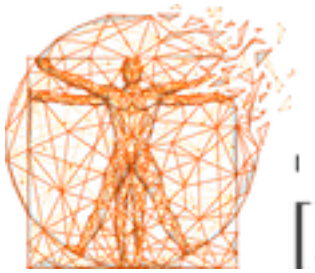


INSTITUTO
[D'VINCI]^3

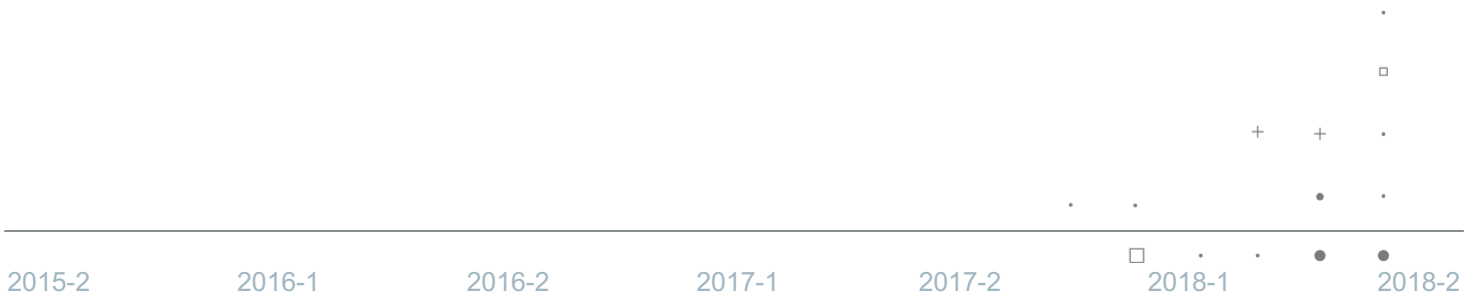
Data Science
Abordagens e Práticas

PROF. CLÁUDIO PINHEIRO

1C Prática

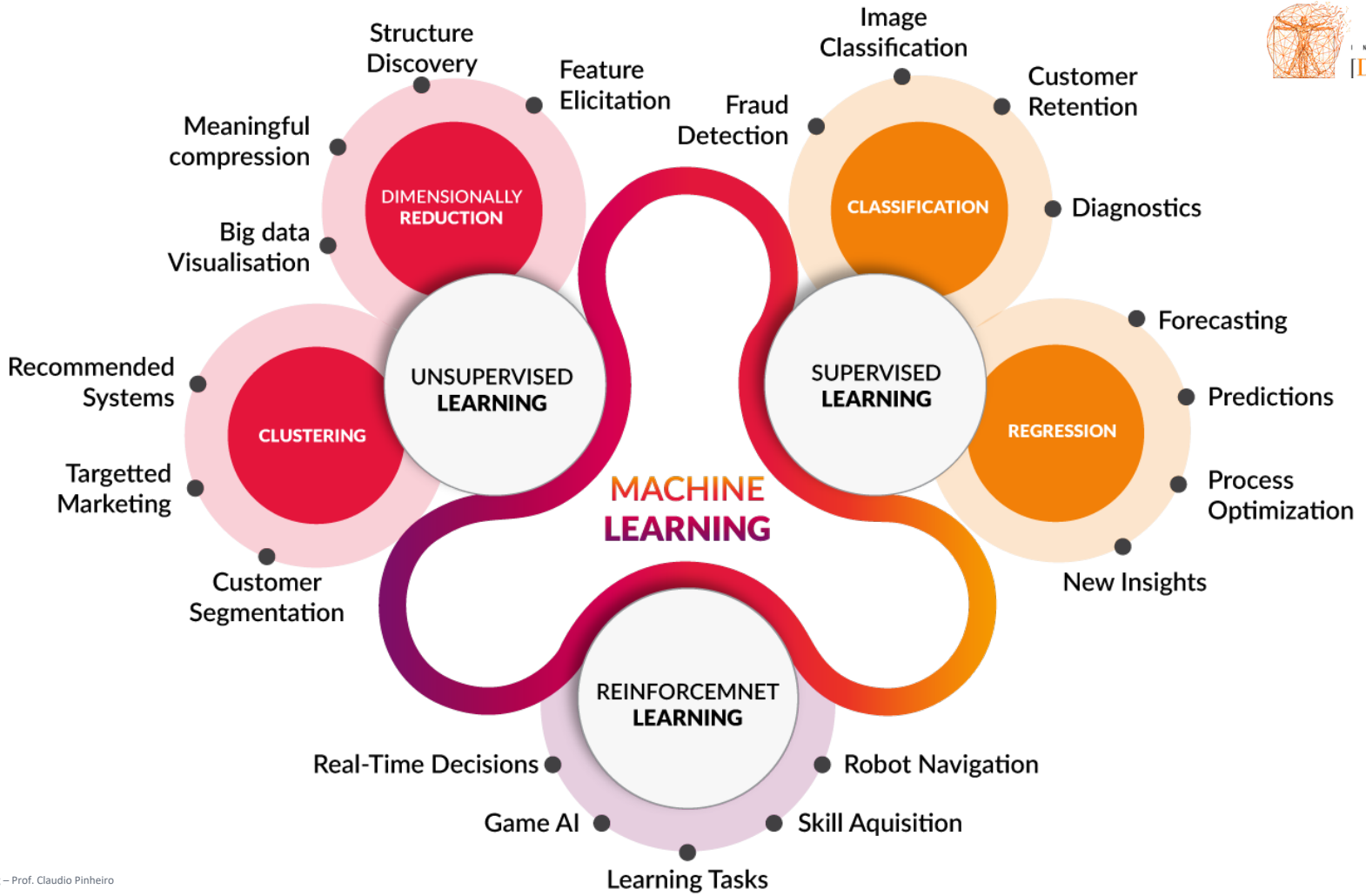
INTRODUÇÃO À MODELAGEM PREDITIVA

CRIANDO UM MODELO ANALÍTICO

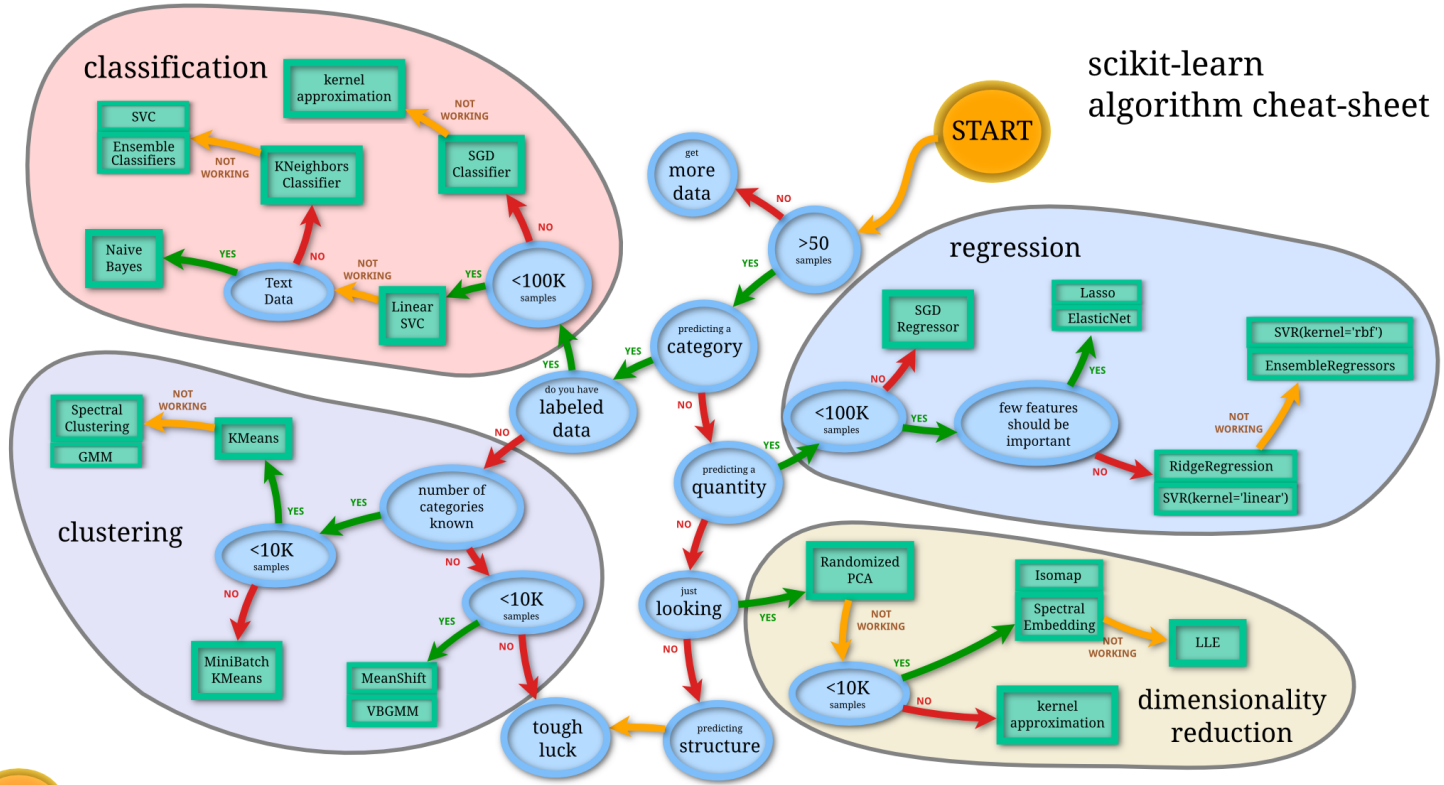


Tipos de Abordagem

- Supervisionada
- Não Supervisionada
- Aprendizagem por Reforço

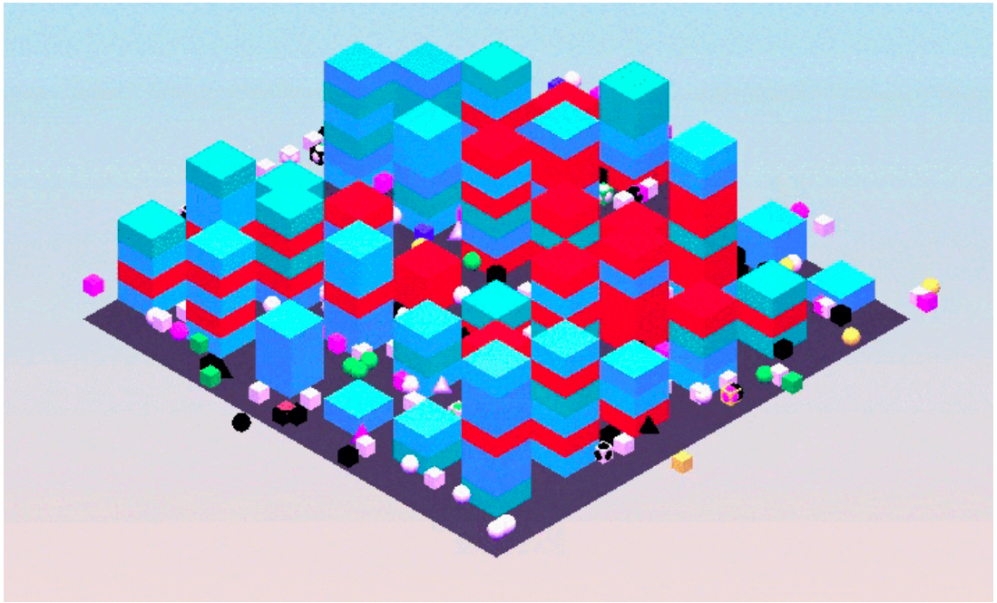

Structure Discovery
Feature Elicitation
Meaningful compression
Big data Visualisation
DIMENSIONALLY REDUCTION
Image Classification
Customer Retention
Fraud Detection
Diagnostics
CLASSIFICATION
Recommended Systems
Targetted Marketing
Customer Segmentation
CLUSTERING
UNSUPERVISED LEARNING
SUPERVISED LEARNING
Forecasting
Predictions
Process Optimization
New Insights
REGRESSION
MACHINE LEARNING
REINFORCEMNET LEARNING
Real-Time Decisions
Robot Navigation
Game AI
Skill Aquisition
Learning Tasks

