Apresentação Datasets Depressão + Titanic

Gabriel Marchezi, Manoel Rodrigues, Matheus Oliveira

Titanic

Data Dictionary

Variable	Definition	Key
survival	Survival	0 = No, 1 = Yes
pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Sex	
Age	Age in years	
sibsp	# of siblings / spouses aboard the Titanic	
parch	# of parents / children aboard the Titanic	
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

Depressão

Variável	Definição
CC	Perguntas sobre Doenças Crônicas
D	Perguntas sobre Depressão
SC	Perguntas gerais sobre Transtornos Comuns
DSM_MDDH	Indica se a pessoa tem depressão ou não baseado nas perguntas

O processo de Machine Learning - Titanic

Survived	Pclass	Age	SibSp	Parch	Fare	Sex
0	3	24.0	0	0	0.017274	0
0	3	20.0	0	0	0.015330	1
0	2	59.0	0	0	0.026350	1
0	2	24.0	0	0	0.025374	0
0	3	40.0	0	0	0.014102	1

O processo de Machine Learning - Depressão

IA	D64B	D64C	D64D	D64E	D64G	D64H	D64I	D64J	D64K	D64L	D64M	D64N	D66A	D66B	D68	SC20	SC20A	SC21	SC22	SC23	SC25	SC25A	SC26	SC26A	SC26B	dsm_mddh
N	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5	5.0	1	1	5	5	NaN	5	5.0	5.0	5
N	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5	5.0	1	1	5	1	5.0	1	NaN	NaN	5
N	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5	5.0	5	1	1	1	5.0	5	5.0	5.0	5
N	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5	5.0	1	1	1	5	NaN	1	NaN	NaN	5
N	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5	5.0	5	5	5	5	NaN	5	5.0	5.0	5

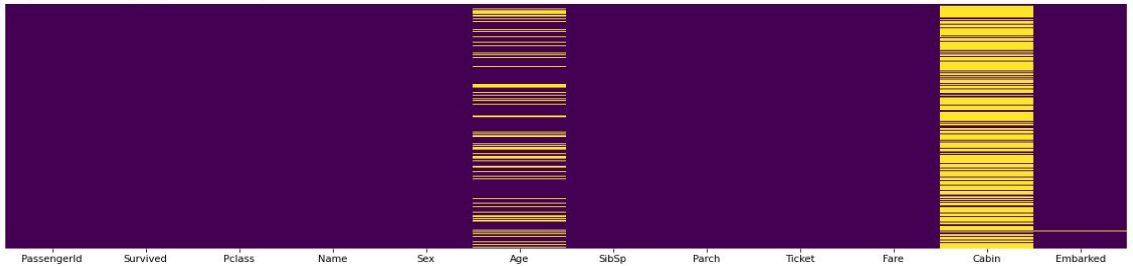
Pré Processamento/Tratamento de Dados - Titanic

```
[ ] df_titanic.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
[ ] fig, ax = plt.subplots(figsize=(20,5))
    sns.heatmap(df_titanic.isnull(),
                yticklabels=False,
                cbar=False,
                cmap='viridis',
               ax=ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f00b05be8d0>
```




```
df_titanic.drop(['Cabin', 'PassengerId', 'Name', 'Ticket', 'Embarked'], axis='columns', inplace=True)
df_titanic.dropna(axis=0, subset=['Age'], inplace=True)
df_titanic.head()
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
0	0	3	male	22.0	1	0	7.2500
1	1	1	female	38.0	1	0	71.2833
2	1	3	female	26.0	0	0	7.9250
3	1	1	female	35.0	1	0	53.1000
4	0	3	male	35.0	0	0	8.0500

```
[ ] #df_titanic
    le = LabelEncoder()
    label = le.fit_transform(df_titanic['Sex'])
    df_titanic.drop('Sex',axis='columns',inplace=True)
    df_titanic['Sex'] = label
```

```
[ ] df_titanic.head()
    # 1 --> Male
    # 0 --> Female
```

	Survived	Pclass	Age	SibSp	Parch	Fare	Sex
0	0	3	22.0	1	0	7.2500	1
1	1	1	38.0	1	0	71.2833	0
2	1	3	26.0	0	0	7.9250	0
3	1	1	35.0	1	0	53.1000	0
4	0	3	35.0	0	0	8.0500	1

Pré Processamento/Tratamento de Dados - Titanic

