**Veja como usei Python para construir um modelo de regressão usando um conjunto de dados de comércio eletrônico**

Publicados: 2020-11-19

A linguagem de programação Python está ganhando popularidade entre os SEOs por sua facilidade de uso para automatizar tarefas diárias de rotina. Isso pode economizar tempo e gerar um aprendizado de máquina sofisticado para resolver problemas mais significativos que podem ajudar sua marca e sua carreira. Além das automações, este artigo ajudará aqueles que desejam aprender mais sobre ciência de dados e como o Python pode ajudar.

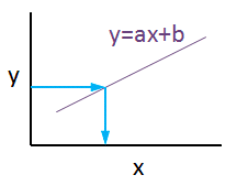
No exemplo abaixo, uso um conjunto de dados de comércio eletrônico para construir um modelo de regressão. Também explico como determinar se o modelo revela algo estatisticamente significativo, bem como como os valores discrepantes podem distorcer seus resultados.

Eu uso Python 3 e Jupyter Notebooks para gerar gráficos e equações com regressão linear em dados Kaggle. Verifiquei as correlações e construí um modelo básico de aprendizado de máquina com esse conjunto de dados. Com essa configuração, agora tenho uma equação para prever minha variável de destino.

Antes de construir meu modelo, quero dar um passo atrás para oferecer uma definição fácil de entender de regressão linear e por que ela é vital para a análise de dados.

**O que é regressão linear?**

A regressão linear é um algoritmo básico de aprendizado de máquina usado para prever uma variável com base em sua relação linear entre outras variáveis ​​independentes. Vamos ver um gráfico de regressão linear simples:



Se você conhece a equação aqui, também pode saber os valores de y em relação aos valores de x. '' a '' é o coeficiente de '' x '' e também a inclinação da reta, '' b '' é a interceptação, o que significa que quando x = 0, b = y.

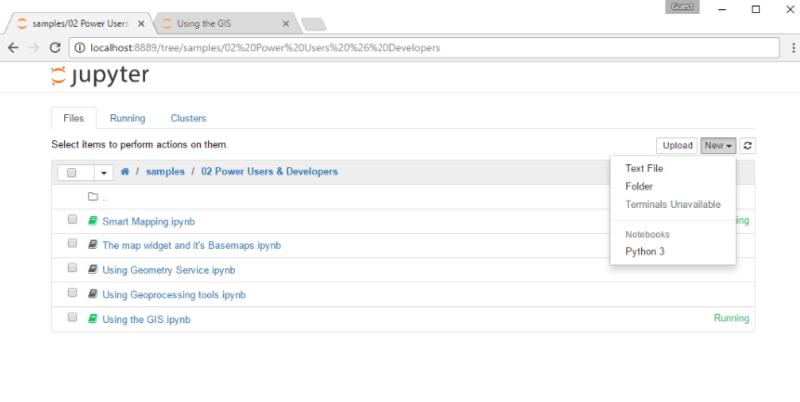
**Meu conjunto de dados de comércio eletrônico**

Eu usei este conjunto de dados do Kaggle. Não é muito complicado ou detalhado, mas o suficiente para estudar o conceito de regressão linear.

Se você é novo e nunca usou o Jupyter Notebook, aqui está uma dica rápida para você:

* Inicie o Terminal e escreva este comando: **notebook jupyter**

Uma vez inserido, este comando iniciará automaticamente seu navegador da Web padrão com um novo notebook. Clique em Novo e Python 3.



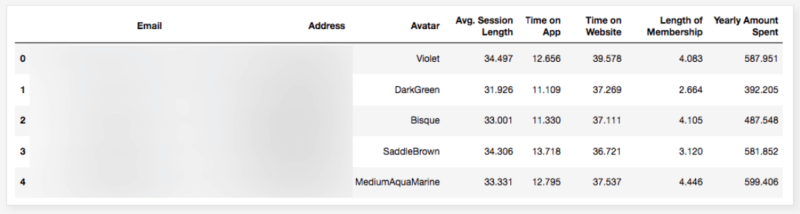
Agora é hora de usar alguns códigos Python sofisticados.