Técnicas por Abordagem



Back

Processo de modelagem com Machine Learning



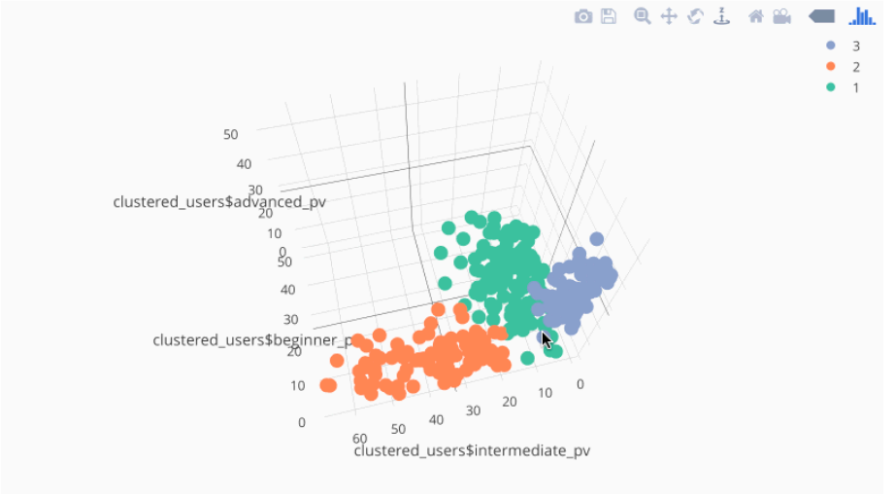
Neste notebook vamos apresentar conceitos e códigos referentes a um processo de modelagem do inicio ao fim, testando o modelo desenvolvido.

Ainda que simples serve como uma referência para modelagens futuras.

O processo mais adequado e completo será apresentado durante as aulas e trata-se do CRISP-DM

<img src = 'https://frnsys.com/assets/hosny/disparity.gif' width =600>

Agora vamos carregar uma base de dados sobre consumo de cerveja e analisar o quanto o clima impacta nisto




```
df_beer = pd.read_excel("data/beer_consumption.xlsx")
df_beer.head(10)
```

```
<img src = 'https://media.giphy.com/media/42dsvcMDP3diU/giphy.gif' width=500>
```

```
import sys
import types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share your notebook.
client_58b2fff1efe64e29abda9172ee977f54 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='zVhXqBXZPZvMSzzZVmuMpRuQBECJZg3K7VMQGovteaSE',
    ibm_auth_endpoint="https://iam.ng.bluemix.net/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3-api.us-geo.objectstorage.service.networklayer.com')

body = client_58b2fff1efe64e29abda9172ee977f54.get_object(Bucket='1bpwatsonstudioclaudio-donotdelete-pr-fulmgae1wqh
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

df_beer = pd.read_csv(body)
df_beer.head()
```

Out[4]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
0	42005.0	27.30	23.9	32.5	0.0	False	25461.0
1	42006.0	27.02	24.5	33.5	0.0	False	28972.0

In [5]:
```
1 df_beer.describe()
```

Out[5]:

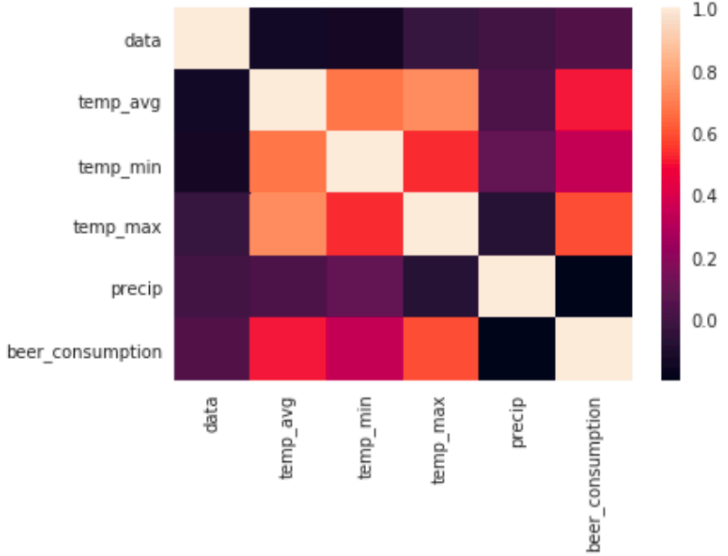
	data	temp_avg	temp_min	temp_max	precip	beer_consumption
count	365.000000	361.000000	363.000000	362.000000	365.000000	365.000000
mean	42187.000000	21.101385	17.396970	26.480663	5.196712	25401.367123
std	105.510663	3.546618	2.969405	4.758531	12.417844	4399.142703
min	42005.000000	0.000000	0.000000	0.000000	0.000000	14343.000000
25%	42096.000000	18.920000	15.200000	23.725000	0.000000	22008.000000
50%	42187.000000	21.360000	17.900000	26.900000	0.000000	24867.000000
75%	42278.000000	23.280000	19.550000	29.400000	3.200000	28631.000000
max	42369.000000	28.860000	24.500000	36.500000	94.800000	37937.000000

In [6]:
```
1 df_beer.count()
```

Out[6]:
```
data                365
temp_avg            361
temp_min            363
temp_max            362
precip              365
weekend             363
beer_consumption    365
dtype: int64
```

In [7]:
```
1 sns.heatmap(df_beer.corr())
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5cd1889668>



In [8]:
```
1 df_beer.corr()
```

Out[8]:

	data	temp_avg	temp_min	temp_max	precip	beer_consumption
data	1.000000	-0.144410	-0.127932	-0.029035	0.007490	0.043541
temp_avg	-0.144410	1.000000	0.678633	0.735339	0.026834	0.503227

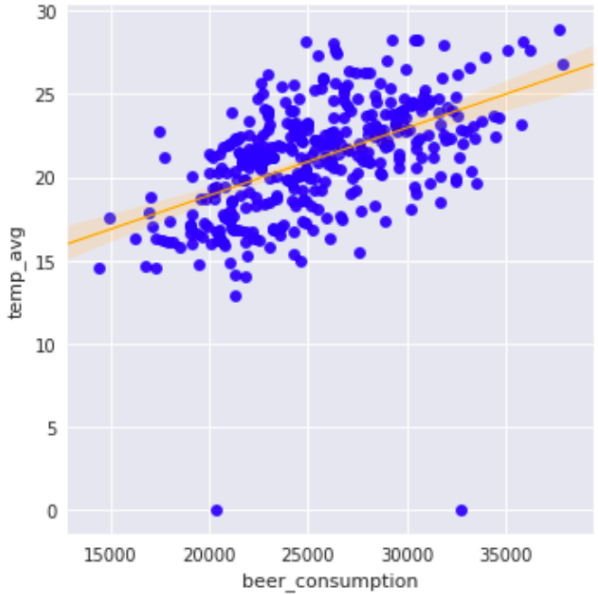
Compreendendo relações e correlações de variáveis

