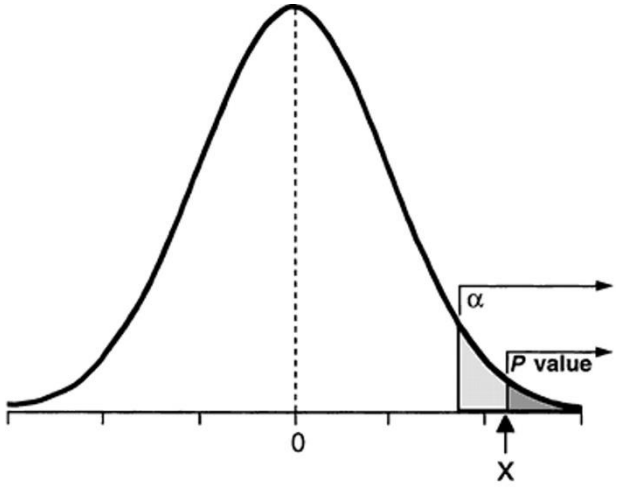

Facens

Teste de normalidade

No código abaixo simulam-se dados com distribuição log-normal e sua normalidade é testada. Espera-se que o teste falhe:

```
from scipy.stats import shapiro
x = np.random.randn(100)
stats, p = shapiro(x)

(0.98558509349823, 0.3496838808059692 4)
```



A hipótese nula é de que os dados são normalmente distribuídos

Se o *valor-p* é menor do que o nível de significância (α), então a hipótese nula é rejeitada

Logo, os dados NÃO SÃO REJEITADOS → são normalmente distribuídos

Facens

Comparação da Média de um Conjunto de Dados com uma Constante

```
from scipy.stats import norm
from scipy import stats

x1 = norm.rvs(loc=10000, scale=1000, size=50)
x2 = norm.rvs(loc=1000, scale=100, size=50)
x3 = norm.rvs(loc=1000, scale=100, size=50)

print(x1.mean())
print(x2.mean())
print(x3.mean())

stats.ttest_1samp(x1,0)
Ttest_1sampResult(statistic=70.68906854740204,
pvalue=5.551055102945442e-51)
```

Facens

Comparação da Média de Dois Conjuntos de Dados

Comparando x1 com x2

```
stats.ttest_ind(x1,x2)
```

```
Ttest_indResult(statistic=72.022883333939518,
pvalue=1.1191380960232934e-86)
```

Comparando x2 com x3

```
stats.ttest_ind(x2,x3)
```

```
Ttest_indResult(statistic=0.10080661932376216,
pvalue=0.91990992237400093)
```

Facens

REGRESSÃO LINEAR

Facens

Regressão

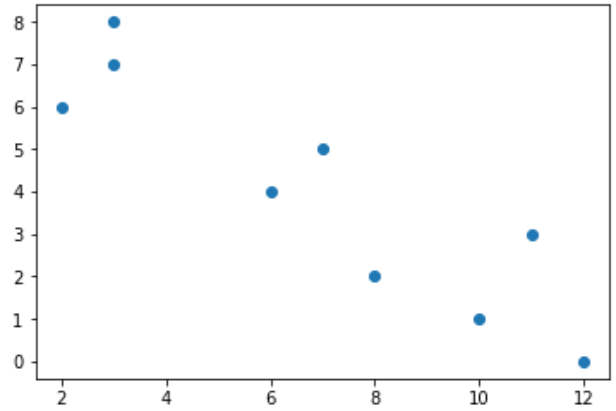
Regressão é o método estatístico usado para estimar as relações entre as variáveis.

A maneira mais fácil de analisar se a regressão é um método aplicável a seus dados, é plotando um gráfico de dispersão.