* **Importando bibliotecas**

import matplotlib.pyplot as plt  
import numpy as np  
import pandas as pd  
from sklearn.linear\_model import LinearRegression  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import mean\_absolute\_error  
import statsmodels.api as sm  
from statsmodels.tools.eval\_measures import mse, rmse  
import seaborn as sns  
pd.options.display.float\_format = '{:.5f}'.format  
import warnings  
import math  
import scipy.stats as stats  
import scipy  
from sklearn.preprocessing import scale  
warnings.filterwarnings('ignore')

* **Lendo dados**

df = pd.read\_csv("Ecom\_Customers.csv")  
df.head()

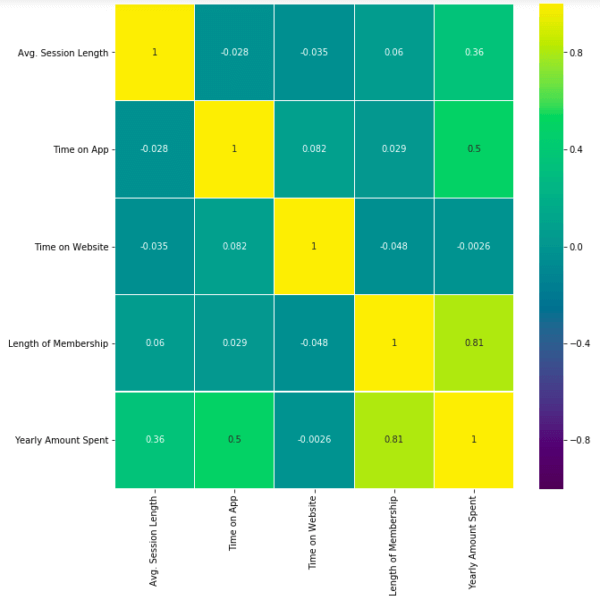


Minha variável de destino será *Quantidade anual gasta* e tentarei encontrar sua relação com outras variáveis. Seria ótimo se eu pudesse dizer que os usuários gastarão tanto, por exemplo, se o *tempo no aplicativo* aumentar 1 minuto a mais. Este é o objetivo principal do estudo.

* **Análise exploratória de dados**

Primeiro, vamos verificar o mapa de calor de correlação:

df\_kor = df.corr()  
plt.figure(figsize=(10,10))  
sns.heatmap(df\_kor, vmin=-1, vmax=1, cmap="viridis", annot=True, linewidth=0.1)



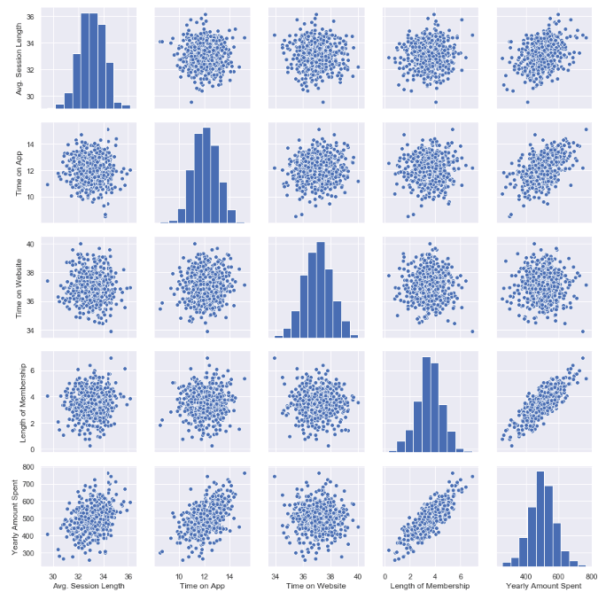
Este mapa de calor mostra as correlações entre cada variável, atribuindo-lhes um peso de -1 a +1.