```
1  sns.lmplot()
```

In [13]:

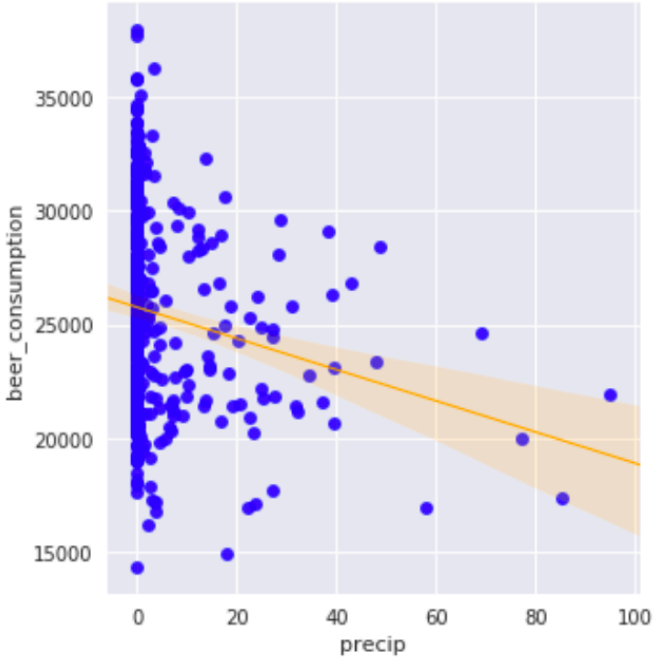
```
1  sns.lmplot("beer_consumption", "temp_avg", df_beer,
2             scatter_kws={"marker":"x", "color":"blue"},
3             line_kws={"linewidth":1, "color": "orange"})
```

Out[13]: <seaborn.axisgrid.FacetGrid at 0x7f5cc1d6ca90>



In [15]:
```
1 sns.lmplot("precip", "beer_consumption", df_beer,
2           scatter_kws={"marker":"x", "color":"blue"},
3           line_kws={"linewidth":1, "color": "orange"})
```

Out[15]: <seaborn.axisgrid.FacetGrid at 0x7f5cc007fb70>



Mapear dados em label para numéricos

In [16]:
```
class_weekend = {True: 1, False: 0}
df_beer["weekend"] = df_beer["weekend"].map(class_weekend)
```

In [17]:
```
df_beer.head(5)
```

Out[17]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
0	42005.0	27.30	23.9	32.5	0.0	0.0	25461.0
1	42006.0	27.02	24.5	33.5	0.0	0.0	28972.0
2	42007.0	24.82	22.4	29.9	0.0	1.0	30814.0
3	42008.0	23.98	21.5	28.6	1.2	1.0	29799.0
4	42009.0	23.82	21.0	28.3	0.0	0.0	28900.0

Busca por dados nulos e inválidos

In [18]:
```
df_beer.isnull().any()
```

Out[18]:
```
data              False
temp_avg           True
temp_min           True
temp_max           True
precip            False
weekend            True
beer_consumption  False
dtype: bool
```

In [19]:
```
df_beer[df_beer["temp_avg"].isnull()]
```

Out[19]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
168	42173.0	NaN	15.8	26.2	0.0	0.0	24534.0
181	42186.0	NaN	16.2	20.5	0.0	0.0	20824.0
309	42314.0	NaN	18.0	22.8	0.0	0.0	20575.0
314	42319.0	NaN	19.8	32.7	0.0	0.0	29569.0

In [20]:
```
df_beer[df_beer["temp_min"].isnull()]
```

Out[20]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
7	42012.0	24.90	NaN	32.8	48.6	0.0	28397.0
116	42121.0	19.82	NaN	24.9	0.0	0.0	21838.0

In [21]:
```
df_beer[df_beer["temp_max"].isnull()]
```

Out[21]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
98	42103.0	19.40	15.9	NaN	0.0	0.0	20298.0
165	42170.0	16.02	13.1	NaN	0.0	0.0	19119.0
237	42242.0	18.92	14.8	NaN	0.6	0.0	23357.0

In [22]:
```
df_beer[df_beer["weekend"].isnull()]
```

Out[22]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
21	42026.0	21.74	19.2	0.0	31.0	NaN	25795.0
27	42032.0	25.68	20.1	29.9	4.9	NaN	22603.0

Temperatura média pode ser calculada entre a temperatura máxima e mínima

In [23]:
```
df_beer_temp_avg_null = df_beer[df_beer["temp_avg"].isnull()].copy()
df_beer_temp_avg_null
```

Out[23]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
168	42173.0	NaN	15.8	26.2	0.0	0.0	24534.0
181	42186.0	NaN	16.2	20.5	0.0	0.0	20824.0
309	42314.0	NaN	18.0	22.8	0.0	0.0	20575.0
314	42319.0	NaN	19.8	32.7	0.0	0.0	29569.0

In [24]:
```
df_beer_temp_avg_null["temp_avg"] = (df_beer["temp_max"] + df_beer["temp_min"])/2
df_beer_temp_avg_null
```

Out[24]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
168	42173.0	21.00	15.8	26.2	0.0	0.0	24534.0
181	42186.0	18.35	16.2	20.5	0.0	0.0	20824.0
309	42314.0	20.40	18.0	22.8	0.0	0.0	20575.0
314	42319.0	26.25	19.8	32.7	0.0	0.0	29569.0

In [25]:
```
df_beer["temp_avg"] = df_beer["temp_avg"].replace(np.nan,(df_beer["temp_max"] + df_beer["temp_min"])/2)
```

In [26]:
```
df_beer.loc[[168,181,309,314]]
```

Out[26]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
168	42173.0	21.00	15.8	26.2	0.0	0.0	24534.0
181	42186.0	18.35	16.2	20.5	0.0	0.0	20824.0
309	42314.0	20.40	18.0	22.8	0.0	0.0	20575.0
314	42319.0	26.25	19.8	32.7	0.0	0.0	29569.0

In [27]:
```
df_beer["temp_min"] = df_beer["temp_min"].replace(np.nan,df_beer["temp_min"].mean())
```

In [28]:
```
df_beer.loc[[7,116]]
```

Out[28]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
7	42012.0	24.90	17.39697	32.8	48.6	0.0	28397.0
116	42121.0	19.82	17.39697	24.9	0.0	0.0	21838.0

In [29]:
```
df_beer["temp_max"] = df_beer["temp_max"].replace(np.nan,df_beer["temp_max"].mean())
```

In [30]:
```
df_beer.loc[[98, 165, 237]]
```

Out[30]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
98	42103.0	19.40	15.9	26.480663	0.0	0.0	20298.0
165	42170.0	16.02	13.1	26.480663	0.0	0.0	19119.0
237	42242.0	18.92	14.8	26.480663	0.6	0.0	23357.0

In [31]:
```
df_beer["weekend"] = df_beer["weekend"].replace(np.nan,df_beer["weekend"].mode()[0])
```

In [32]:
```
df_beer.loc[[21,27]]
```

Out[32]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
21	42026.0	21.74	19.2	0.0	31.0	0.0	25795.0
27	42032.0	25.68	20.1	29.9	4.9	0.0	22603.0

In [33]:
```
df_beer["weekend"].mode()[0]
```

Out[33]:
```
0.0
```

In [34]:
```
df_beer.isnull().any()
```

Out[34]:
```
data                False
temp_avg            False
temp_min            False
temp_max            False
precip              False
weekend             False
beer_consumption    False
dtype: bool
```

Verifcar ocorrência de números iguais a 0, que seria inválido

In [35]:
```
(df_beer == 0).any()
```

Out[35]:
```
data                False
temp_avg             True
temp_min             True
temp_max             True
precip               True
weekend              True
beer_consumption    False
dtype: bool
```

In [36]:
```
df_beer[df_beer["temp_avg"] == 0]
```

Out[36]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
323	42328.0	0.0	19.6	27.0	6.8	0.0	20332.0
339	42344.0	0.0	20.6	28.0	0.1	1.0	32780.0

In [37]:
```
df_beer[df_beer["temp_min"] == 0]
```

Out[37]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
13	42018.0	25.96	0.0	34.0	1.6	0.0	31825.0

In [38]:
```
df_beer[df_beer["temp_max"] == 0]
```

Out[38]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption

In [39]:
```
1 df_beer[df_beer["precip"] == 0].head(10)
```

Out[39]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
0	42005.0	27.30	23.9	32.5	0.0	0.0	25461.0
1	42006.0	27.02	24.5	33.5	0.0	0.0	28972.0
2	42007.0	24.82	22.4	29.9	0.0	1.0	30814.0
4	42009.0	23.82	21.0	28.3	0.0	0.0	28900.0
6	42011.0	24.00	19.5	33.7	0.0	0.0	29732.0
9	42014.0	26.76	22.1	34.2	0.0	1.0	37937.0
11	42016.0	25.96	21.4	35.4	0.0	0.0	25743.0
16	42021.0	28.86	22.0	35.8	0.0	1.0	37690.0
17	42022.0	28.26	23.4	35.6	0.0	1.0	30524.0
20	42025.0	25.32	22.7	30.9	0.0	0.0	29130.0

In [40]:
```
1 df_beer[df_beer["weekend"] == 0].head(10)
```

Out[40]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
0	42005.0	27.30	23.90000	32.5	0.0	0.0	25461.0
1	42006.0	27.02	24.50000	33.5	0.0	0.0	28972.0
4	42009.0	23.82	21.00000	28.3	0.0	0.0	28900.0
5	42010.0	23.78	20.10000	30.5	12.2	0.0	28218.0
6	42011.0	24.00	19.50000	33.7	0.0	0.0	29732.0
7	42012.0	24.90	17.39697	32.8	48.6	0.0	28397.0
8	42013.0	28.20	21.90000	34.0	4.4	0.0	24886.0
11	42016.0	25.96	21.40000	35.4	0.0	0.0	25743.0

Utilizar mesma estratégia anterior para os demais atributos.

In [41]:
```
df_beer["temp_avg"] = df_beer["temp_avg"].replace(0,(df_beer["temp_max"] + df_beer["temp_min"])/2)
df_beer["temp_min"] = df_beer["temp_min"].replace(0,df_beer["temp_min"].mean())
df_beer["temp_max"] = df_beer["temp_max"].replace(0,df_beer["temp_max"].mean())
```

In [42]:
```
(df_beer == 0).any()
```

Out[42]:
```
data               False
temp_avg           False
temp_min           False
temp_max           False
precip              True
weekend             True
beer_consumption   False
dtype: bool
```

Seperando dados de treinamento e teste

In [43]:
```
from sklearn.model_selection import train_test_split

feature_col_names = ['temp_max', 'precip', 'weekend']
predicted_class_names = ['beer_consumption']

X = df_beer[feature_col_names].values
y = df_beer[predicted_class_names].values
split_test_size = 0.30

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=split_test_size, random_state=42)
```

In [44]:
```
print("{0:0.2f}% in training set".format((len(X_train)/len(df_beer.index)) * 100))
print("{0:0.2f}% in test set".format((len(X_test)/len(df_beer.index)) * 100))
```

```
69.86% in training set
30.14% in test set
```

In [45]:
```
from sklearn import linear_model

lr_model = linear_model.LinearRegression()
lr_model.fit(X_train, y_train.ravel())
```

Out[45]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

Testando valores par verificar a performance

In [46]:
```
from sklearn.metrics import mean_squared_error, r2_score

y_pred = lr_model.predict(X_test)

print('R2 score: %.2f' % r2_score(y_test, y_pred))
print('R2 score: %.2f' % lr_model.score(X_test, y_test))
```

```
R2 score: 0.74
R2 score: 0.74
```

R2 score está na faixa esperada de >= 70%

Simulação de um caso onde se espera maior consumo, clima quente, 35 graus C, sem chuva e no final de semana.

In [47]:
```
predict_value = [[35, 0, 1]]
lr_model.predict(predict_value)
```

Out[47]:
```
array([ 35092.25246658])
```

Simulação de um caso onde se espera um consumo menor que o de cima, clima quente, 35 graus C, sem chuva

In [48]:
```
1 predict_value = [[35, 0, 0]]
2 lr_model.predict(predict_value)
```

Out[48]:
```
array([ 29753.3647769])
```

Simulação de um caso onde se espera um consumo menor que o de cima, clima quente, 35 graus C, com chuva

In [49]:
```
1 predict_value = [[35, 20, 0]]
2 lr_model.predict(predict_value)
```

Out[49]:
```
array([ 28689.99450697])
```

Pior cenário, baixo consumo, frio (10 graus C), chovendo e em dia de semana.

In [50]:
```
1 predict_value = [[10, 20, 0]]
2 lr_model.predict(predict_value)
```

Out[50]:
```
array([ 12008.88877202])
```

Salvando modelo

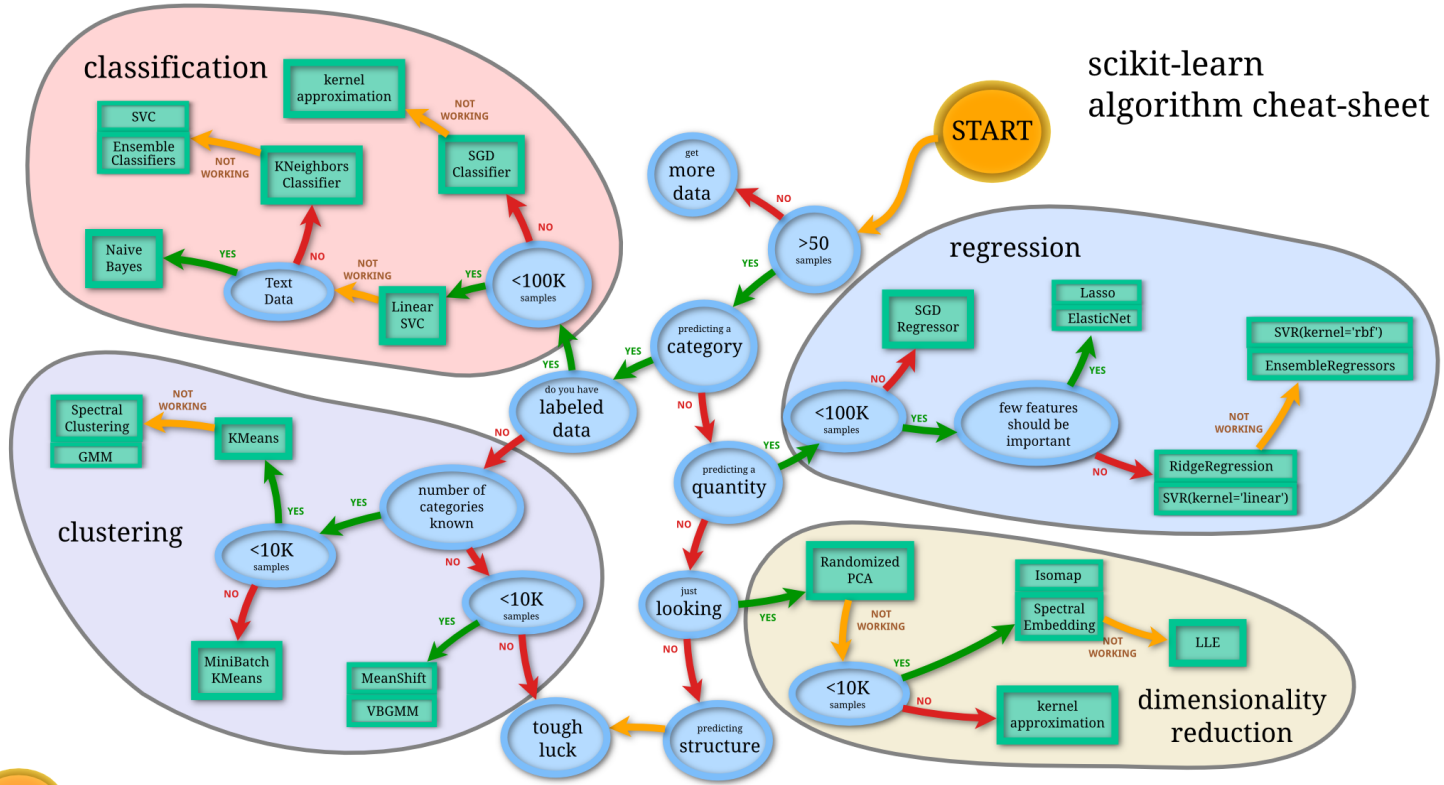
```
from sklearn.externals import joblib
joblib.dump(lr_model, 'models/lr_model.pkl')
```

