s=50,
alpha=0.5)

plt.scatter(centroids[:, 0], centroids[:, 1], c='red', s=50)
```

```
[[43.2 16.7]
 [29.6 66.8]
 [55.1 46.1]]
```



Cluster = 4 ?

3 – Autokeras

O autokeras é um software open source

4 – Automl com dataiku.com

4.1 – Acesse https://www.dataiku.com/

4.2 – Acesse GET STARTED