```
[29] count_0 = len(df_titanic[df_titanic.Survived == 0])# Não sobreviventes
     count_0
```

```
424
```

```
[30] count_1 = len(df_titanic[df_titanic.Survived == 1])# Sobreviventes
     count_1
```

```
290
```

```
[31] df_titanic.shape
```

```
(714, 7)
```

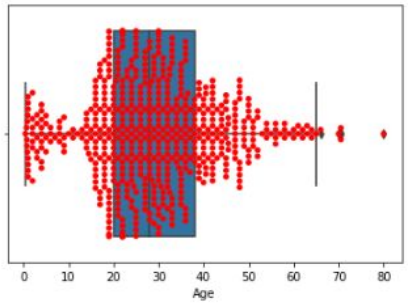
```
[32] df_balanceado = df_titanic[df_titanic.Survived == 0].sample(count_1, replace=True)
     df_balanceado = pd.concat([df_balanceado,df_titanic[df_titanic.Survived == 1]], axis=0)
     df_balanceado.shape
```

```
(580, 7)
```

```
[35] Melhores_Atributos = SelectKBest(chi2, k='all').fit(X_train.fillna(0), y_train)
     Melhores_Atributos.get_support()
     Lista_Melhores = list(X_train.columns[Melhores_Atributos.get_support()])
     Score_Melhores = pd.Series(Melhores_Atributos.scores_)
     Score_Melhores.index = X_train.columns
     Score_Melhores = Score_Melhores.sort_values(ascending=False)
     for i in Score_Melhores.index:
         print('%s --->  %f'%(i,Score_Melhores[i]))
```

```
Sex --->  53.400873
Age --->  32.643678
Pclass --->  18.436577
Fare --->  4.427265
SibSp --->  2.641333
Parch --->  1.604163
```

```
[39] ax = sns.boxplot(x=df_balanceado['Age'])
     ax = sns.swarmplot(x=df_balanceado['Age'], color='red')
```



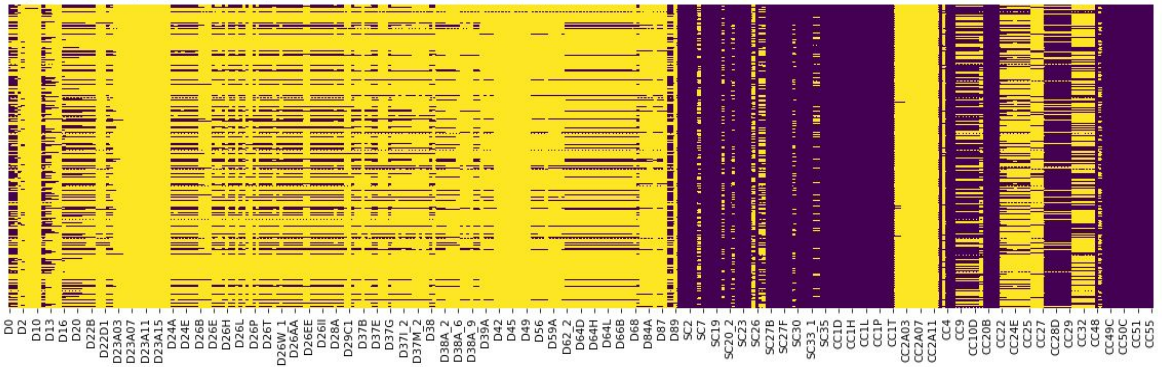
Pré Processamento/Tratamento de Dados - Depressão

```
excluir_nulos = []
nulos = df_depressao.isnull().sum()
nulos.sort_values(ascending=False,inplace=True)
for i in nulos.index:
    print(i,"--->",nulos[i])
    if nulos[i] >= 5037*0.9:
        excluir_nulos.append(i)
```

```
CC2A06 ---> 5037
D23A18 ---> 5037
D23A16 ---> 5037
D23A15 ---> 5037
D23A14 ---> 5037
D23A13 ---> 5037
D23A12 ---> 5037
D23A11 ---> 5037
D23A10 ---> 5037
CC2A04 ---> 5037
CC2A05 ---> 5037
D23A09 ---> 5037
CC2A12 ---> 5037
CC2A11 ---> 5037
CC2A10 ---> 5037
CC2A09 ---> 5037
CC2A08 ---> 5037
D23A17 ---> 5037
CC2A07 ---> 5037
D23A08 ---> 5036
D23A07 ---> 5036
D50 ---> 5035
CC2A03 ---> 5035
```