```
lr_model_loaded = joblib.load('models/lr_model.pkl')
```

```
predict_value = [[10, 20, 0]]
lr_model_loaded.predict(predict_value)
```

Por Claudio Pinheiro https://www.linkedin.com/in/claudio-pinheiro-cloud/

Técnicas por Abordagem



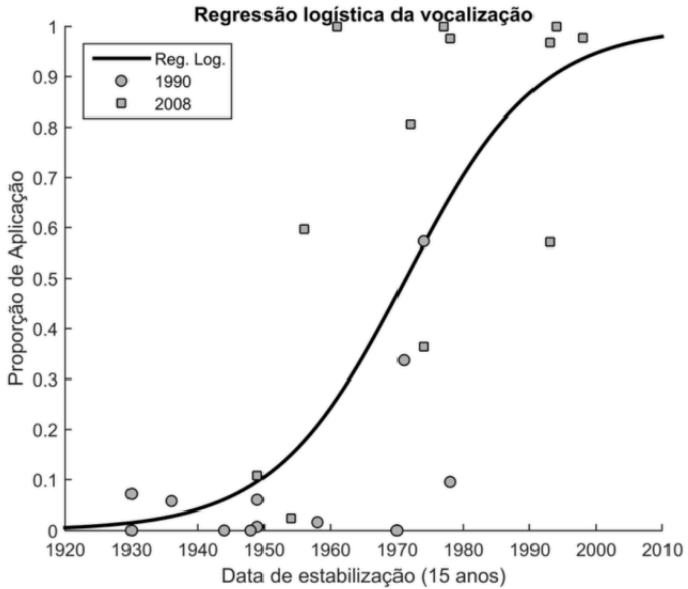
Back

Regressão

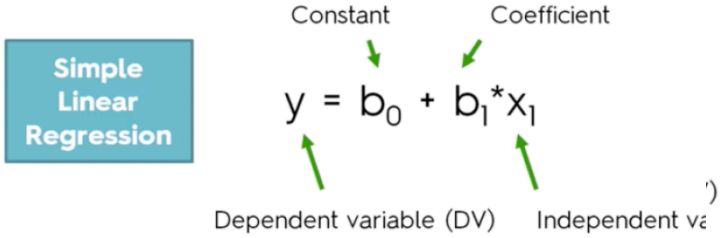
O modelo de **regressão logística** é semelhante ao modelo de **regressão linear**. No entanto, no modelo **logístico** a variável resposta é binária. Uma variável **binária** assume dois valores, como por exemplo, e denominados "fracasso" e "sucesso", respectivamente.

Neste caso, "sucesso" é o evento de interesse.

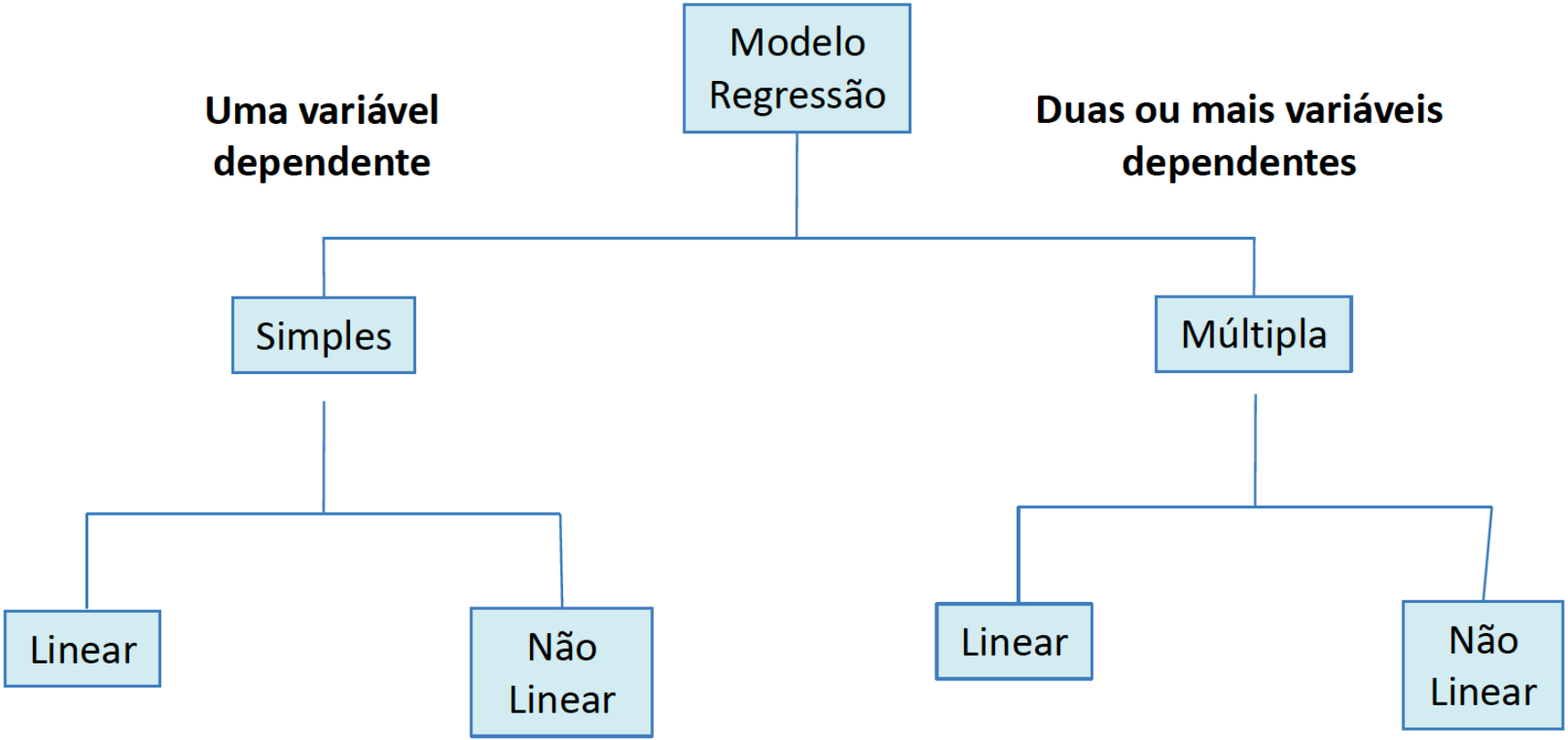
- ✓ **Regressão Logística Multinomial**
- ✓ **Regressão Logística Ordinal**



Regressão Linear

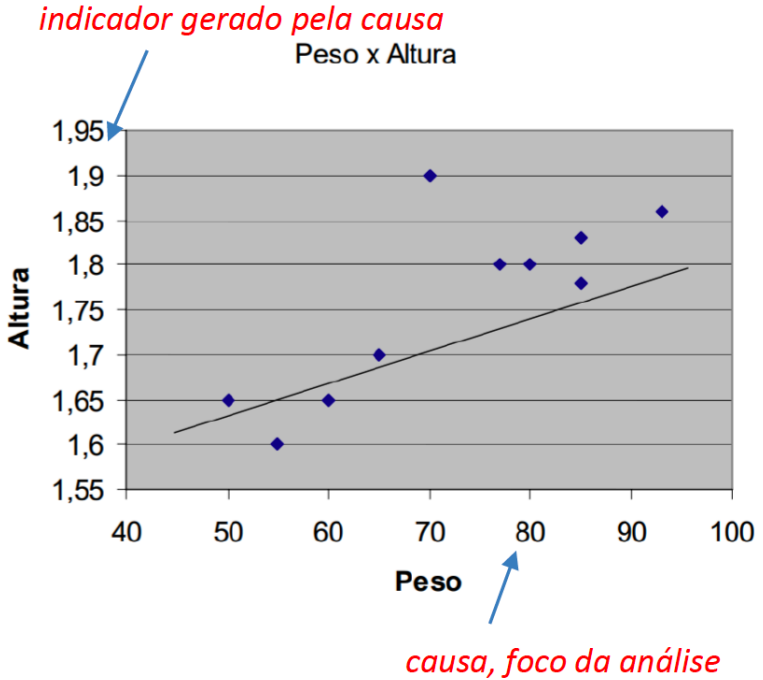


- ✓ A variável que estamos prevendo é chamada de variável de critério (dependente) e é chamada de *"y"*.
- ✓ A variável na qual estamos baseando nossas previsões é chamada de variável preditora (independente) e é chamada de *"x"*.
- ✓ Quando há apenas uma variável preditora, o método de predição é chamado de regressão simples.

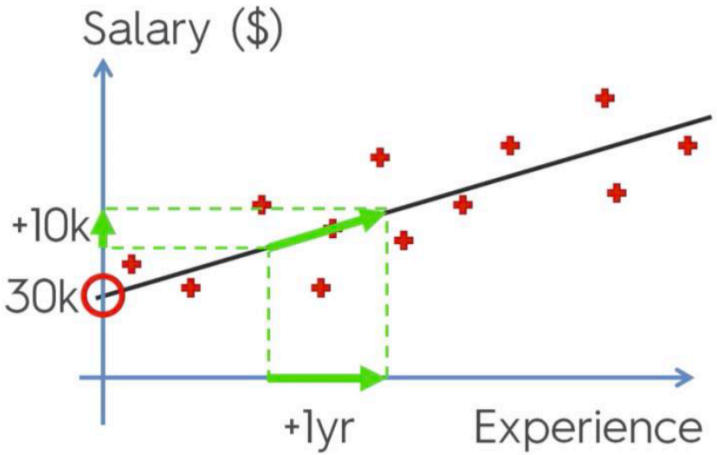

Modelo Regressão
Uma variável dependente
Duas ou mais variáveis dependentes
Simples
Múltipla
Linear
Não Linear
Linear
Não Linear

Eixo ´x´ *- Variável que é alterada por uma modificação no processo (variável independente)*