Facens

Gráfico de Dispersão - Tanino

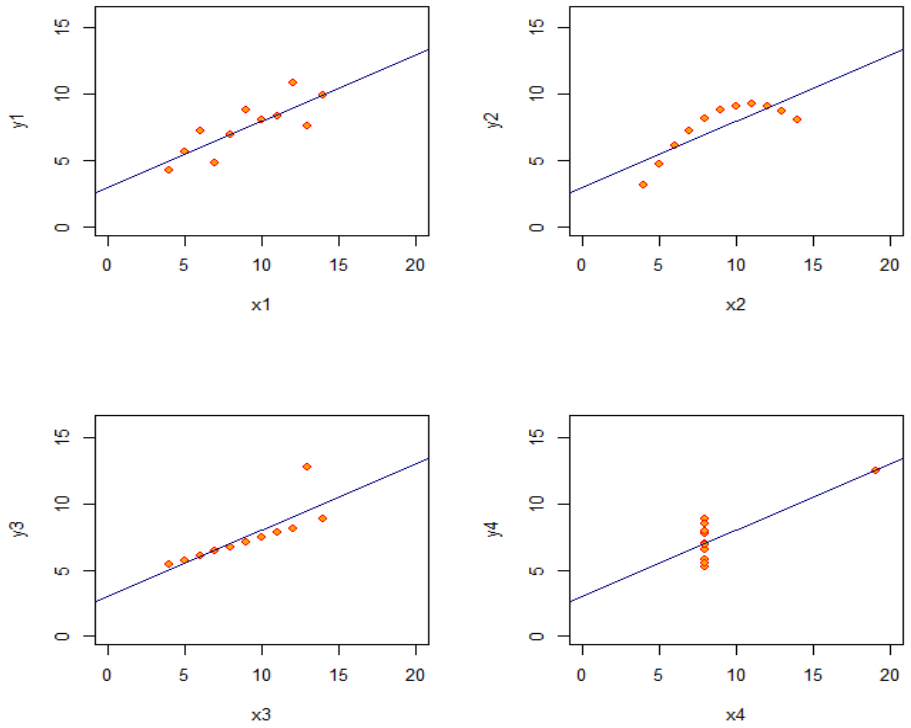
```
import matplotlib.pyplot as plt
os.chdir("D:\Dropbox\Fund Prog e Estatistica\db")
arquivo = pd.read_csv("tannin.csv")
print(arquivo.describe())

arquivo_np = np.array(arquivo)
print(arquivo_np[:,1])
plt.plot(arquivo_np[:,0],arquivo_np[:,1],'o')
plt.title('Tannin vs Growth (Training set)')
plt.xlabel('Growth')
plt.ylabel('Tannin')
plt.show()
```



Por que realizar uma inspeção visual?

Os quatro conjuntos de dados abaixo possuem a mesma média, variância, linha de regressão e coeficiente de correlação.

A regressão linear é aplicável somente no primeiro caso, ou, no máximo, no terceiro, se removermos o *outlier*.



Facens

Regressão Linear

O modelo de regressão mais simples é o linear:

$$y = \beta_0 + \beta_1 x$$

Facens

Regressão Linear - statsmodel

```
import numpy as np
import pandas as pd
import statsmodels.api as sm

X = pd.DataFrame(arquivo["tannin"])
y = pd.DataFrame(arquivo["growth"])

X = sm.add_constant(X)
model = sm.OLS(y, X).fit()
predictions = model.predict(X) # make the
predictions by the model

model.summary()
```

Facens

Dep. Variable:	growth	**R-squared:**	0.816
Model:	OLS	**Adj. R-squared:**	0.789
Method:	Least Squares	**F-statistic:**	30.97
Date:	Sat, 13 Apr 2019	**Prob (F-statistic):**	0.000846
Time:	02:38:32	**Log-Likelihood:**	-16.380
No. Observations:	9	**AIC:**	36.76
Df Residuals:	7	**BIC:**	37.15
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[95.0% Conf. Int.]**
const	11.7556	1.041	11.295	0.000	9.294 14.217
tannin	-1.2167	0.219	-5.565	0.001	-1.734 -0.700

Omnibus:	0.466	**Durbin-Watson:**	2.937
Prob(Omnibus):	0.792	**Jarque-Bera (JB):**	0.227
Skew:	0.319	**Prob(JB):**	0.893
Kurtosis:	2.554	**Cond. No.**	9.06

Exercício

Usando a base HBAT.csv, faça uma regressão linear simples (1 variável independente X) prevendo uma variável dependente Y.

Considere como X as colunas x9 até x18 (serão 10 modelos ao total).
Considere x19 como a variável dependente Y.