```
[ ] fig, ax = plt.subplots(figsize=(20,5))
    sns.heatmap(df_depressao.isnull(),
                yticklabels=False,
                cbar=False,
                cmap='viridis',
                ax=ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1bae433c90>
```



Pré Processamento/Tratamento de Dados - Depressão

```
[12] X_train = df_depressao.drop(['dsm_mddh'], axis=1)
     y_train = df_depressao['dsm_mddh']

     X_train.fillna(0, inplace=True)

     Melhores_Atributos = SelectKBest(chi2, k=50).fit(X_train, y_train)
     Melhores_Atributos.get_support()
     Lista_Melhores = list(X_train.columns[Melhores_Atributos.get_support()])
     Score_Melhores = pd.Series(Melhores_Atributos.scores_)
     Score_Melhores.index = X_train.columns
     Score_Melhores = Score_Melhores.sort_values(ascending=False)
     j = 0
     for i in Score_Melhores.index:
         j += 1
         print(j, '%s --->  %f'%(i,Score_Melhores[i]))
```

```
[13] Lista_Melhores.append('dsm_mddh')
     df_depressao_50 = df_depressao[Lista_Melhores]
     df_depressao_50
```

```
[ ] count_1 = len(df_depressao_50[df_depressao_50.dsm_mddh == 1]) # Diagnostico Negativo
    count_1
```

```
[ ] count_5 = len(df_depressao_50[df_depressao_50.dsm_mddh == 5]) # Diagnostico Positivo
    count_5
```