Roxos significam correlação negativa, amarelos significam correlação positiva e chegar mais perto de 1 ou -1 significa que você tem algo significativo aí, analise. Por exemplo:

* *O número de membros* tem uma correlação positiva e alta com *o valor gasto anualmente.* (81%)
* *O tempo no aplicativo* também tem uma correlação, mas não poderosa, como a *duração da assinatura.* (50%)

Vamos ver essas relações em detalhes. Meu enredo favorito é **sns.pairplot** . Apenas uma linha de código e você verá todas as distribuições.

sns.pairplot(df)



Este gráfico mostra todas as distribuições entre cada variável, desenha todos os gráficos para você. Para entender quais dados eles incluem, verifique os nomes dos eixos esquerdo e inferior. (Se eles forem iguais, você verá um gráfico de barras de distribuição simples.)

Observe a **última linha, os** gráficos de Quantidade anual gasta (minha meta no eixo esquerdo) em comparação com outras variáveis.

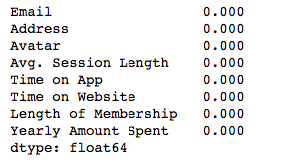
O Comprimento de Filiação tem uma linearidade realmente perfeita, é tão óbvio que se eu puder aumentar a fidelidade do cliente, eles gastarão mais! Mas quanto? Existe algum número ou coeficiente para especificá-lo? Podemos prever isso? Nós vamos descobrir.

* **Verificando valores ausentes**

Antes de construir qualquer modelo, você deve verificar se há alguma célula vazia em seu conjunto de dados. Não é possível manter esses valores NaN porque muitos algoritmos de aprendizado de máquina não oferecem suporte a dados com eles.

Este é o meu código para ver os valores ausentes:

df.isnull().sum()



isnull () detecta os valores NaN e sum () os conta.

Não tenho valores NaN, o que é bom. Se tivesse, deveria tê-los enchido ou deixado cair.

Por exemplo, para descartar todos os valores NaN, use o seguinte:

df.dropna(inplace=True)

Para preencher, você pode usar fillna ():

df["Time on App"].fillna(df["Time on App"].mean(), inplace=True)

Minha sugestão aqui é ler este ótimo artigo sobre como lidar com valores ausentes em seu conjunto de dados. Esse é outro problema para resolver e precisa de abordagens diferentes, se você as tiver.

**Construindo um modelo de regressão linear**

Até agora, explorei o conjunto de dados em detalhes e me familiarizei com ele. Agora é hora de criar o modelo e ver se posso prever a *quantidade anual gasta.*

Vamos definir X e Y. Primeiro, adicionarei todas as outras variáveis ​​a X e analisarei os resultados posteriormente.

Y=df["Yearly Amount Spent"]  
  
X=df[[ "Length of Membership", "Time on App", "Time on Website", 'Avg. Session Length']]

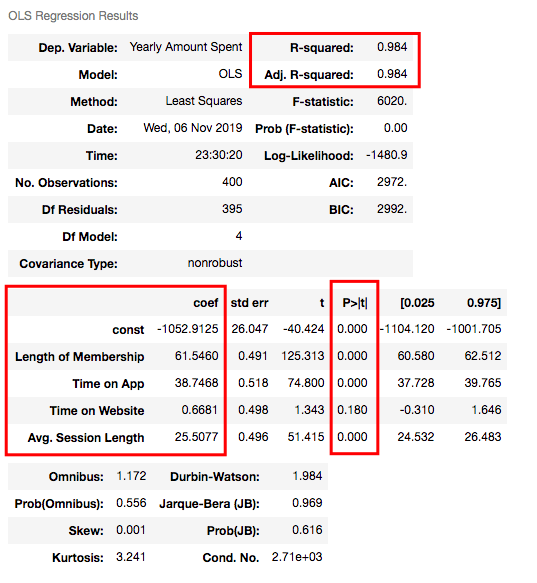
Em seguida, dividirei meu conjunto de dados em dados de treinamento e teste, o que significa que selecionarei 20% dos dados aleatoriamente e os separarei dos dados de treinamento. (test\_size mostra a porcentagem dos dados de teste - 20%) (Se você não especificar o **random\_state** em seu código, então cada vez que você executa (executa) seu código, um novo valor aleatório é gerado e os conjuntos de dados de treinamento e teste teriam valores diferentes a cada vez.)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size = 0.2, random\_state = 465)  
  