Baseado na informação abaixo, sobre cada uma das variáveis analisadas, tire suas conclusões

Facens

Database

Independent Variables	
X_6 Product Quality	X_{13} Competitive Price
X_7 E-Commerce Activities	X_{14} Warranty and Claims
X_8 Technical Support	X_{15} New Products
X_9 Complaint Resolution	X_{16} Ordering and Billing
X_{10} Advertising	X_{17} Price Flexibility
X_{11} Product Line	X_{18} Delivery Speed
X_{12} Salesforce Image	

Dependent Variable
X_{19} Satisfaction

Facens

Exercício

```
import pandas as pd
import statsmodels.api as sm
import os

os.chdir("D:\Dropbox\Fund Prog e
Estatistica\db")
dados = pd.read_csv("HBAT.csv")

# Variável independente
X = ["x" + str(i) for i in range(9,19)]

# Variável dependente
y = pd.DataFrame(dados["x19"])
```

Facens

Exercício

```
for independente in X:
    x = pd.DataFrame(dados[independente])

    x = sm.add_constant(x)

    # OLS - Ordinary Least Squares
    model = sm.OLS(y, x).fit()

    # Exibindo as previsões para o conjunto passado
    # model.predict(x)

    # Exibindo as estatísticas do modelo
    print(model.summary())
```

Facens

Resultados

OLS Regression Results

Dep. Variable:	x19	R-squared:	0.333
Model:	OLS	Adj. R-squared:	0.326
Method:	Least Squares	F-statistic:	48.92
Date:	Fri, 26 Apr 2019	Prob (F-statistic):	3.30e-10
Time:	13:49:22	Log-Likelihood:	-138.69
No. Observations:	100	AIC:	281.4
Df Residuals:	98	BIC:	286.6
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[95.0% Conf.	Int.]
const	3.2791	0.529	6.194	0.000	2.229	4.330
x18	0.9364	0.134	6.994	0.000	0.671	1.202

Omnibus:	0.361	Durbin-Watson:	1.922
Prob(Omnibus):	0.835	Jarque-Bera (JB):	0.448
Skew:	-0.136	Prob(JB):	0.799
Kurtosis:	2.818	Cond. No.	22.7

Coeficiente de determinação R2

Cálculo de R2 pode ser feita através da soma do erro dos quadrados (SSE) e da soma dos quadrados da variável de resposta (SSR)

$$SST = SSE + SSR$$

$$\sum (y - \bar{y})^2 = \sum (y - \hat{y})^2 + \sum (\hat{y} - \bar{y})^2$$

Facens

Exercício

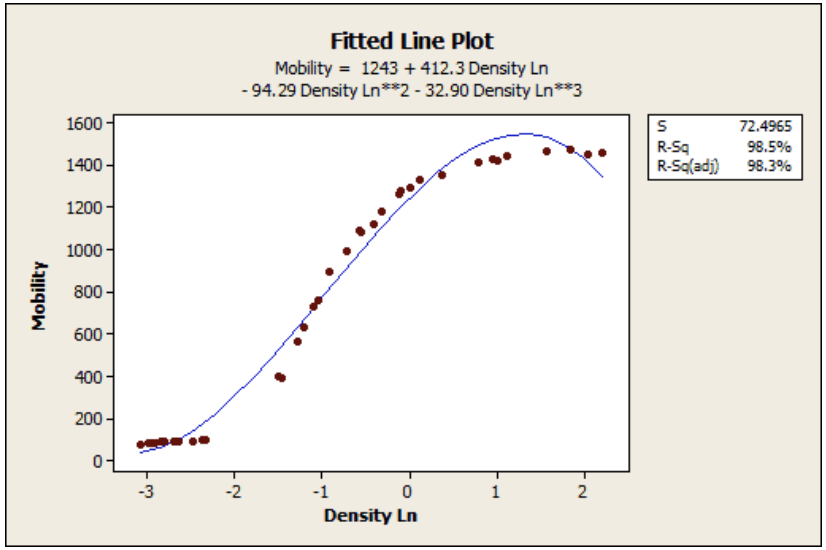
Sabendo as fórmulas para a soma dos quadrados do erro, da regressão e do total, escreva funções que retornem seus valores e, a partir delas, calcule o coeficiente de determinação