```
[ ] df_balanceado = df_depressao_50[df_depressao_50.dsm_mddh == 5].sample(count_1, replace=True)
    df_balanceado = pd.concat([df_balanceado,df_depressao_50[df_depressao_50.dsm_mddh == 1]], axis=0)
    df_balanceado.shape
```

Análise Exploratória (Promental)

D12A Quantos dias durou o período mais longo em que o(a) Sr(a). se sentiu (FRASE-CHAVE) durante a maior parte do dia?

Value	Count	Frequency (%)
2	422	8.4%
1	341	6.8%
3	261	5.2%
7	249	4.9%
0	105	2.1%
4	73	1.4%
5	55	1.1%
10	45	0.9%
998	36	0.7%
8	22	0.4%
Other values (18)	46	0.9%
(Missing)	3382	67.1%

D26L Quase todos os dias,o(a) Sr(a). falava ou se movia mais lentamente do que o habitual?

Value	Count	Frequency (%)
1.0	622	12.3%
5.0	564	11.2%
8.0	26	0.5%
(Missing)	3825	75.9%

D37D Episódios em que uma pessoa se sente (FRASE-CHAVE) algumas vezes ocorrem sem nenhuma causa e, outras vezes, ocorrem após a morte de uma pessoa próxima, ou em resposta a uma experiência estressante.
Como foi (o seu/a primeira vez que o(a) Sr(a). teve um) episódio desse tipo---começou sem motivo, após a morte de uma pessoa próxima ou em resposta a alguma experiência estressante que tinha acontecido com
o(a) Sr(a).?

Value	Count	Frequency (%)
3.0	735	14.6%
2.0	312	6.2%
1.0	131	2.6%
8.0	24	0.5%
9.0	7	0.1%
(Missing)	3828	76.0%

1 = SEM CAUSA

2 = MORTE DE UMA PESS
PRÓXIMA

3 = RESPOSTA AO
ESTRESSE

8 = NÃO SABE

9 = RECUSOU

Análise Exploratória (Promental)

D64A

A primeira série se refere a dificuldades para dormir:
1: O(A) Sr(a). demorava menos de 30 minutos para pegar no sono.
2: O(A) Sr(a). demorava no mínimo 30 minutos para pegar no sono, menos do que a metade das noites.
3: O(A) Sr(a). demorava no mínimo 30 minutos para pegar no sono, mais do que a metade das noites.
4: O(A) Sr(a). demorava mais de 60 minutos para pegar no sono, mais do que a metade das noites.

Value	Count	Frequency (%)
4	322	6.4%
1	136	2.7%
