**Regressão linear com scikit-learn**

Utilizando uma das bibliotecas mais famosas em Machine Learning para criação de um modelo linear

Após os 2 primeiros artigos sobre os fundamentos de um modelo de regressão linear e sobre seus pressupostos, esse artigo irá tratar da etapa da criação do modelo em si, além da utilização da biblioteca *scikit-learn (sk-learn)*, também apresentarei um exemplo utilizando a biblioteca *statsmodels*, essa por sua vez, é mais voltada à estatística, conseguiremos por exemplo, obter os valores dos testes de significância global e individual, além de alguns outros testes que no mundo do ML não possuem tanta importância para o modelo linear.

Conforme deixado claro no [último artigo](https://medium.com/@joaogbsczip/pressupostos-da-regress%C3%A3o-linear-d07f784fe11a), para obter um modelo com alto poder de precisão em suas previsões, é necessário que o conjunto de dados coletado cumpra com algumas condições específicas, essas por sua vez, caso não cumpridas, implicam diretamente no desempenho do método dos Mínimos Quadrados Ordinários (MQO) para obter o menor valor de erro possível, consequentemente, os parâmetros encontrados não serão os mais precisos.

Em nosso caso, já percebemos que de fato não há relação linear entre as variáveis, a correlação é bastante baixa e também não seguiam uma distribuição normal (caso que realizamos uma transformação no *dataset* para tentar melhorar).

E agora, será que nosso modelo irá obter os melhores parâmetros possíveis? Em primeiro momento, criaremos um modelo utilizando a biblioteca *statsmodels*, essa biblioteca não é muito utilizada no mundo de ML pois é mais voltada à estatística, mas para a construção de modelos para ML, acaba sendo usada principalmente para obter valores de testes de hipóteses por exemplo, o que a *sk-learn* não nos provê inicialmente.

Após esse primeiro modelo, poderemos por exemplo, observar quais variáveis não possuem significância alguma para predizer o valor de y (variável dependente), apesar de que existem várias técnicas de *feature selection*específicas para isso, não é o intuito desta série de artigos, que visa apenas mostrar os fundamentos de um modelo de regressão linear.

Conforme já realizado no último artigo, para esse também será utilizado a linguagem *Python*, os testes serão realizados na plataforma *[Colab](https://colab.research.google.com/" \t "_blank)*da*Google.*

**Modelo com statsmodels**

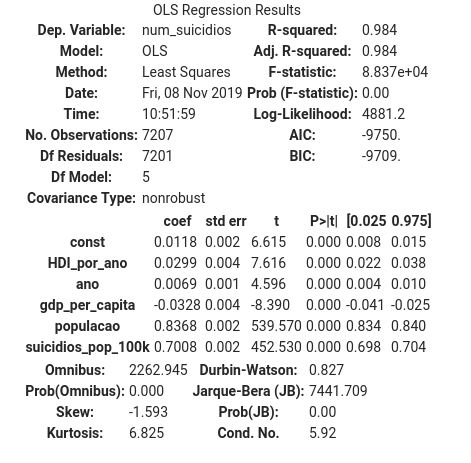
Conforme apresentado em sua documentação, *statsmodels*é uma biblioteca escrita em *Python* que provê diversas classes e métodos para a grande maioria de modelos e testes estatísticos.

statsmodels*is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration.*(<https://www.statsmodels.org/stable/index.html>)

Utilizaremos o seu método OLS, que advém de *Ordinary Least Squares*, passando as nossas variáveis X e y (variáveis independentes e dependente, respectivamente) treinaremos o modelo e, após o treinamento, será utilizado o método *summary*para verificar as estatísticas do mesmo*.*

Treinamento do modelo com OLS

E como resultado obtemos o seguinte:



Resultado do método summary

Ao olharmos para as estatísticas do modelo, podemos observar bons resultados, o primeiro é tanto o valor do R² como do R² ajustado, dado por *R-squared*e*Adj. R-squared* respectivamente, como discutido no [primeiro artigo](https://medium.com/@joaogbsczip/conhecendo-o-modelo-linear-e69aa8c3ab21), esse é o valor que dirá o quanto da variação de y é explicado pela variação total das variáveis independentes, o R² ajustado tem o mesmo significado porém é utilizado em casos de regressão linear múltipla, em nosso exemplo os valores foram os mesmos, porém podem variar. O valor de 0.98 nos diz que 98% do valor de y é explicado pelas variáveis de nosso modelo, o que é um ótimo resultado.