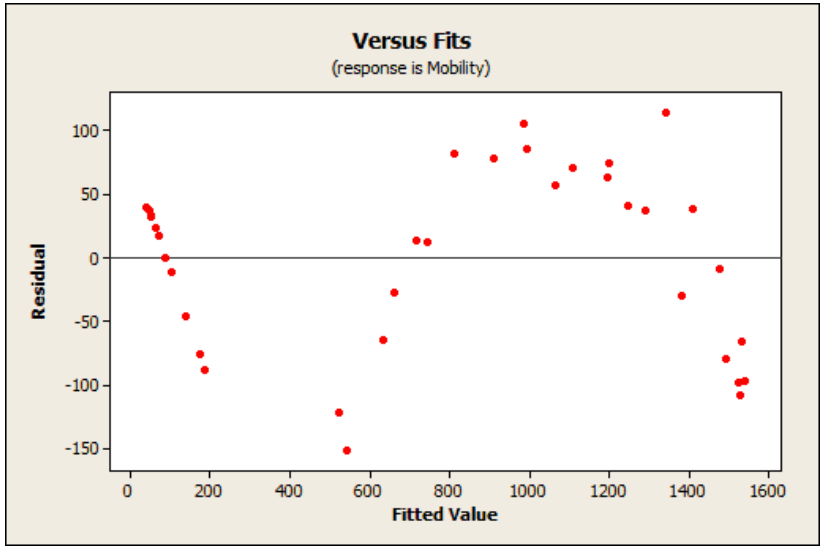
```
real = np.array(y)
yhat = np.array(pd.DataFrame(model.fittedvalues))

SSE = sum((real-yhat)**2)
SSR = sum((yhat-real.mean())**2)
SST = sum((real-real.mean())**2)
R2 = SSR/SST
1 - SSE/SST
```

Facens

R^2 alto é bom?

Segue um exemplo de R^2 = 98,5%, onde o gráfico resíduos vs previstos apresenta um padrão (isto é indesejável)





Em algumas áreas do conhecimento, como previsão de comportamento humano, é esperado um R^2 abaixo de 0,5. Isto porque seres humanos são difíceis de serem previstos.

Facens

R^2 ajustado

R^2 ajustado é preferível ao R^2, pois considera os graus de liberdade de cada soma dos quadrados

$$R^2_{adj} = 1 - \frac{SSE/df_\varepsilon}{SST/df_T}$$

R^2 penaliza o pesquisador quando este insere mais uma variável independente na regressão

Facens


Correlação NÃO É Causalidade
exoteric.roach.org

Premissas da Regressão Linear

- Linearidade
- Y é normalmente distribuído
- $E(\varepsilon|X) = 0$
- Residuos normal i.i.d.
- $Var(\varepsilon|X) = \sigma^2 I$ (Variância homogênea = Homocedasticidade)

Facens

Análise do Erro

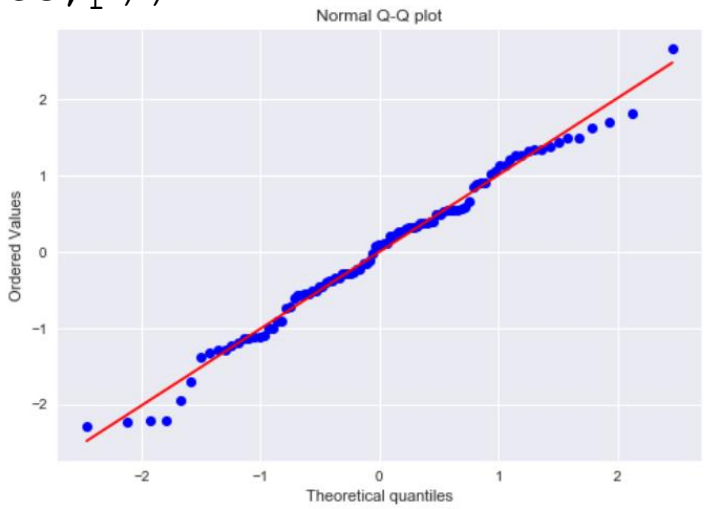
Homocedasticidade

```
import matplotlib.pyplot as plt
import scipy.stats as stats

plt.plot(model.fittedvalues, model.resid, 'o')
plt.title('Residuals vs Fitted Values')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.show()
```