2	83	1.6%
3	65	1.3%
998	7	0.1%
999	4	0.1%
(Missing)	4420	87.8%

D64B

A próxima série se refere a acordar durante a noite:
Um: O(A) Sr(a). não acordava durante a noite.
Dois:O(A) Sr(a). tinha um sono inquieto, leve, com alguns breves despertares a cada noite.
Três:O(A) Sr(a). acordava no mínimo uma vez durante a noite, mas voltava a dormir facilmente.
Quatro:O(A) Sr(a). acordava mais de uma vez durante a noite e ficava acordado(a) por 20 minutos ou mais, e isso ocorria em mais do que a metade das noites.

Value	Count	Frequency (%)
4	323	6.4%
2	101	2.0%
1	99	2.0%
3	86	1.7%
998	6	0.1%
999	2	< 0.1%
(Missing)	4420	87.8%

D64C

A próxima série se refere a acordar cedo demais de manhã:
Um: Na maior parte dos dias, o(a) Sr(a). acordava no máximo 30 minutos antes do que precisava.
Dois: Em mais da metade dos dias, o(a) Sr(a). acordava mais de 30 minutos antes do que precisava.
Três: Quase sempre o(a) Sr(a). acordava no mínimo cerca de uma hora antes do que precisava, mas às vezes voltava a dormir.
Quatro:O(A) Sr(a). acordava no mínimo uma hora antes do que precisava e não conseguia voltar a dormir.

Value	Count	Frequency (%)
4	204	4.1%
1	191	3.8%
3	118	2.3%
2	70	1.4%
998	28	0.6%
999	6	0.1%
(Missing)	4420	87.8%

Análise Exploratória (Titanic)

Pandas Profiling

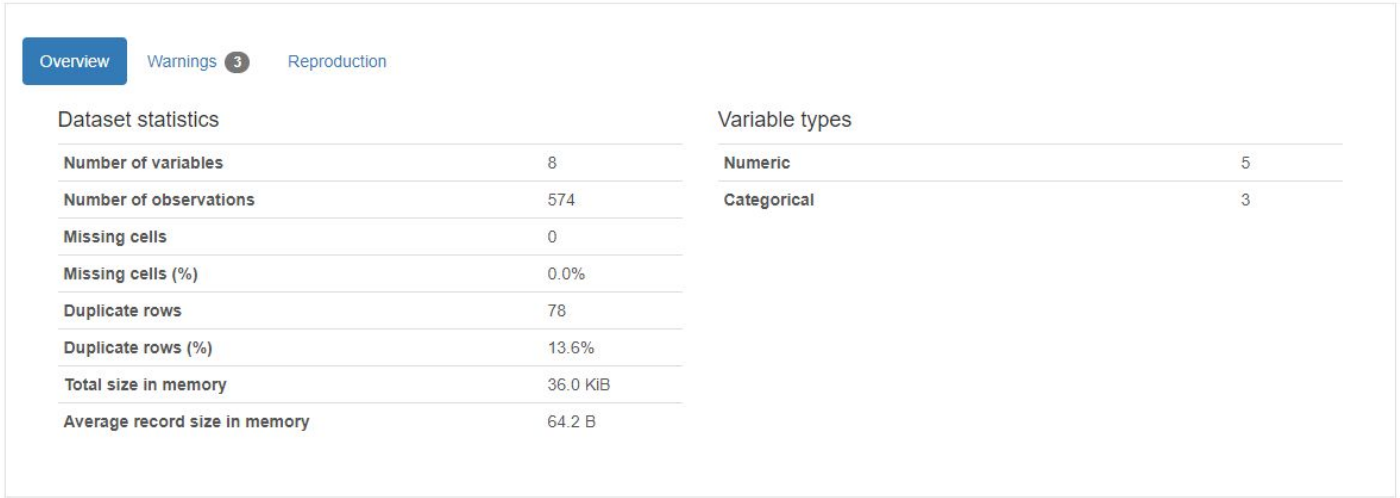
```
[46] pfl_df_titanic = ProfileReport(dffill, title='Pandas Profiling Report', html={'style':{'full_width':False}})
     pfl_df_titanic.to_file(output_file="output_df_depressao2.html")
```

Summarize dataset: 100% 21/21 [00:06<00:00, 1.90it/s, Completed]

Generate report structure: 100% 1/1 [00:05<00:00, 5.03s/it]

Render HTML: 100% 1/1 [00:01<00:00, 1.34s/it]

Export report to file: 100% 1/1 [00:00<00:00, 11.79it/s]



Overview | Warnings 3 | Reproduction

Dataset statistics

Number of variables	8
Number of observations	574
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	78