Outra estatística de bastante importância é o teste de significância global, dado pelo *F-statistic*, esse teste basicamente nos diz se ao menos uma de nossas variáveis explicativas podem ser utilizadas para explicar o valor de y, o seu *P-value*precisa ser menor do 0.05 para a hipótese ser verdadeira, o resultado obtido foi de 0.00.

A última medida que iremos levar em consideração é o teste de significância individual, dado pela estatística *t*de cada uma de nossas variáveis independentes, esse teste tem por objetivo dizer para cada variável escolhida se o seu valor pode ser utilizado para explicar a variabilidade de y, assim como a estatística F, seu *P-value* precisa ser menor do que 0.05 para a hipótese ser verdadeira, em nosso caso, obtivemos 0.00 para todas as variáveis.

Apesar dos resultados serem aparentemente bons, só poderemos comprovar a qualidade do modelo após verificarmos se o mesmo está cumprindo com o pressuposto da variação constante dos resíduos, visto que, caso haja a heterocedasticidade, a mesma implica diretamente nos erros-padrão da regressão, diminuindo o seu valor, logo, com essa influência sobre o erro-padrão, os testes *t*e*F*tornam-se insignificantes, por conta disso é preciso realizar mais algumas verificações como faremos no modelo com *sk-learn*.

Além dos valores dos testes de hipóteses, também é possível observar os valores dos coeficientes, a coluna *coef*representa esses valores, sendo a primeira linha o valor do intercepto e as demais, os valores de cada um dos nossos coeficientes de regressão.

Obtivemos bons resultados com a construção do modelo utilizando *statsmodels*, agora o mesmo modelo será construído com a biblioteca *sk-learn*.

**Modelo com sk-learn**

A biblioteca*sk-learn*tambémnos provê uma classe específica para trabalharmos com modelos lineares, em contrapartida à *statsmodels*, não poderemos visualizar algumas estatística como as apresentadas acima, entretanto, poderemos trabalhar com alguns plots para verificar a qualidade do ajuste do modelo e também realizarmos algumas predições para observar se de fato o modelo se ajustou aos dados.

Construindo um modelo com sk-learn

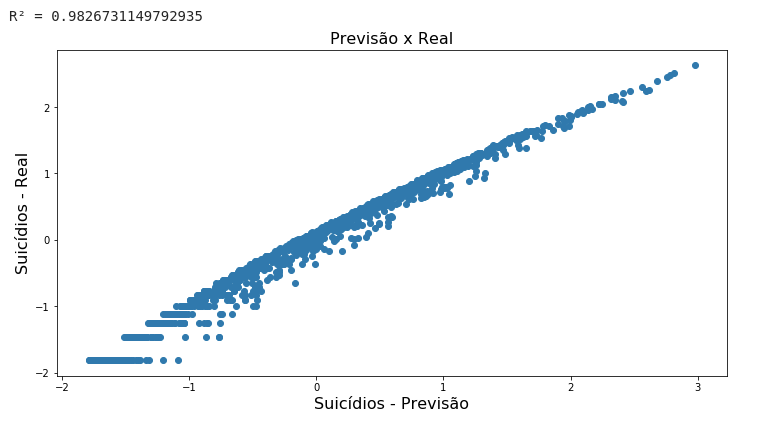
Agora que já possuímos o modelo treinando, realizaremos previsões para que possamos comprovar se o modelo se ajustou aos dados, tornando-o realmente significante para o problema apresentado. A partir da obtenção das previsões, obteremos os resíduos (erros) para verificar se o modelo está cumprindo com o pressuposto da homocedasticidade.

