# Big Data Mining & Inteligência Artificial

### Charles Guimarães Cavalcante – RM 334409

## **Análise OLSR**

Com o dataframe Titanic, fazer *data praparation* e propor um modelo preditivo OLSR utilizando Python.

1) Dicionário de dados:

|  |  |  |
| --- | --- | --- |
| **Variável** | **Definição** | **Valores** |
| Survived | Sobreviveu | 0 = Não, 1 = Sim |
| Pclass | Classe da passagem | 1 = 1ª classe, 2 = 2ª classe, 3 = 3ª classe |
| Name | Nome |  |
| Sex | Sexo | male = masculino, female = feminino |
| Age | Idade em anos | valores de 0,42 a 80 |
| Siblings/Spouses Aboard | Número de irmãos/cônjuge abordo | valores de 0 a 8 |
| Parents/Children Aboard | Número de pais/filhos abordo | valores de 0 a 6 |
| Fare | Valor pago pela passagem | valore de 0 a 512,3292 |

2) Bibliotecas utilizadas:

import pandas as pd

import numpy as np

import matplotlib as plt

from sklearn.linear\_model import LogisticRegression

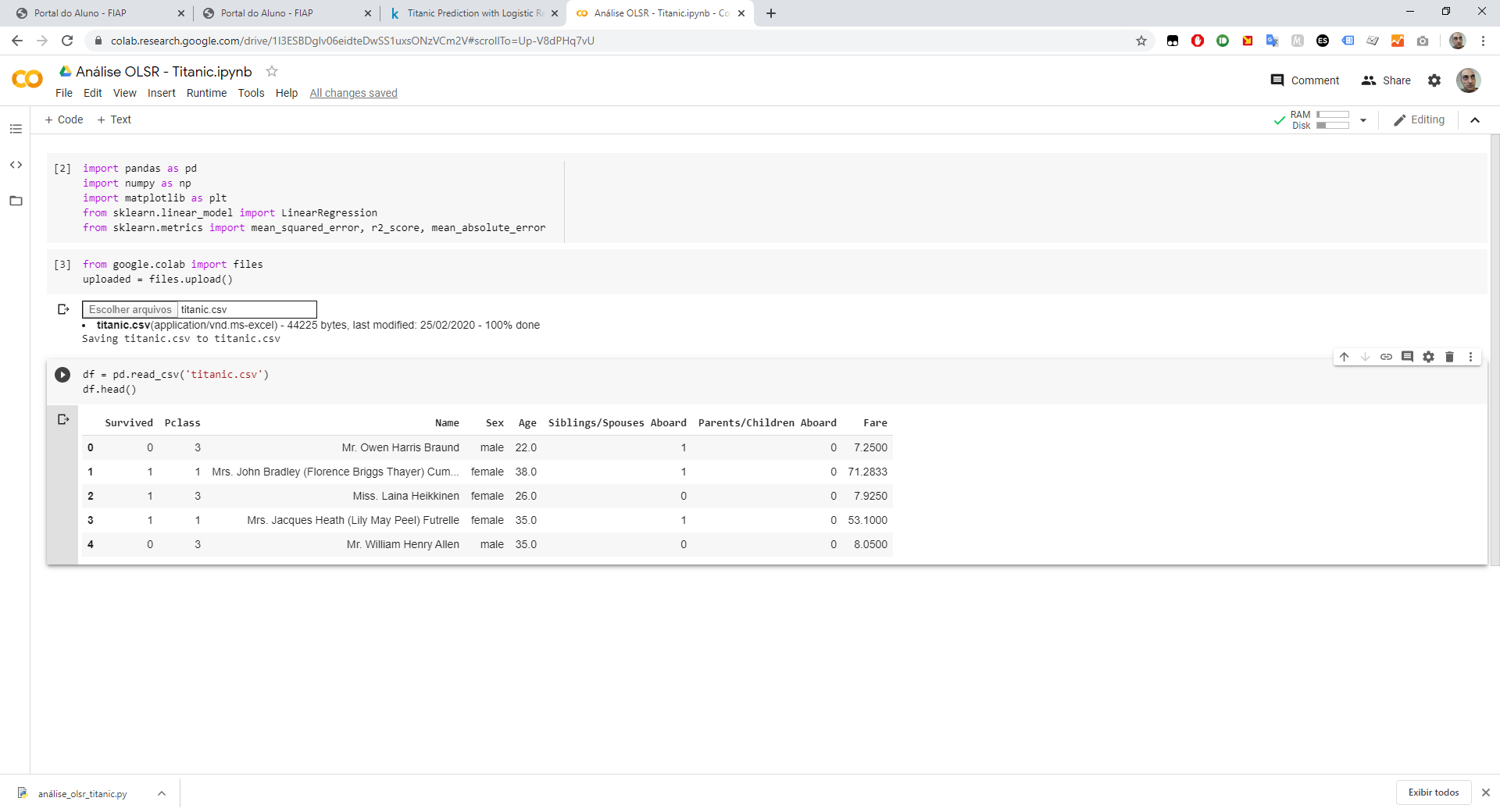
from sklearn.metrics import accuracy\_score