print('Training Data Count: {}'.format(X\_train.shape[0]))  
print('Testing Data Count: {}'.format(X\_test.shape[0]))

https://www.affde.com/uploads/article/1295/oRNH1PLmf5Xutbcm.png

Agora, vamos construir o modelo:

X\_train = sm.add\_constant(X\_train)  
results = sm.OLS(y\_train, X\_train).fit()  
results.summary()



**Compreendendo os resultados do modelo: isso é estatisticamente significativo?**

Então, o que todos esses números significam realmente?

Antes de continuar, será melhor explicar esses termos estatísticos básicos aqui, porque vou decidir se meu modelo é suficiente ou não olhando para esses números.

* **Qual é o valor p?**

O valor P ou valor de probabilidade mostra significância estatística. Digamos que você tenha a hipótese de que a CTR média das palavras-chave de sua marca é de 70% ou mais e seu valor p é 0,02. Isso significa que há **2% de probabilidade de** ver CTRs das palavras-chave de sua marca **abaixo de** 70%. É estatisticamente significativo? 0,05 é geralmente usado para o limite máximo (nível de confiança de 95%), portanto, se você tiver o valor p menor do que 0,05, sim! É significativo. Quanto menor for o valor p, melhores serão os resultados!

Agora vamos dar uma olhada na tabela de resumo. Minhas 4 variáveis ​​têm alguns valores-p que mostram suas relações, sejam elas significativas ou insignificantes, com *o valor anual gasto* . Como você pode ver, o *Tempo no site* é estatisticamente **insignificante** porque seu valor p é 0,180. Portanto, será melhor abandoná-lo.

* **O que é R ao quadrado e R ao quadrado ajustado?**

R quadrado é uma métrica simples, mas poderosa, que mostra quanta **variância é explicada** pelo modelo. Ele conta todas as variáveis ​​que você definiu em X e dá uma porcentagem de explicação. É algo como as capacidades do seu modelo.

**O** R ao quadrado **ajustado** também é semelhante ao R ao quadrado, mas conta apenas as variáveis ​​estatisticamente significativas. É por isso que é melhor olhar para R ao quadrado ajustado o tempo todo.

No meu modelo, 98,4% da variância pode ser explicada, o que é muito alto.

* **O que é Coef?**

Eles são coeficientes das variáveis ​​que nos fornecem a equação do modelo.

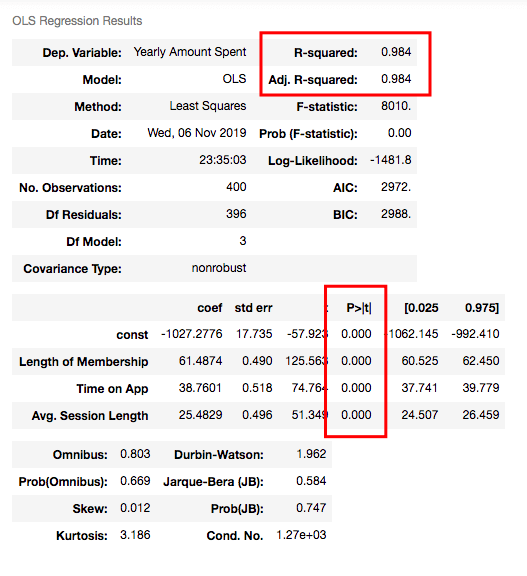
Então acabou? Não! Tenho *a* variável *Tempo no site* em meu modelo que é estatisticamente insignificante.