Quando o algoritmo treina o modelo, ele está descobrindo os parâmetros da regressão, o intercepto e os coeficientes para cada uma das variáveis independentes. Em posse desses parâmetros, torna-se possível obter as previsões, para isso, utiliza-se o método *predict,* passando os dados de teste que separamos anteriormente.

Após obter as previsões, iremos obter os resíduos, com a possa desses valores, podemos captar o valor do R² para avaliar o ajuste do modelo e também visualizar alguns gráficos que comprovarão esse ajuste.

Obtendo R² do modelo e plotando valor real x previsto

O modelo obteve o mesmo R² do modelo criado anteriormente utilizando o *statsmodels,*e seu gráfico do valor real x previsto real mostra uma relação bastante forte, um bom indício de que o modelo se ajustou aos dados

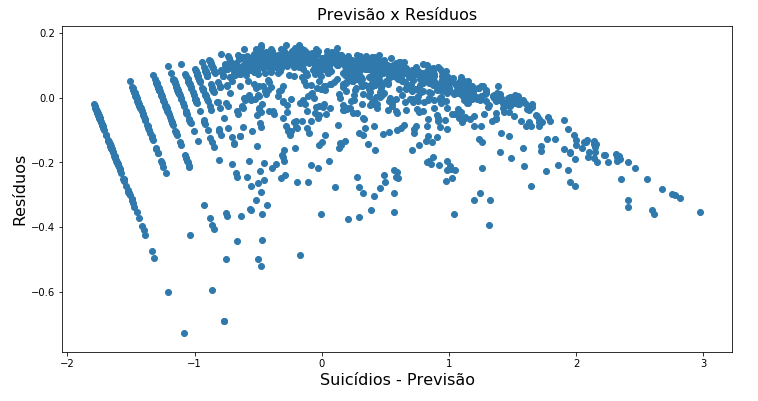


Plot do valor real x previsto

Com esses dois resultados já temos algum indício de que as previsões serão acuradas, visto que o modelo **aparentemente** se ajustou bem aos dados apresentados. Para uma última avaliação, iremos obter o gráfico dos resíduos, conhecido como *residual plot,*a partir deste, será possível observar se o erro é constante durante toda a variável independente, caso seja, o modelo estará cumprindo com o pressuposto da homocedasticidade, indicando que não houve forte influência no erro padrão dos estimadores do MQO.

Obtendo o residual plot

E como resultado:



Residual plot

O ideal para um *Residual Plot*é que a sua média seja em torno de 0 e de que não possua nenhum padrão visível entre os pontos do gráfico, observando o *plot*acima, pode-se observar que o mesmo conta com uma curvatura, em função disso, não podemos confirmar a hipótese da variação constante do resíduos, entretanto, por conta da sua forma, uma regressão quadrática (elevando X ao quadrado) poderia resolver o problema apresentado.

Embora nosso R² tenha obtido um valor significante, através do *residual plot* é possível observar que o modelo não cumpriu com o pressuposto da homocedasticidade, logo, não se pode comprovar a qualidade dos estimadores calculados pelo MQO, de forma que, para obter um resultado expressivo e previsões acuradas, seria necessário algum outro tipo de modelo capaz de expressar esse relacionamento, como a regressão quadrática. por exemplo.

A regressão linear é um modelo bastante eficaz em alguns casos, porém, por conta de suas condições impostas aos dados, acaba não sendo totalmente eficiente em outros.

Como falei no início do primeiro artigo, para trabalhar com ML é preciso pensar bem em qual modelo utilizar, o que pode ser definido durante a análise exploratória dos dados, onde já se tem um conhecimento de como o seu conjunto está distribuído e de quais seriam as suas possíveis implicações para cada um dos modelos que estão sendo levados em consideração para a utilização.

Por fim, por mais que o resultado não tenha sido significativo, pode-se concluir que toda experiência é válida, visto que para esse caso apresentado, realizar mais tratamentos ou utilizar algum outro modelo poderia ser a solução, porém, como o intuito era de apresentar os fundamentos e a construção de um modelo linear, quem sabe fique para um próximo artigo.