Eixo ´y´ *- Variável que pode mudar de acordo com a mudança da variável em ´x´ (variável dependente)*



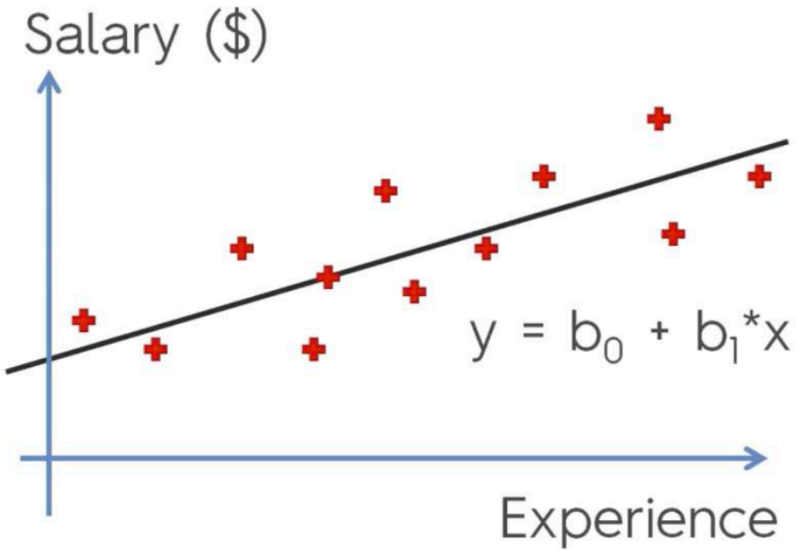
	YearsExperience	Salary
2	1.3	46205
4	2.0	43525
5	2.2	39891
8	3.2	54445
11	3.9	63218
16	4.9	67938
20	6.0	93940
21	6.8	91738
24	8.2	113812
26	9.0	105582



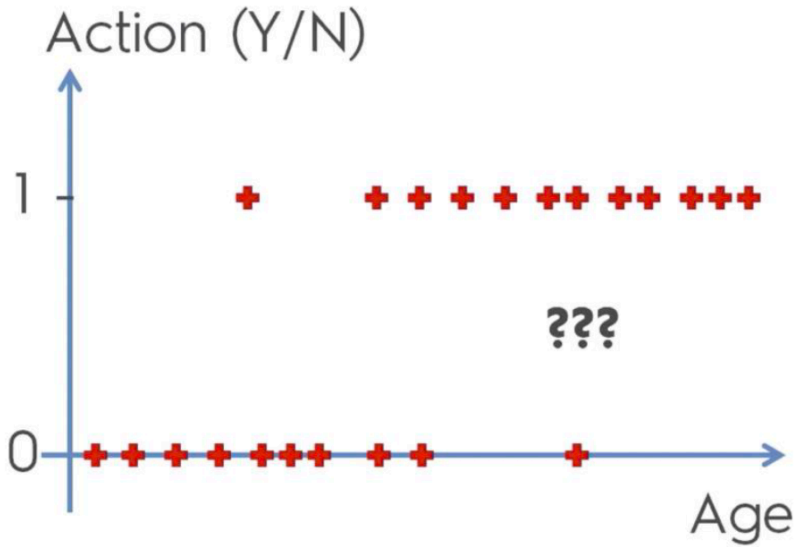
$$y = b_0 + b_1 * x$$

Salary = b_0 + b_1 *Experience

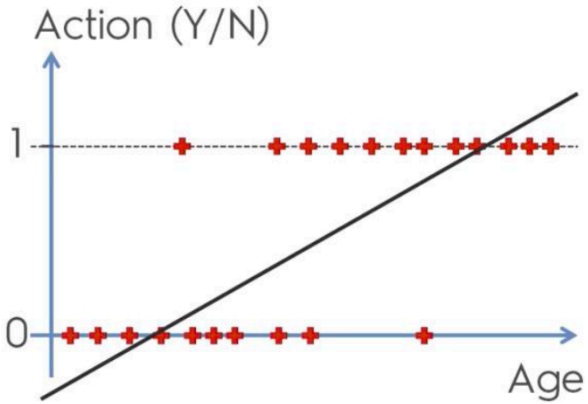
Regressão Linear – Já sabemos!

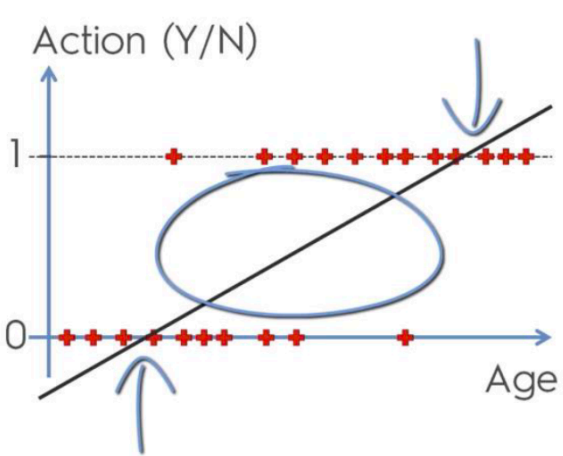


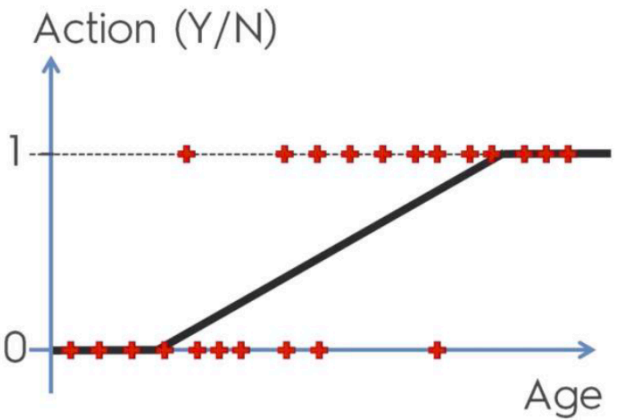
E agora???

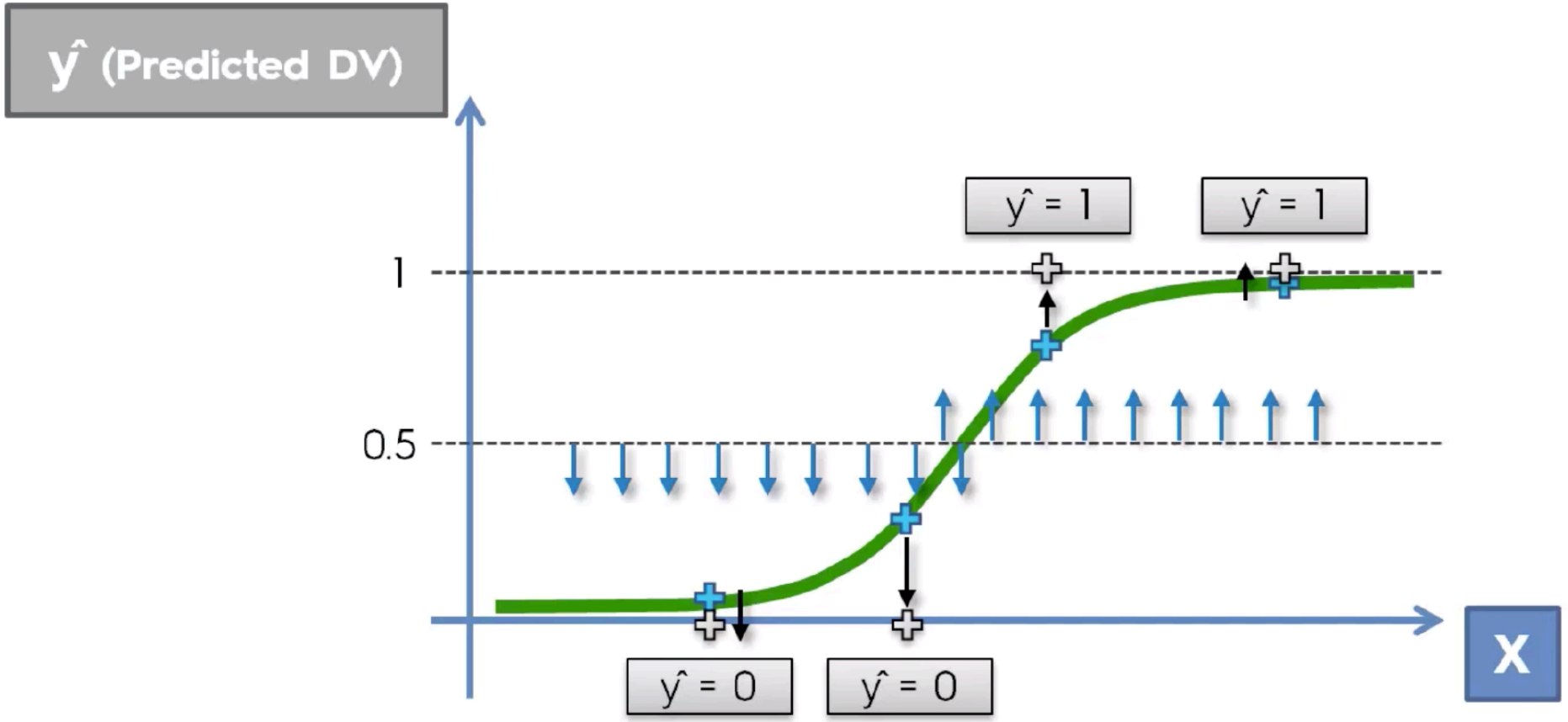


Regressão Logística








ŷ (Predicted DV)
ŷ = 1
ŷ = 1
1
0.5
ŷ = 0
ŷ = 0
X

	User.ID	Gender	Age	EstimatedSalary	Purchased
1	15624510	Male	19	19000	0
2	15810944	Male	35	20000	0
3	15668575	Female	26	43000	0
4	15603246	Female	27	57000	0
5	15804002	Male	19	76000	0
6	15728773	Male	27	58000	0
7	15598044	Female	27	84000	0
8	15694829	Female	32	150000	1
9	15600575	Male	25	33000	0
10	15727311	Female	35	65000	0
11	15570769	Female	26	80000	0
12	15606274	Female	26	52000	0
13	15746139	Male	20	86000	0
14	15704987	Male	32	18000	0

Base de dados:

- 400 observações;
- 5 variáveis.

Pesquisa feita nas redes sociais marcando as pessoas interessadas em um modelo de carro SUV.

Importando os pacotes

```
import pandas as pd
import matplotlib.pyplot as pl
import numpy as np
```

Importando a base de dados

```
data = pd.read_csv("data/Social_Network_Ads.csv")
data.head()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