Duplicate rows (%)	13.6%
Total size in memory	36.0 KiB
Average record size in memory	64.2 B

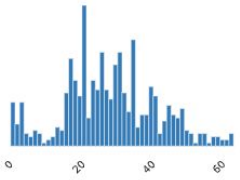
Variable types

Numeric	5
Categorical	3

Análise Exploratória (Titanic)

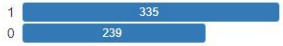
Age
Real number ($\mathbb{R}_{\geq 0}$)

Distinct	78	Minimum	0.42
Distinct (%)	13.6%	Maximum	64
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	28.8321777	Memory size	4.6 KiB



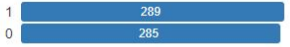
Sex
Categorical

Distinct	2
Distinct (%)	0.3%
Missing	0
Missing (%)	0.0%
Memory size	4.6 KiB



Survived
Categorical

Distinct	2
Distinct (%)	0.3%
Missing	0
Missing (%)	0.0%
Memory size	4.6 KiB



Overview | Warnings 3 | Reproduction

Warnings

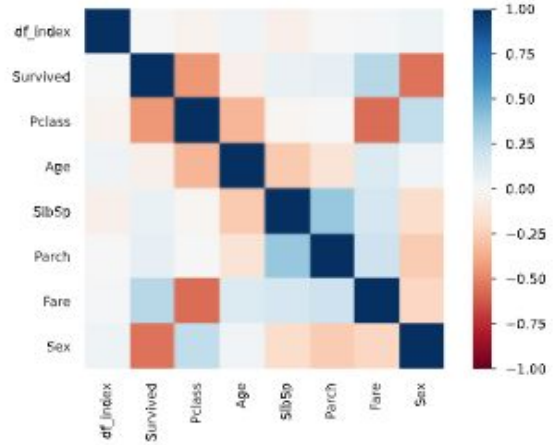
Dataset has 78 (13.6%) duplicate rows	Duplicates
`SibSp` has 394 (68.6%) zeros	Zeros
`Parch` has 416 (72.5%) zeros	Zeros

Análise Exploratória (Titanic)

Correlations

Pearson's r | Spearman's ρ | Kendall's τ | Phik (φk) | Cramér's V (φc)

Toggle correlation descriptions

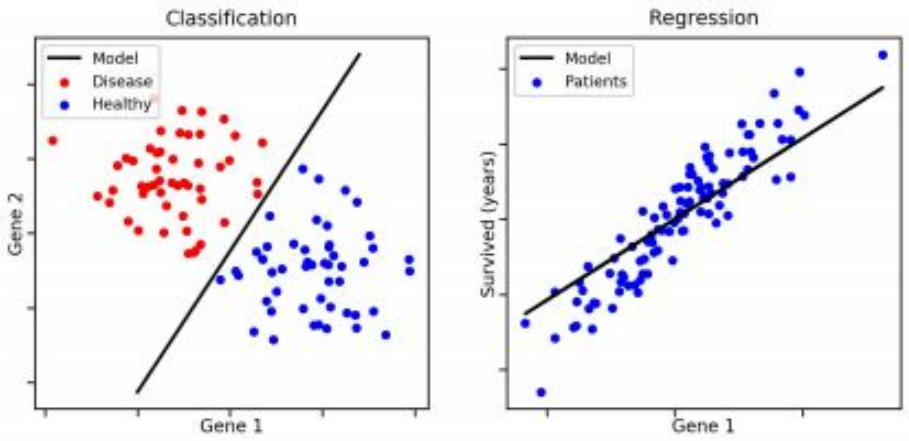


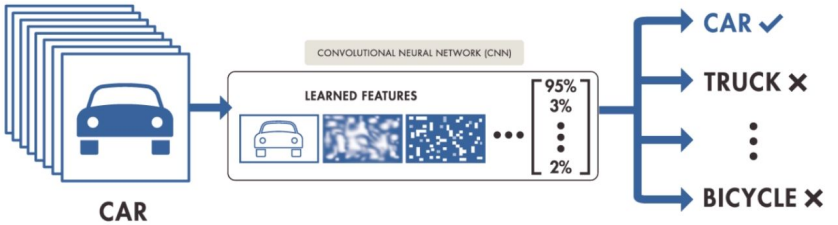
Pearson's r

The Pearson's correlation coefficient (*r*) is a measure of linear correlation between two variables. It's value lies between -1 and +1, -1 indicating total negative linear correlation, 0 indicating no linear correlation and 1 indicating total positive linear correlation. Furthermore, *r* is invariant under separate changes in location and scale of the two variables, implying that for a linear function the angle to the x-axis does not affect *r*.

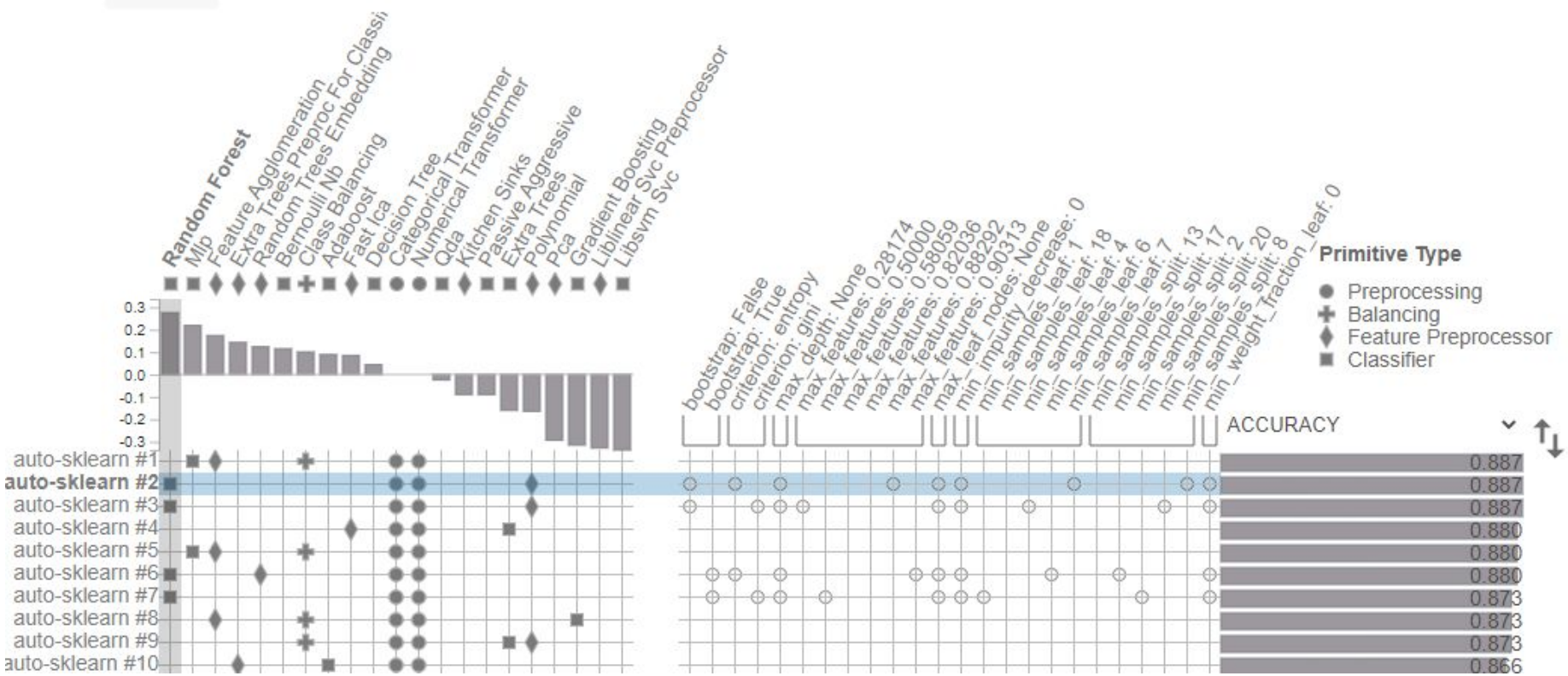
To calculate *r* for two variables *X* and *Y*, one divides the covariance of *X* and *Y* by the product of their standard deviations.

Classificação vs Estimação





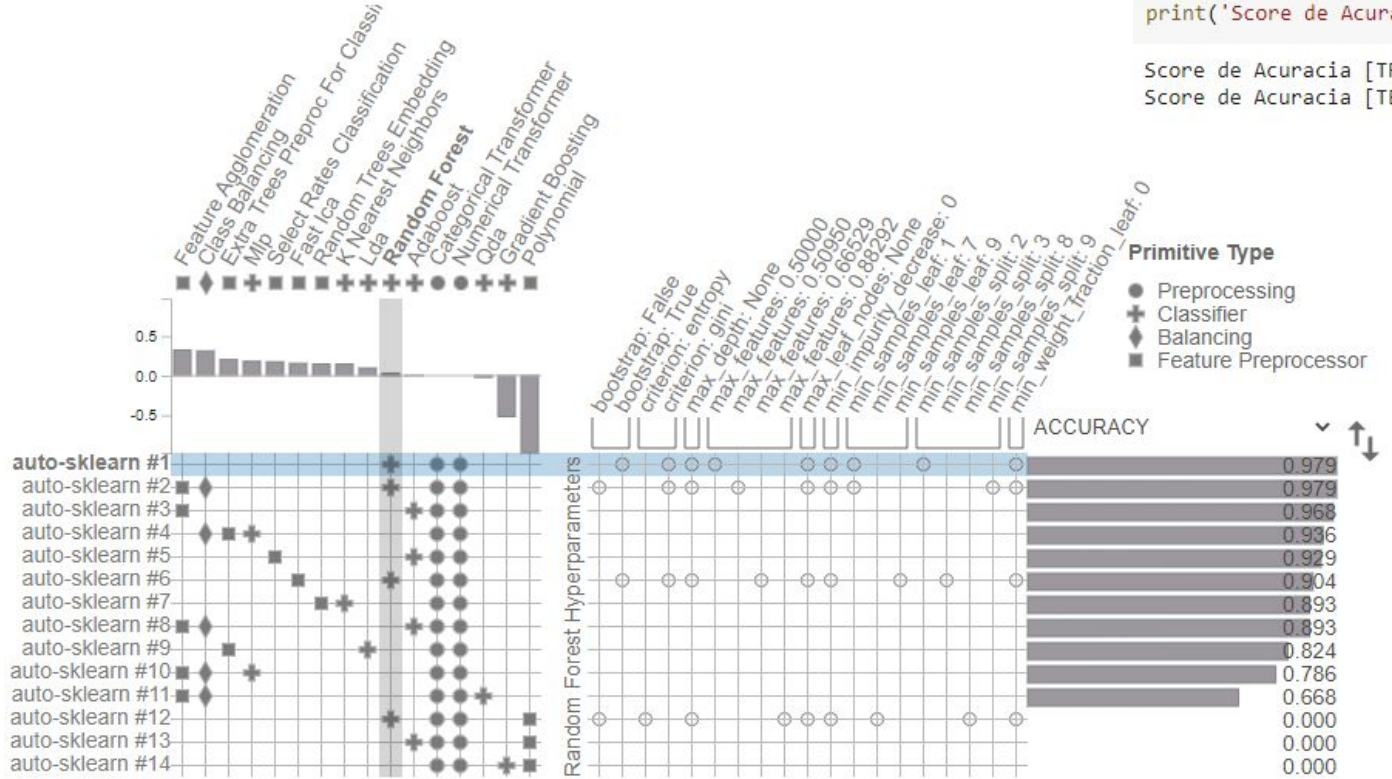
Auto-ML (Titanic)



Auto-ML (Depressão)