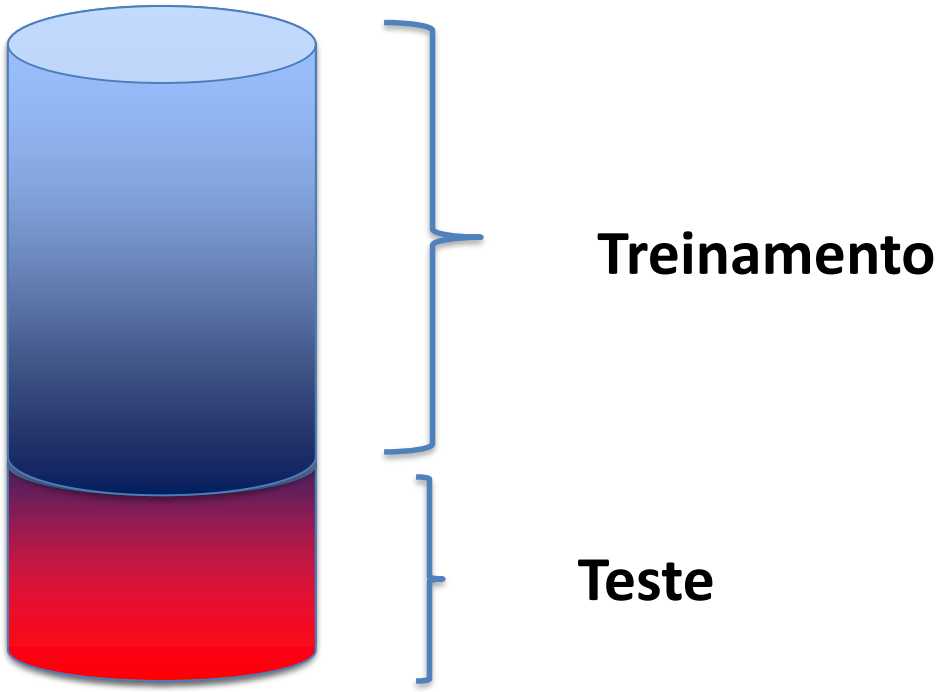
Análise do Erro

Normalidade do Erro – $N(0, \sigma^2)$

```
z = (model.resid - model.resid.mean()) / model.resid.std()
stats.probplot(z, dist="norm", plot=plt)
plt.title("Normal Q-Q plot")
plt.show()

#Teste de Normalidade
stats, p = stats.shapiro(model.resid)
print("W: %.4f   p-value: %.4f" % (stats,p))

#Gráfico de Densidade
import seaborn as sns
sns.distplot( model.resid)
```