Analisando os dados

```
#Analisando os dados
data.describe()
```

	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	400.000000	400.000000	400.000000
mean	1.569154e+07	37.655000	69742.500000	0.357500
std	7.165832e+04	10.482877	34096.960282	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	29.750000	43000.000000	0.000000
50%	1.569434e+07	37.000000	70000.000000	0.000000
75%	1.575036e+07	46.000000	88000.000000	1.000000
max	1.581524e+07	60.000000	150000.000000	1.000000

```
#Correlação entre as variáveis
data.corr()
```

	User ID	Age	EstimatedSalary	Purchased
User ID	1.000000	-0.000721	0.071097	0.007120
Age	-0.000721	1.000000	0.155238	0.622454
EstimatedSalary	0.071097	0.155238	1.000000	0.362083
Purchased	0.007120	0.622454	0.362083	1.000000

```
#Facilitando a visualização
data.corr().style.format("{:.2}").background_gradient()
```

	User ID	Age	EstimatedSalary	Purchased
User ID	1.0	-0.00072	0.071	0.0071
Age	-0.00072	1.0	0.16	0.62
EstimatedSalary	0.071	0.16	1.0	0.36
Purchased	0.0071	0.62	0.36	1.0

```
data["Age"].corr(data["Purchased"])
```

```
0.62245419888452913
```

```
data["EstimatedSalary"].corr(data["Purchased"])
```

```
0.36208302580467916
```

Separando a base - Variáveis dependentes e Independentes

```
X = data.iloc[ : , [2,3]].values
pd.DataFrame(X).head()
```

	0	1
0	19	19000
1	35	20000
2	26	43000
3	27	57000
4	19	76000

```
y = data.iloc[ : , 4].values
pd.DataFrame(y).head()
```

	0
0	0
1	0
2	0
3	0
4	0

Processo básico de modelagem

É composto por 5 passos:

- Fazer a pergunta certa
- Preparar os dados
- Selecionar o algoritmo
- Treinar o modelo
- Testar o modelo

Preparando os dados

Antes de iniciar a preparação dos dados necessitamos importar um conjunto de bibliotecas referente a criação de estruturas de dados, entendimento numérico dos tipos de dados além de bibliotecas gráficas.

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

plt.style.use('seaborn')
```

Separando a base de dados em dois conjuntos: treinamento e teste

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

```
len(data)
```

```
400
```

```
len(X_train)
```

```
300
```

```
len(X_test)
```

```
100
```

Escala das variáveis

```
from sklearn.preprocessing import StandardScaler
```

```
ss_X = StandardScaler()
X_train = ss_X.fit_transform(X_train)
X_test = ss_X.fit_transform(X_test)
```

```
pd.DataFrame(X_train).head()
```

	0	1
0	0.581649	-0.886707
1	-0.606738	1.461738
2	-0.012544	-0.567782
3	-0.606738	1.896635
4	1.373907	-1.408584

```
from sklearn.linear_model import LogisticRegression
```

```
regressaoLogistica = LogisticRegression(random_state=0)
regressaoLogistica.fit(X_train, y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=0, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

```
regressaoLogistica.score(X_train, y_train)
```

```
0.82333333333333336
```

Utilizando o modelo criado para fazer a predição da variável "target"

```
y_pred = regressaoLogistica.predict(X_test)
y_pred
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 1])
```

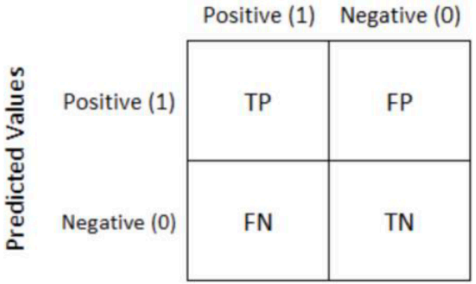
```
pd.DataFrame({'Predição': y_pred, 'Real': y_test})
```

	Predição	Real
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	1	1
8	0	0
9	1	0

Como avaliar o modelo?

Matriz de Confusão

		Valor Observado (valor verdadeiro)	
		Y=1	Y=0
Valor Predito	Y=1	VP (verdadeiro positivo)	FP (falso positivo)
	Y=0	FN (falso negativo)	VN (verdadeiro negativo)



- ✓ TP – Verdadeiro Positivo
  - ✓ Predição positiva & Real positivo
- ✓ FP – Falso Positivo
  - ✓ Predição positiva & Real negativo
- ✓ TN – Verdadeiro Negativo
  - ✓ Predição negativa & Real negativo
- ✓ FN – Falso Negativo
  - ✓ Predição negativa & Real positivo

Accuracy = (TP +TN) / (TP + FP +TN +FN)

Quantidade de acertos no resultado total

precision = true positive / (true positive + false positive)

Predição positiva pela quantidade total de predição

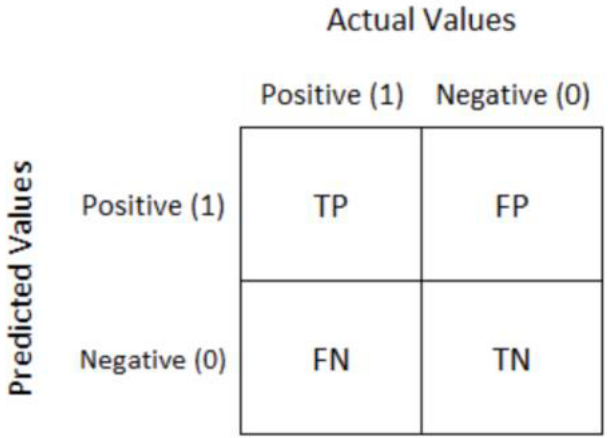
$$Recall = \frac{TP}{TP + FN}$$

$$F\text{ - measure} = \frac{2*Recall*Precision}{Recall + Precision}$$

Validando os dados - Matriz de Confusão

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[63,  5],
       [ 7, 25]])
```




```
#Acurácia do Modelo
(cm[0,0]+cm[1,1])/len(y_pred)
```

```
0.88
```

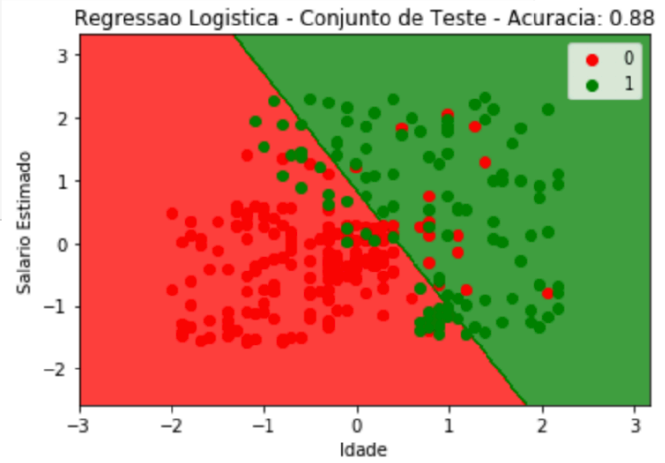
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
0.88
```

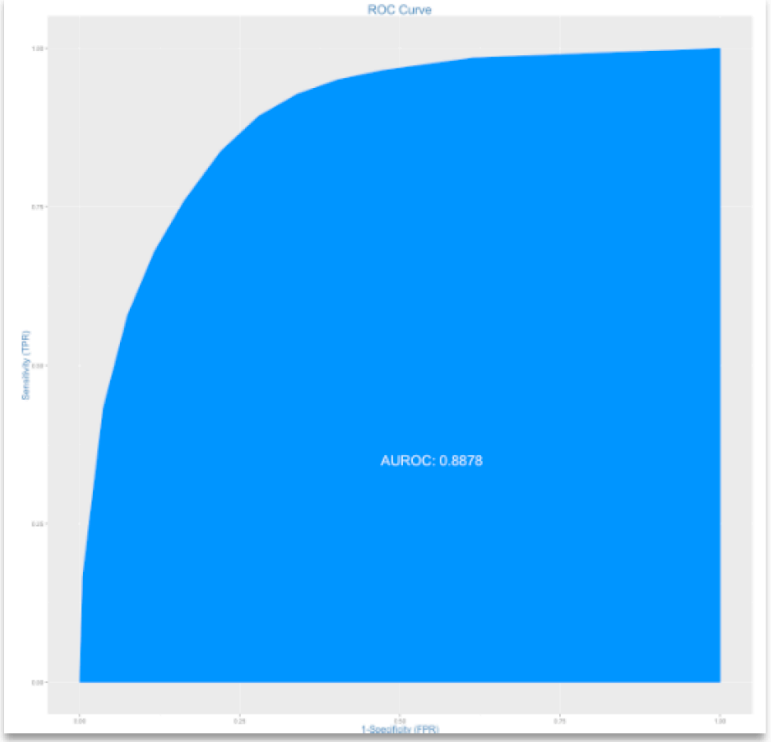
Visualisando o Resultado - Conjunto de Teste

```
from matplotlib.colors import ListedColormap
```

```
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, regressaoLogistica.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Regressao Logistica - Conjunto de Teste - Acuracia: ' + str(acuracia))
plt.xlabel('Idade')
plt.ylabel('Salario Estimado')
plt.legend()
plt.show()
```



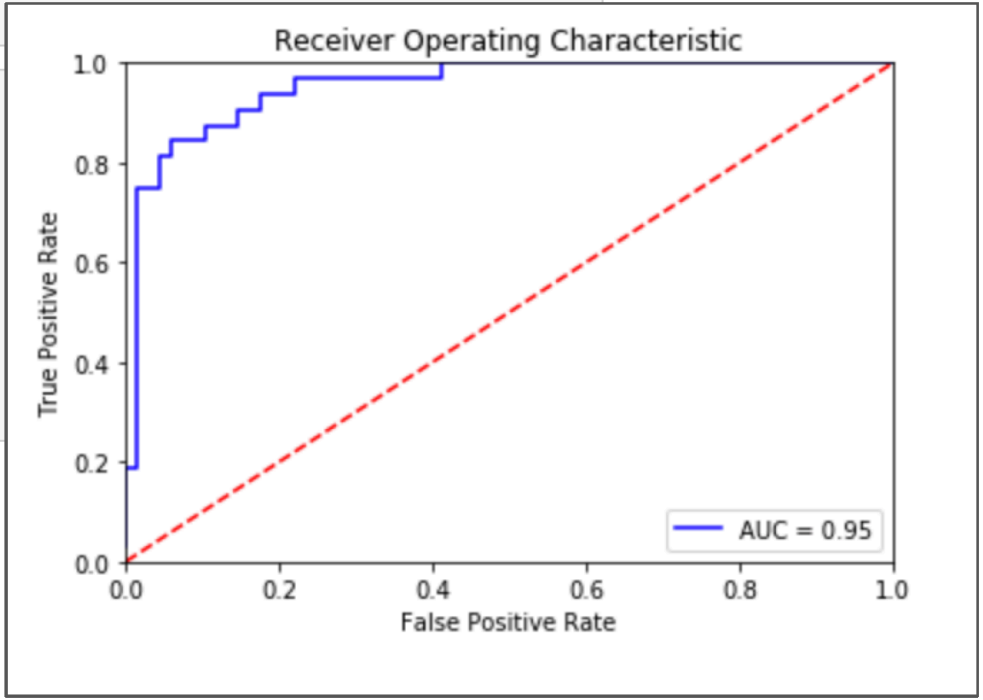
Característica de Operação do Receptor (COR) ou Receiver Operating Characteristic (ROC), ilustra o desempenho de um sistema classificador binário.