3) Leitura do arquivo:

df = pd.read\_csv('titanic.csv')

df.head()

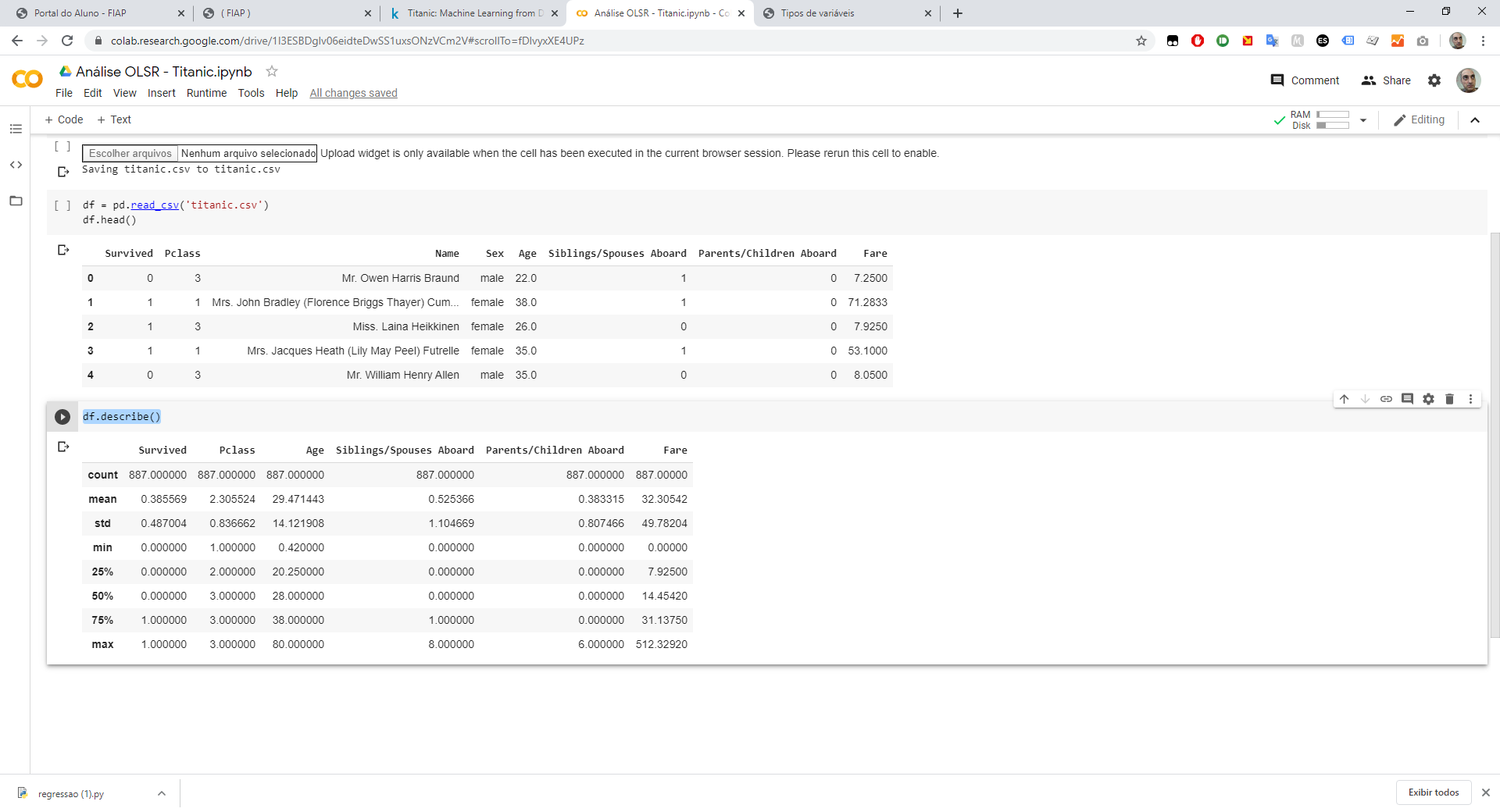
Output com a amostra dos dados:



4) Descrição dos dados:

df.describe()

Output com a descrição dos dados:



df.info()

Output com informações sobre os dados:

RangeIndex: 887 entries, 0 to 886

Data columns (total 8 columns):

Survived 887 non-null int64

Pclass 887 non-null int64

Name 887 non-null object

Sex 887 non-null object

Age 887 non-null float64

Siblings/Spouses Aboard 887 non-null int64

Parents/Children Aboard 887 non-null int64

Fare 887 non-null float64

dtypes: float64(2), int64(4), object(2)

memory usage: 55.6+ KB

O data frame não contém nenhum dado nulo, não será necessário tratamento para ajustar dados nulos.