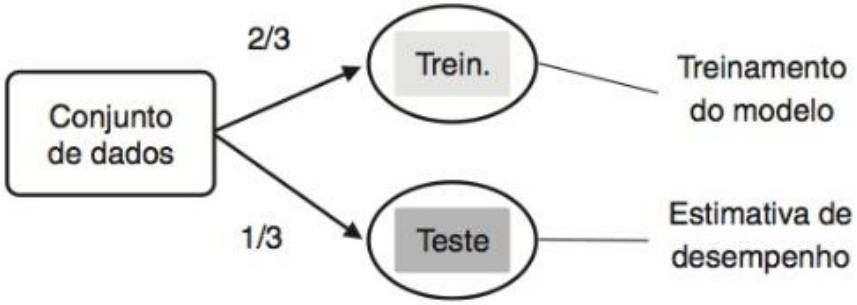


Regressão Linear – Treinamento e Teste

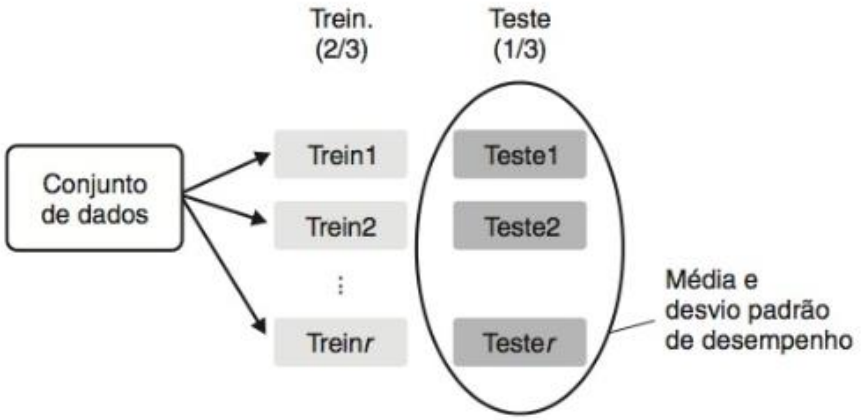


Facens

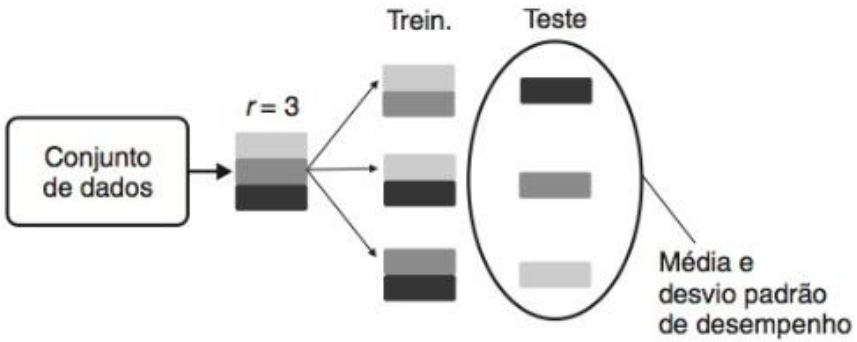
Métodos de Treinamento



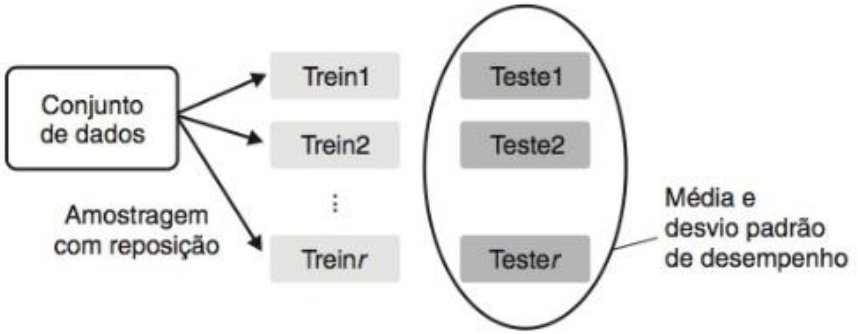
(a) *Holdout*



(b) Amostragem aleatória



(c) Validação cruzada



(d) *Bootstrap*

Métodos de treinamento

Holdout:

- Divisão em dois grupos de amostra: p e (1-p), p=2/3
- não permite avaliar o quanto o desempenho de uma técnica varia quando diferentes combinações de objetos são apresentadas em seu treinamento.

Amostragem aleatória

- diversas partições aleatórias e obter uma média de desempenho em holdout, um método às vezes referenciado como random subsampling

Facens

Métodos de treinamento

Validação cruzada r-fold (Ex: r=3)

- No método de validação cruzada r-fold, o conjunto de exemplos é dividido em r subconjuntos de tamanho aproximadamente igual. Os objetos de r − 1 partições são utilizados no treinamento de um preditor, o qual é então testado na partição restante. Esse processo é repetido r vezes, utilizando em cada ciclo uma partição diferente para teste. O desempenho final do preditor é dado pela média dos desempenhos observados sobre cada subconjunto de teste. Figura 9.2(c) emprega r = 3.