```
score_treino2 = random_f2.score(X_train.fillna(0), y_train)
print('Score de Acuracia [TREINO]:',score_treino2)
score_teste2 = random_f2.score(X_test.fillna(0), y_test)
print('Score de Acuracia [TESTE]:',score_teste2)
```

```
Score de Acuracia [TREINO]: 1.0
Score de Acuracia [TESTE]: 0.9774011299435028
```



Logistic Regression - Titanic

```
[41] X_train,X_test,y_train,y_test = train_test_split(dffill.drop(['Survived', 'Fare', 'SibSp', 'Parch'], axis=1),
                                                      dffill['Survived'],
                                                      train_size = 0.75, test_size = 0.25,
                                                      random_state=0, shuffle=True)
```

```
[42] modelo = LogisticRegression(random_state=100, C=3).fit(X_train, y_train)
     score_treino = modelo.score(X_train, y_train)
     print('Score de Acuracia [TREINO]:',score_treino)
     score_teste = modelo.score(X_test, y_test)
     print('Score de Acuracia [TESTE]:',score_teste)
```

```
Score de Acuracia [TREINO]: 0.8023255813953488
Score de Acuracia [TESTE]: 0.8194444444444444
```

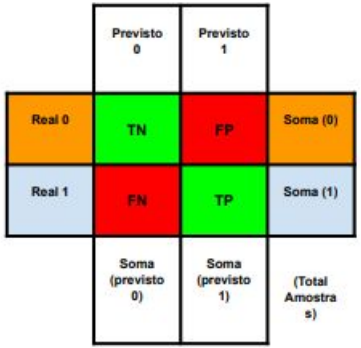
Logistic Regression - Depressão