Agora vou construir outro modelo e soltar *a* variável *Tempo no site* :

X2=df[["Length of Membership", "Time on App", 'Avg. Session Length']]  
X2\_train, X2\_test, y2\_train, y2\_test = train\_test\_split(X2, Y, test\_size = 0.2, random\_state = 465)  
  
print('Training Data Count:', X2\_train.shape[0])  
print('Testing Data Count::', X2\_test.shape[0])

https://www.affde.com/uploads/article/1295/CeUOUgiCMFBr1QM7.png

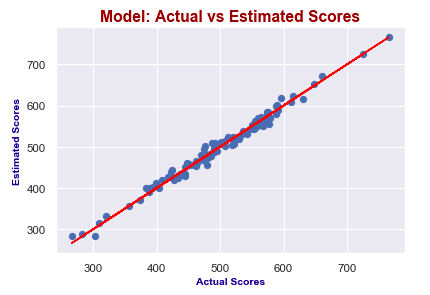
X2\_train = sm.add\_constant(X2\_train)  
  
results2 = sm.OLS(y2\_train, X2\_train).fit()  
results2.summary()



R ao quadrado ainda é bom e não tenho nenhuma variável com valor de p superior a 0,05.

Vejamos o gráfico do modelo aqui:

X2\_test = sm.add\_constant(X2\_test)  
  
y2\_preds = results2.predict(X2\_test)  
  
plt.figure(dpi = 75)  
plt.scatter(y2\_test, y2\_preds)  
plt.plot(y2\_test, y2\_test, color="red")  
plt.xlabel("Actual Scores", fontdict=ex\_font)  
plt.ylabel("Estimated Scores", fontdict=ex\_font)  
plt.title("Model: Actual vs Estimated Scores", fontdict=header\_font)  
plt.show()



Parece que eu prevejo valores muito bons! As pontuações reais e as pontuações previstas têm linearidade quase perfeita.

Por fim, verificarei os erros.

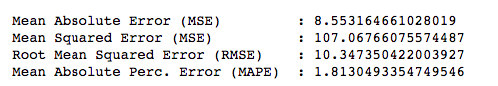
* **Erros**

Ao construir modelos, compará-los e decidir qual é o melhor é uma etapa crucial. Você deve testar muitas coisas e, em seguida, analisar os resumos. Elimine algumas variáveis, some ou multiplique-as e teste novamente. Após concluir a série de análises, você verificará os valores de p, erros e R ao quadrado. O melhor modelo terá:

* Valores P menores que 0,05
* Erros menores
* R quadrado mais alto ajustado

Vejamos os erros agora:

print("Mean Absolute Error (MAE) : {}".format(mean\_absolute\_error(y2\_test, y2\_preds)))  
print("Mean Squared Error (MSE) : {}".format(mse(y2\_test, y2\_preds)))  
print("Root Mean Squared Error (RMSE) : {}".format(rmse(y2\_test, y2\_preds)))  
print("Root Mean Squared Error (RMSE) : {}".format(rmse(y2\_test, y2\_preds)))  
print("Mean Absolute Perc. Error (MAPE) : {}".format(np.mean(np.abs((y2\_test - y2\_preds) / y2\_test)) \* 100))

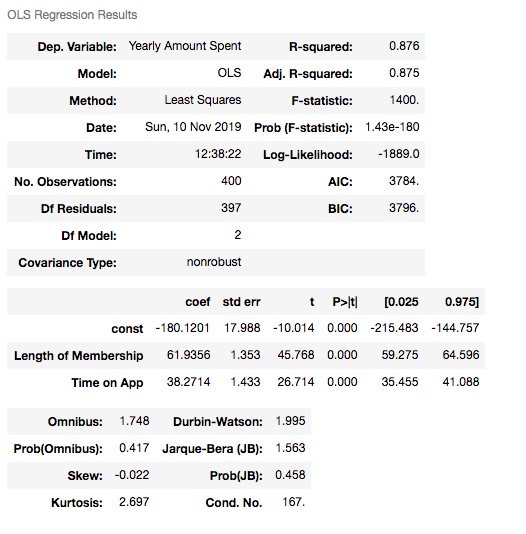


Se você quer saber o que é MSE, RMSE ou MAPE, você pode ler este artigo.