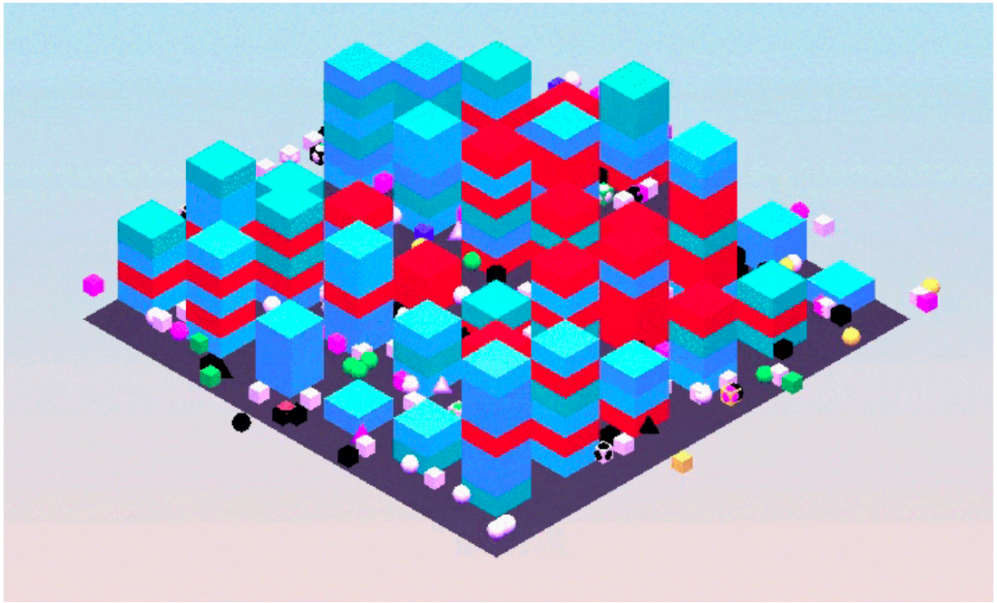
- ✓ Este indicador traça a porcentagem de positivos verdadeiros previstos com precisão por um determinado modelo logit à medida que o limite de probabilidade de previsão é reduzido de 1 para 0. Para um um bom modelo, a curva deve subir acentuadamente, ou seja, quanto maior a área sob a curva ROC, melhor a capacidade preditiva do modelo.
- ✓ **AUC – área under the curve – mede o desempenho.**

```
import sklearn.metrics as metrics
probs = classifier.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)
```

```
# method I: plt
import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



Processo de modelagem com Machine Learning



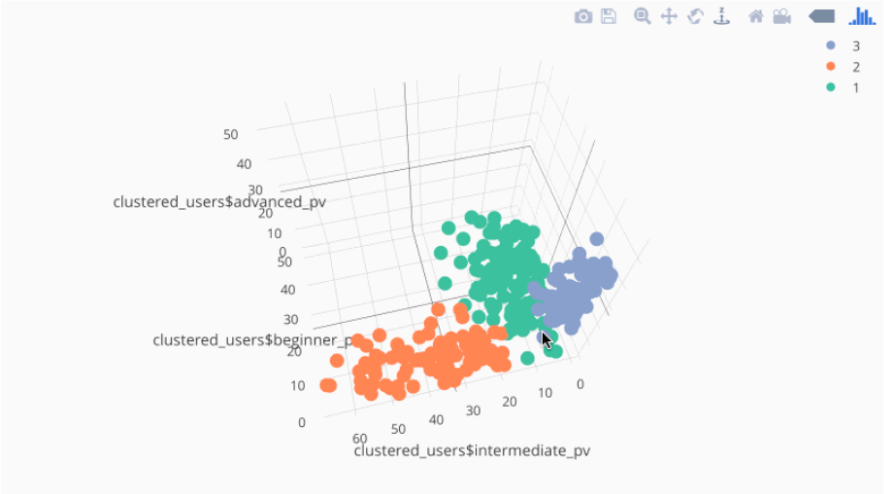
Neste notebook vamos apresentar conceitos e códigos referentes a um processo de modelagem do inicio ao fim, testando o modelo desenvolvido.

Ainda que simples serve como uma referência para modelagens futuras.

O processo mais adequado e completo será apresentado durante as aulas e trata-se do CRISP-DM

<img src = 'https://frnsys.com/assets/hosny/disparity.gif' width =600>

Agora vamos carregar uma base de dados sobre consumo de cerveja e analisar o quanto o clima impacta nisto




```
df_beer = pd.read_excel("data/beer_consumption.xlsx")
df_beer.head(10)
```

```
<img src = 'https://media.giphy.com/media/42dsvcMDP3diU/giphy.gif' width=500>
```

```
import sys
import types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share your notebook.
client_58b2fff1efe64e29abda9172ee977f54 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='zVhXqBXZPZvMSzzZVmuMpRuQBECJZg3K7VMQGovteaSE',
    ibm_auth_endpoint="https://iam.ng.bluemix.net/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3-api.us-geo.objectstorage.service.networklayer.com')

body = client_58b2fff1efe64e29abda9172ee977f54.get_object(Bucket='1bpwatsonstudioclaudio-donotdelete-pr-fulmgae1wqh
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

df_beer = pd.read_csv(body)
df_beer.head()
```

Out[4]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
0	42005.0	27.30	23.9	32.5	0.0	False	25461.0
1	42006.0	27.02	24.5	33.5	0.0	False	28972.0

In [5]:
```
1 df_beer.describe()
```

Out[5]:

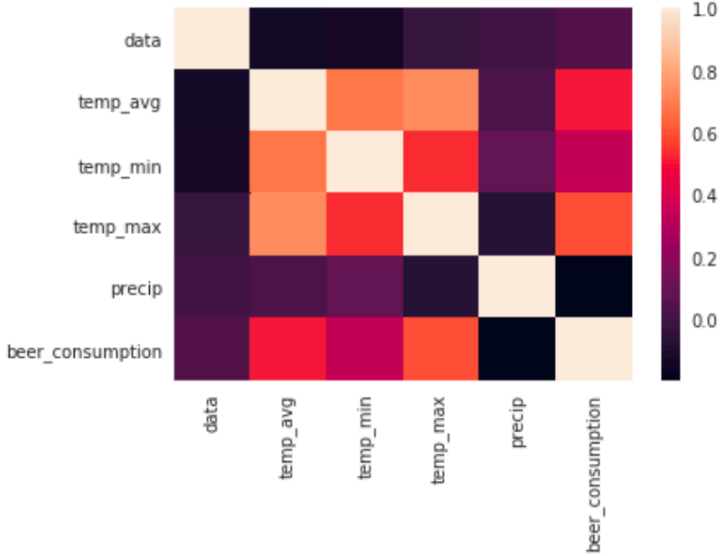
	data	temp_avg	temp_min	temp_max	precip	beer_consumption
count	365.000000	361.000000	363.000000	362.000000	365.000000	365.000000
mean	42187.000000	21.101385	17.396970	26.480663	5.196712	25401.367123
std	105.510663	3.546618	2.969405	4.758531	12.417844	4399.142703
min	42005.000000	0.000000	0.000000	0.000000	0.000000	14343.000000
25%	42096.000000	18.920000	15.200000	23.725000	0.000000	22008.000000
50%	42187.000000	21.360000	17.900000	26.900000	0.000000	24867.000000
75%	42278.000000	23.280000	19.550000	29.400000	3.200000	28631.000000
max	42369.000000	28.860000	24.500000	36.500000	94.800000	37937.000000

In [6]:
```
1 df_beer.count()
```

Out[6]:
```
data                365
temp_avg            361
temp_min            363
temp_max            362
precip              365
weekend             363
beer_consumption    365
dtype: int64
```

In [7]:
```
1 sns.heatmap(df_beer.corr())
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5cd1889668>



In [8]:
```
1 df_beer.corr()
```

Out[8]:

	data	temp_avg	temp_min	temp_max	precip	beer_consumption
data	1.000000	-0.144410	-0.127932	-0.029035	0.007490	0.043541
temp_avg	-0.144410	1.000000	0.678633	0.735339	0.026834	0.503227

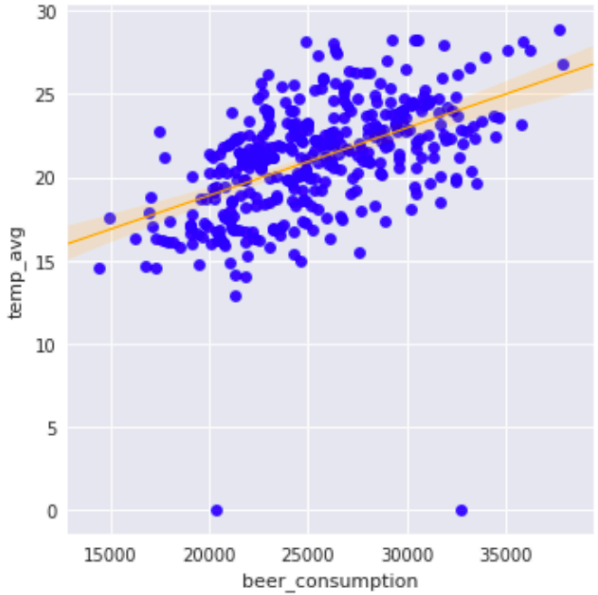
Compreendendo relações e correlações de variáveis

```
1  sns.lmplot()
```

In [13]:

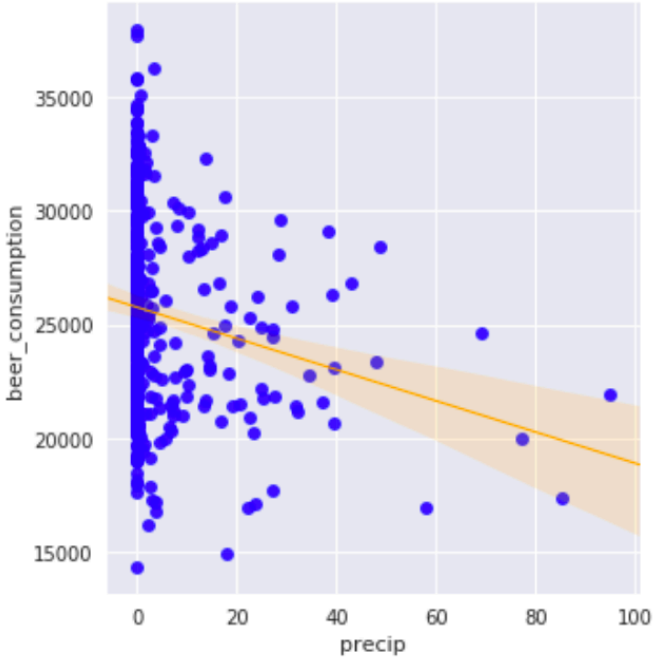
```
1  sns.lmplot("beer_consumption", "temp_avg", df_beer,
2             scatter_kws={"marker":"x", "color":"blue"},
3             line_kws={"linewidth":1, "color": "orange"})
```

Out[13]: <seaborn.axisgrid.FacetGrid at 0x7f5cc1d6ca90>



In [15]:
```
1 sns.lmplot("precip", "beer_consumption", df_beer,
2           scatter_kws={"marker":"x", "color":"blue"},
3           line_kws={"linewidth":1, "color": "orange"})
```