Leave-one-out

- No caso extremo em que r = n, em que n representa o número de casos disponíveis, tem-se o método leave-one-out. No leave-one-out, a cada ciclo exatamente um exemplo é separado para teste, enquanto os n−1 exemplos restantes são utilizados no treinamento do preditor. O desempenho é dado pela soma dos desempenhos verificados para cada exemplo de teste individual. Esse método produz uma estimativa mais fiel do desempenho preditivo do modelo. Porém, ele é computacionalmente caro, e geralmente aplicado somente em amostras de dados pequenas.

Facens

Métodos de treinamento

Bootstrap

- No método booststrap, r subconjuntos de treinamento são gerados a partir do conjunto de exemplos original. Os exemplos são amostrados aleatoriamente desse conjunto, com reposição.

Facens

Regressão Linear – Treinamento e Teste

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import seaborn as seabornInstance

X = ["x" + str(i) for i in range(9,19)]
X = dados[X]
y = dados["x19"]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)

linearRegressor = LinearRegression()
linearRegressor.fit(X_train, y_train)
```

Facens

Obtenção dos Resultados

Intercepto ou Coeficiente Linear

```
print(linearRegressor.intercept_)
```

Coeficiente Angular (slope)

```
print(linearRegressor.coef_)
```

Previsão

```
y_pred = linearRegressor.predict(X_test)
print(y_pred)

df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

Facens

Métricas de Avaliação da Previsão

Erro Médio Absoluto (MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y - \hat{y}|$$

Erro Quadrático Médio (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2$$

Raiz Quadrada do Erro Quadrático Médio (RMSE)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2}$$

Facens

Previsão

```
from sklearn import metrics
```

MAE

```
print('Mean Absolute Error:',
metrics.mean_absolute_error(y_test, y_pred))
```

MSE

```
print('Mean Squared Error:',
metrics.mean_squared_error(y_test, y_pred))
```

RMSE

```
print('Root Mean Squared Error:',
np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

Facens

Regressão Múltipla

```
os.chdir("D:\Henrique\Documents\POS\Aula")
data = pd.read_csv("train.csv")

X = data[["Pclass", "Fare"]]
y = data["Survived"]
X = sm.add_constant(X)

model = sm.OLS(y, X).fit()
predictions = model.predict(X)
model.summary()
```

Facens

Dep. Variable:	Survived	**R-squared:**	0.122
Model:	OLS	**Adj. R-squared:**	0.120
Method:	Least Squares	**F-statistic:**	61.61
Date:	Sat, 13 Apr 2019	**Prob (F-statistic):**	8.78e-26
Time:	03:02:29	**Log-Likelihood:**	-564.07
No. Observations:	891	**AIC:**	1134.
Df Residuals:	888	**BIC:**	1149.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[95.0% Conf. Int.]**
const	0.7310	0.060	12.197	0.000	0.613 0.849
Pclass	-0.1643	0.022	-7.501	0.000	-0.207 -0.121
Fare	0.0010	0.000	2.714	0.007	0.000 0.002

Omnibus:	1030.575	**Durbin-Watson:**	1.966
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	88.979
Skew:	0.456	**Prob(JB):**	4.77e-20
Kurtosis:	1.750	**Cond. No.**	246.

Matriz de Correlação

Permite a escolha de qual variável entra primeiro na fórmula da regressão múltipla

```
data = pd.DataFrame(dados, columns=X)
data['y'] = y
np_data = np.array(data)
print(np.corrcoef(data[:10])
```

Facens

Regressão Múltipla

Modelo de regressão múltipla, usando fórmula

```
model = smf.ols('y ~ x9 + x10 + x11', data=data).fit()
```

Exibindo resultados

```
model.summary()
```

Exibindo resultados

```
model.params
model.tvalues
```

Facens

Trabalho

Encontrar uma base de dados pública e rodar uma regressão logística e uma regressão linear, usando conceitos de treinamento e teste.

Discutir variáveis e apresentar a qualidade dos modelos treinados.

Resultados deverão ser apresentados em ppt, na aula do dia **07 de dezembro de 2019**

Submissão do ppt: via Canvas, até o início da aula (7:59h de 07/12)

Tempo de apresentação: 10 minutos

Grupo: 2 integrantes

Facens

Exercício

Escreva um código para rodar amostragem aleatória com 1.000 repetições, para prever X19 em função de X9

Facens


Facens