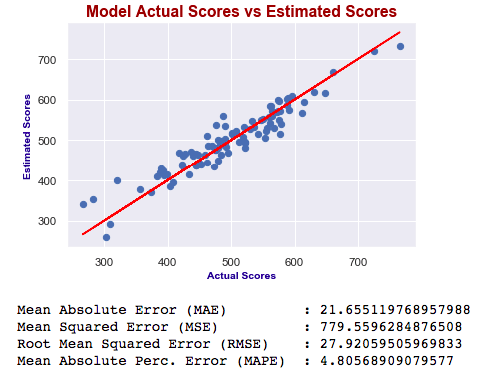
Eles são todos cálculos de erros diferentes e agora, vamos nos concentrar apenas nos menores enquanto comparamos modelos diferentes.

Portanto, para comparar meu modelo com outro, irei criar mais um modelo incluindo *Duração da Associação* e *Tempo no Aplicativo* apenas *.*

X3=df[['Length of Membership', 'Time on App']]  
Y = df['Yearly Amount Spent']  
X3\_train, X3\_test, y3\_train, y3\_test = train\_test\_split(X3, Y, test\_size = 0.2, random\_state = 465)  
  
X3\_train = sm.add\_constant(X3\_train)  
  
results3 = sm.OLS(y3\_train, X3\_train).fit()  
results3.summary()



X3\_test = sm.add\_constant(X3\_test)  
y3\_preds = results3.predict(X3\_test)  
  
plt.figure(dpi = 75)  
plt.scatter(y3\_test, y3\_preds)  
plt.plot(y3\_test, y3\_test, color="red")  
plt.xlabel("Actual Scores", fontdict=eksen\_font)  
plt.ylabel("Estimated Scores", fontdict=eksen\_font)  
plt.title("Model Actual Scores vs Estimated Scores", fontdict=baslik\_font)  
plt.show()  
  
print("Mean Absolute Error (MAE)  
: {}".format(mean\_absolute\_error(y3\_test, y3\_preds)))  
print("Mean Squared Error (MSE) : {}".format(mse(y3\_test, y3\_preds)))  
print("Root Mean Squared Error (RMSE) :  
{}".format(rmse(y3\_test, y3\_preds))) print("Mean Absolute Perc. Error (MAPE) :  
{}".format(np.mean(np.abs((y3\_test - y3\_preds) / y3\_test)) \* 100))



**Qual é o melhor?**

Como você pode ver, os erros do último modelo são maiores do que o primeiro. Também o R quadrado ajustado é diminuído. *Se os erros fossem menores, então diríamos que o último é melhor - independente de R ao quadrado.* Por fim, escolhemos erros menores e R ao quadrado mais alto. Acabei de adicionar este segundo para mostrar como você pode comparar os modelos e decidir qual é o melhor.

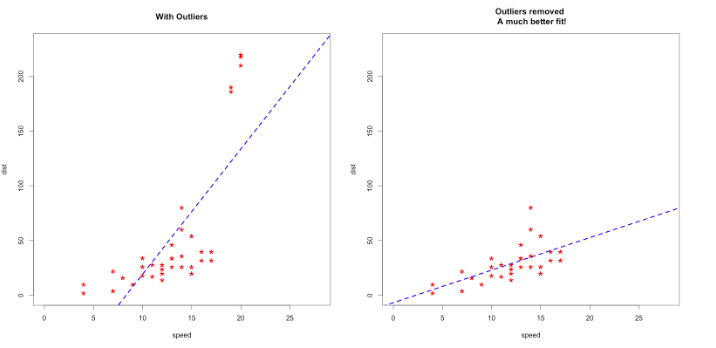
Agora, nosso modelo é este:

***Valor anual gasto = -1027,28 + 61,49x (duração da assinatura) + 38,76x (tempo no aplicativo) + 25,48x (duração média da sessão)***

Isso significa, por exemplo, se pudermos aumentar a duração da assinatura em mais 1 ano e manter todas as outras características fixas, uma pessoa gastará 61,49 dólares a mais!