Out[15]: <seaborn.axisgrid.FacetGrid at 0x7f5cc007fb70>



Mapear dados em label para numéricos

In [16]:
```
class_weekend = {True: 1, False: 0}
df_beer["weekend"] = df_beer["weekend"].map(class_weekend)
```

In [17]:
```
df_beer.head(5)
```

Out[17]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
0	42005.0	27.30	23.9	32.5	0.0	0.0	25461.0
1	42006.0	27.02	24.5	33.5	0.0	0.0	28972.0
2	42007.0	24.82	22.4	29.9	0.0	1.0	30814.0
3	42008.0	23.98	21.5	28.6	1.2	1.0	29799.0
4	42009.0	23.82	21.0	28.3	0.0	0.0	28900.0

Busca por dados nulos e inválidos

In [18]:
```
df_beer.isnull().any()
```

Out[18]:
```
data              False
temp_avg           True
temp_min           True
temp_max           True
precip            False
weekend            True
beer_consumption  False
dtype: bool
```

In [19]:
```
df_beer[df_beer["temp_avg"].isnull()]
```

Out[19]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
168	42173.0	NaN	15.8	26.2	0.0	0.0	24534.0
181	42186.0	NaN	16.2	20.5	0.0	0.0	20824.0
309	42314.0	NaN	18.0	22.8	0.0	0.0	20575.0
314	42319.0	NaN	19.8	32.7	0.0	0.0	29569.0

In [20]:
```
df_beer[df_beer["temp_min"].isnull()]
```

Out[20]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
7	42012.0	24.90	NaN	32.8	48.6	0.0	28397.0
116	42121.0	19.82	NaN	24.9	0.0	0.0	21838.0

In [21]:
```
df_beer[df_beer["temp_max"].isnull()]
```

Out[21]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
98	42103.0	19.40	15.9	NaN	0.0	0.0	20298.0
165	42170.0	16.02	13.1	NaN	0.0	0.0	19119.0
237	42242.0	18.92	14.8	NaN	0.6	0.0	23357.0

In [22]:
```
df_beer[df_beer["weekend"].isnull()]
```

Out[22]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
21	42026.0	21.74	19.2	0.0	31.0	NaN	25795.0
27	42032.0	25.68	20.1	29.9	4.9	NaN	22603.0

Temperatura média pode ser calculada entre a temperatura máxima e mínima

In [23]:
```
df_beer_temp_avg_null = df_beer[df_beer["temp_avg"].isnull()].copy()
df_beer_temp_avg_null
```

Out[23]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
168	42173.0	NaN	15.8	26.2	0.0	0.0	24534.0
181	42186.0	NaN	16.2	20.5	0.0	0.0	20824.0
309	42314.0	NaN	18.0	22.8	0.0	0.0	20575.0
314	42319.0	NaN	19.8	32.7	0.0	0.0	29569.0

In [24]:
```
df_beer_temp_avg_null["temp_avg"] = (df_beer["temp_max"] + df_beer["temp_min"])/2
df_beer_temp_avg_null
```

Out[24]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
168	42173.0	21.00	15.8	26.2	0.0	0.0	24534.0
181	42186.0	18.35	16.2	20.5	0.0	0.0	20824.0
309	42314.0	20.40	18.0	22.8	0.0	0.0	20575.0
314	42319.0	26.25	19.8	32.7	0.0	0.0	29569.0

In [25]:
```
df_beer["temp_avg"] = df_beer["temp_avg"].replace(np.nan,(df_beer["temp_max"] + df_beer["temp_min"])/2)
```

In [26]:
```
df_beer.loc[[168,181,309,314]]
```

Out[26]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
168	42173.0	21.00	15.8	26.2	0.0	0.0	24534.0
181	42186.0	18.35	16.2	20.5	0.0	0.0	20824.0
309	42314.0	20.40	18.0	22.8	0.0	0.0	20575.0
314	42319.0	26.25	19.8	32.7	0.0	0.0	29569.0

In [27]:
```
df_beer["temp_min"] = df_beer["temp_min"].replace(np.nan,df_beer["temp_min"].mean())
```

In [28]:
```
df_beer.loc[[7,116]]
```

Out[28]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
7	42012.0	24.90	17.39697	32.8	48.6	0.0	28397.0
116	42121.0	19.82	17.39697	24.9	0.0	0.0	21838.0

In [29]:
```
df_beer["temp_max"] = df_beer["temp_max"].replace(np.nan,df_beer["temp_max"].mean())
```

In [30]:
```
df_beer.loc[[98, 165, 237]]
```

Out[30]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
98	42103.0	19.40	15.9	26.480663	0.0	0.0	20298.0
165	42170.0	16.02	13.1	26.480663	0.0	0.0	19119.0
237	42242.0	18.92	14.8	26.480663	0.6	0.0	23357.0

In [31]:
```
df_beer["weekend"] = df_beer["weekend"].replace(np.nan,df_beer["weekend"].mode()[0])
```

In [32]:
```
df_beer.loc[[21,27]]
```

Out[32]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
21	42026.0	21.74	19.2	0.0	31.0	0.0	25795.0
27	42032.0	25.68	20.1	29.9	4.9	0.0	22603.0

In [33]:
```
df_beer["weekend"].mode()[0]
```

Out[33]:
```
0.0
```

In [34]:
```
df_beer.isnull().any()
```

Out[34]:
```
data                False
temp_avg            False
temp_min            False
temp_max            False
precip              False
weekend             False
beer_consumption    False
dtype: bool
```

Verifcar ocorrência de números iguais a 0, que seria inválido

In [35]:
```
(df_beer == 0).any()
```

Out[35]:
```
data                False
temp_avg             True
temp_min             True
temp_max             True
precip               True
weekend              True
beer_consumption    False
dtype: bool
```

In [36]:
```
df_beer[df_beer["temp_avg"] == 0]
```

Out[36]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
323	42328.0	0.0	19.6	27.0	6.8	0.0	20332.0
339	42344.0	0.0	20.6	28.0	0.1	1.0	32780.0

In [37]:
```
df_beer[df_beer["temp_min"] == 0]
```

Out[37]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
13	42018.0	25.96	0.0	34.0	1.6	0.0	31825.0

In [38]:
```
df_beer[df_beer["temp_max"] == 0]
```

Out[38]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption

In [39]:
```
1 df_beer[df_beer["precip"] == 0].head(10)
```

Out[39]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
0	42005.0	27.30	23.9	32.5	0.0	0.0	25461.0
1	42006.0	27.02	24.5	33.5	0.0	0.0	28972.0
2	42007.0	24.82	22.4	29.9	0.0	1.0	30814.0
4	42009.0	23.82	21.0	28.3	0.0	0.0	28900.0
6	42011.0	24.00	19.5	33.7	0.0	0.0	29732.0
9	42014.0	26.76	22.1	34.2	0.0	1.0	37937.0
11	42016.0	25.96	21.4	35.4	0.0	0.0	25743.0
16	42021.0	28.86	22.0	35.8	0.0	1.0	37690.0
17	42022.0	28.26	23.4	35.6	0.0	1.0	30524.0
20	42025.0	25.32	22.7	30.9	0.0	0.0	29130.0

In [40]:
```
1 df_beer[df_beer["weekend"] == 0].head(10)
```

Out[40]:

	data	temp_avg	temp_min	temp_max	precip	weekend	beer_consumption
0	42005.0	27.30	23.90000	32.5	0.0	0.0	25461.0
1	42006.0	27.02	24.50000	33.5	0.0	0.0	28972.0
4	42009.0	23.82	21.00000	28.3	0.0	0.0	28900.0
5	42010.0	23.78	20.10000	30.5	12.2	0.0	28218.0
6	42011.0	24.00	19.50000	33.7	0.0	0.0	29732.0
7	42012.0	24.90	17.39697	32.8	48.6	0.0	28397.0
8	42013.0	28.20	21.90000	34.0	4.4	0.0	24886.0
11	42016.0	25.96	21.40000	35.4	0.0	0.0	25743.0

Utilizar mesma estratégia anterior para os demais atributos.

In [41]:
```
df_beer["temp_avg"] = df_beer["temp_avg"].replace(0,(df_beer["temp_max"] + df_beer["temp_min"])/2)
df_beer["temp_min"] = df_beer["temp_min"].replace(0,df_beer["temp_min"].mean())
df_beer["temp_max"] = df_beer["temp_max"].replace(0,df_beer["temp_max"].mean())
```

In [42]:
```
(df_beer == 0).any()
```

Out[42]:
```
data               False
temp_avg           False
temp_min           False
temp_max           False
precip              True
weekend             True
beer_consumption   False
dtype: bool
```

Seperando dados de treinamento e teste

In [43]:
```
from sklearn.model_selection import train_test_split

feature_col_names = ['temp_max', 'precip', 'weekend']
predicted_class_names = ['beer_consumption']

X = df_beer[feature_col_names].values
y = df_beer[predicted_class_names].values
split_test_size = 0.30

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=split_test_size, random_state=42)
```

In [44]:
```
print("{0:0.2f}% in training set".format((len(X_train)/len(df_beer.index)) * 100))
print("{0:0.2f}% in test set".format((len(X_test)/len(df_beer.index)) * 100))
```

```
69.86% in training set
30.14% in test set
```

In [45]:
```
from sklearn import linear_model

lr_model = linear_model.LinearRegression()
lr_model.fit(X_train, y_train.ravel())
```

Out[45]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

Testando valores par verificar a performance

In [46]:
```
from sklearn.metrics import mean_squared_error, r2_score

y_pred = lr_model.predict(X_test)

print('R2 score: %.2f' % r2_score(y_test, y_pred))
print('R2 score: %.2f' % lr_model.score(X_test, y_test))
```

```
R2 score: 0.74
R2 score: 0.74
```

R2 score está na faixa esperada de >= 70%

Simulação de um caso onde se espera maior consumo, clima quente, 35 graus C, sem chuva e no final de semana.

In [47]:
```
predict_value = [[35, 0, 1]]
lr_model.predict(predict_value)
```

Out[47]:
```
array([ 35092.25246658])
```

Simulação de um caso onde se espera um consumo menor que o de cima, clima quente, 35 graus C, sem chuva

In [48]:
```
1 predict_value = [[35, 0, 0]]
2 lr_model.predict(predict_value)
```

Out[48]:
```
array([ 29753.3647769])
```

Simulação de um caso onde se espera um consumo menor que o de cima, clima quente, 35 graus C, com chuva

In [49]:
```
1 predict_value = [[35, 20, 0]]
2 lr_model.predict(predict_value)
```

Out[49]:
```
array([ 28689.99450697])
```

Pior cenário, baixo consumo, frio (10 graus C), chovendo e em dia de semana.

In [50]:
```
1 predict_value = [[10, 20, 0]]
2 lr_model.predict(predict_value)
```

Out[50]:
```
array([ 12008.88877202])
```

Salvando modelo

```
from sklearn.externals import joblib
joblib.dump(lr_model, 'models/lr_model.pkl')
```

```
lr_model_loaded = joblib.load('models/lr_model.pkl')
```

```
predict_value = [[10, 20, 0]]
lr_model_loaded.predict(predict_value)
```

Por Claudio Pinheiro https://www.linkedin.com/in/claudio-pinheiro-cloud/