```
[25] X_train,X_test,y_train,y_test = train_test_split(df_balanceado.drop(['dsm_mddh'], axis=1),
                                                      df_balanceado['dsm_mddh'],
                                                      train_size = 0.8, test_size = 0.2,
                                                      random_state=0, shuffle=True)
```

```
[27] modelo1 = LogisticRegression(random_state=100, C=3, max_iter=100000).fit(X_train.fillna(0), y_train)
     score_treino = modelo1.score(X_train.fillna(0), y_train)
     print('Score de Acuracia [TREINO]:',score_treino)
     score_teste = modelo1.score(X_test.fillna(0), y_test)
     print('Score de Acuracia [TESTE]:',score_teste)
```

```
Score de Acuracia [TREINO]: 0.9752475247524752
Score de Acuracia [TESTE]: 0.9689265536723164
```

Métricas de Desempenho - Titanic




```
[43] predicted = modelo.predict(X_test.fillna(0))
     matriz_confusao = confusion_matrix(y_test, predicted)
     print(matriz_confusao)
```

```
[[52  9]
 [17 66]]
```

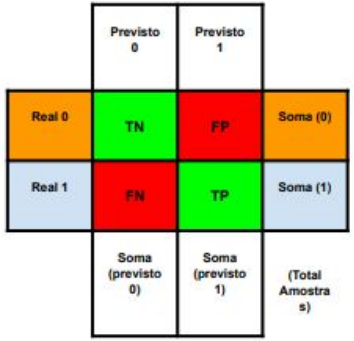
```
[44] report = classification_report(y_test, predicted)
     print('==== Report ====')
     print(report)
```

```
==== Report ====
              precision    recall  f1-score   support

           0       0.75      0.85      0.80        61
           1       0.88      0.80      0.84        83

    accuracy                           0.82       144
   macro avg       0.82      0.82      0.82       144
weighted avg       0.83      0.82      0.82       144
```

```
[45] scoring = ['accuracy', 'recall_macro', 'f1_macro', 'precision_macro']
     kfold = KFold(n_splits = 5, random_state = 5, shuffle = True)
     cv_results1 = cross_validate(modelo1, X_test.fillna(0), y_test, cv = kfold, scoring = scoring)

     print('Accuracy mean:  ', cv_results1['test_accuracy'].mean())
     print('Precision mean: ', cv_results1['test_precision_macro'].mean())
     print('Recall mean:    ', cv_results1['test_recall_macro'].mean())
     print('F1 mean:        ', cv_results1['test_f1_macro'].mean())
```

```
Accuracy mean:   0.833743842364532
Precision mean:  0.8346253076129238
Recall mean:     0.8305777555777556
F1 mean:         0.8241196428644268
```

Métricas de Desempenho - Depressão




```
[27] predicted = modelo1.predict(X_test.fillna(0))
     matriz_confusao = confusion_matrix(y_test, predicted)
     print(matriz_confusao)
```

```
[[170   2]
 [ 14 168]]
```

```
[29] report = classification_report(y_test, predicted)
     print('==== Report ====')
     print(report)
```

```
==== Report ====
              precision    recall  f1-score   support

           1       0.94      0.99      0.97       172
           5       0.99      0.95      0.97       182

    accuracy                           0.97       354
   macro avg       0.97      0.97      0.97       354
weighted avg       0.97      0.97      0.97       354
```

```
[29] scoring = ['accuracy', 'recall_macro', 'f1_macro', 'precision_macro']
     kfold = KFold(n_splits = 5, random_state = 5, shuffle = True)
     cv_results1 = cross_validate(modelo1, X_test.fillna(0), y_test, cv = kfold, scoring = scoring)

     print('Accuracy mean:  ', cv_results1['test_accuracy'].mean())
     print('Precision mean: ', cv_results1['test_precision_macro'].mean())
     print('Recall mean:    ', cv_results1['test_recall_macro'].mean())
     print('F1 mean:        ', cv_results1['test_f1_macro'].mean())
```

```
Accuracy mean:   0.8898591549295775
Precision mean:  0.8912943709946626
Recall mean:     0.8846963623042265
F1 mean:         0.8865291020694229
```