**Dicas avançadas: outliers e não linearidade**

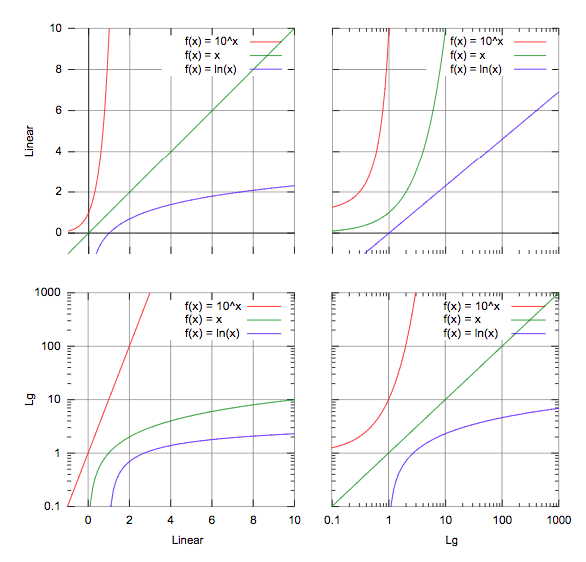
Quando você está lidando com dados reais, geralmente as coisas não são tão fáceis. Para encontrar linearidade ou modelos mais precisos, pode ser necessário fazer outra coisa. Por exemplo, se o seu modelo não for preciso o suficiente, verifique se há outliers. Às vezes, valores discrepantes podem enganar seus resultados!

*Fonte: http://r-statistics.co/Outlier-Treatment-With-R.html*

Além disso, às vezes você obterá linhas curvas em vez de lineares, mas verá que também existe uma relação entre as variáveis!

Então você deve pensar em transformar suas variáveis ​​usando logaritmos ou quadrados.

Aqui está um truque para você decidir qual usar:

*Fonte: https://courses.lumenlearning.com/boundless-algebra/chapter/graphs-of-exponential-and-logarithmic-functions/*

Por exemplo, no terceiro gráfico, se você tiver uma linha semelhante à verde, você deve considerar o uso de logaritmos para torná-la linear!

Há muitas coisas a serem feitas, então testar todas elas é muito importante.

**Conclusão**

Se você gosta de brincar com números e aprimorar seu conjunto de habilidades de ciência de dados, aprenda Python. Não é uma linguagem de programação muito difícil de aprender, e as estatísticas que você pode gerar com ela podem fazer uma grande diferença no seu trabalho diário.

Google Analytics, Google Ads, Search Console ... O uso dessas ferramentas já oferece toneladas de dados, e se você conhece os conceitos de manipulação de dados com precisão, obterá insights muito valiosos deles. Você pode criar previsões de tráfego mais precisas ou analisar dados do Analytics, como taxa de rejeição, tempo na página e suas relações com a taxa de conversão. No final do dia, pode ser possível prever o futuro da sua marca. Mas esses são apenas alguns exemplos.

Se você quiser ir mais longe na regressão linear, verifique meu modelo OLS do Google Page Speed ​​Insights. Eu construí meu próprio conjunto de dados e tentei prever o cálculo com base em métricas de velocidade, como FCP (Primeira pintura significativa), FMP (Primeira pintura significativa) e TTI (tempo de interação).

Para encerrar, combine seus dados, tente encontrar correlações e preveja seu alvo. Hamlet Batista tem um ótimo artigo sobre combinação prática de dados. Eu recomendo fortemente antes de construir qualquer modelo de regressão.