5) Análise dos dados:

#### Sobreviventes (Survived):

Variável qualitativa nominal, com valores possíveis (0, 1).

df['Survived'].value\_counts()

Output:

0 545

1 342

Name: Survived, dtype: int64

545 mortos e 342 sobreviventes.

#### Classe da passagem (Pclass):

Variável qualitativa nominal, com valores possíveis (1, 2, 3).

df['Pclass'].value\_counts(sort=False)

Output:  
  
1 216

2 184

3 487

Name: Pclass, dtype: int64

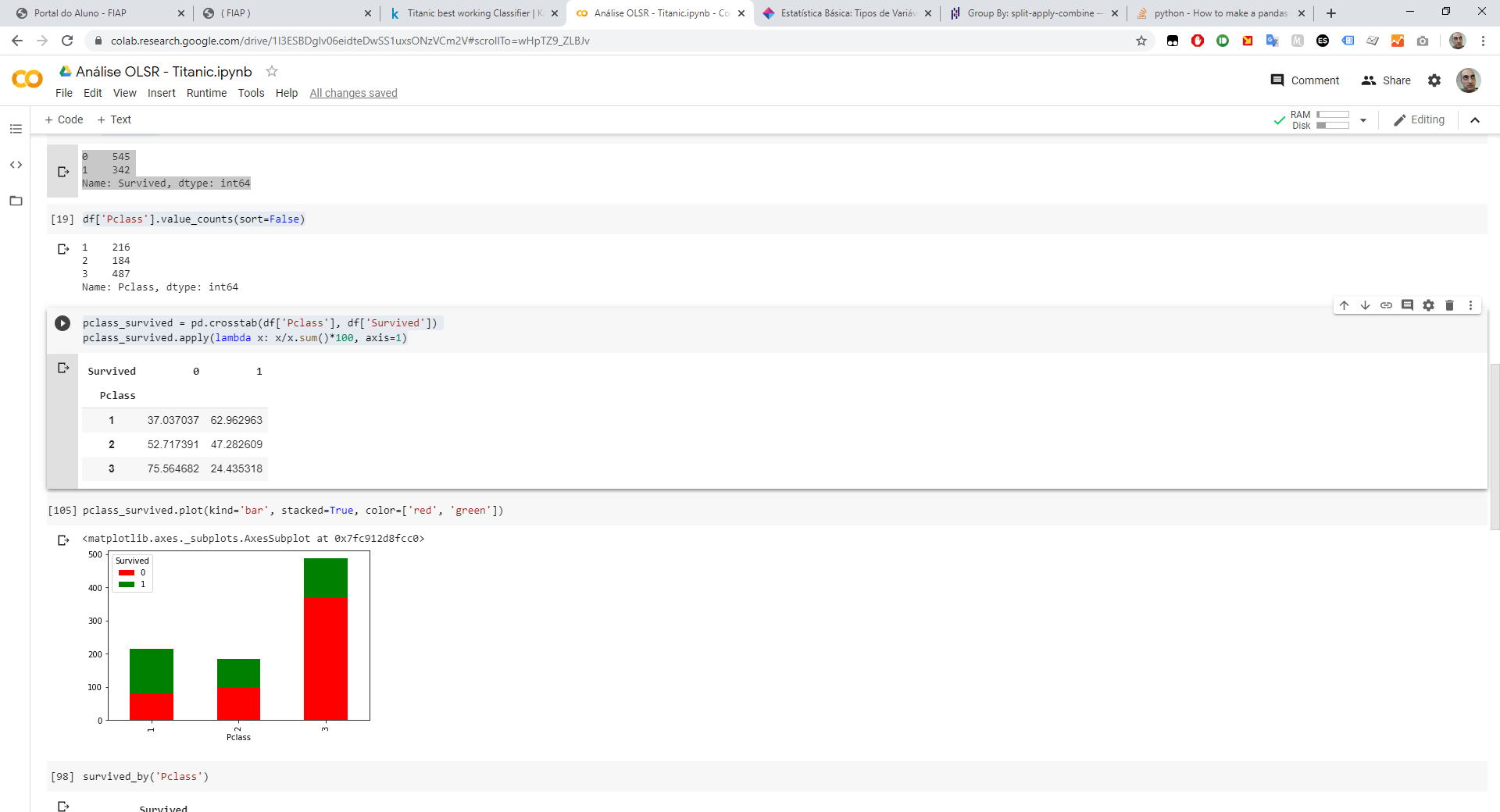
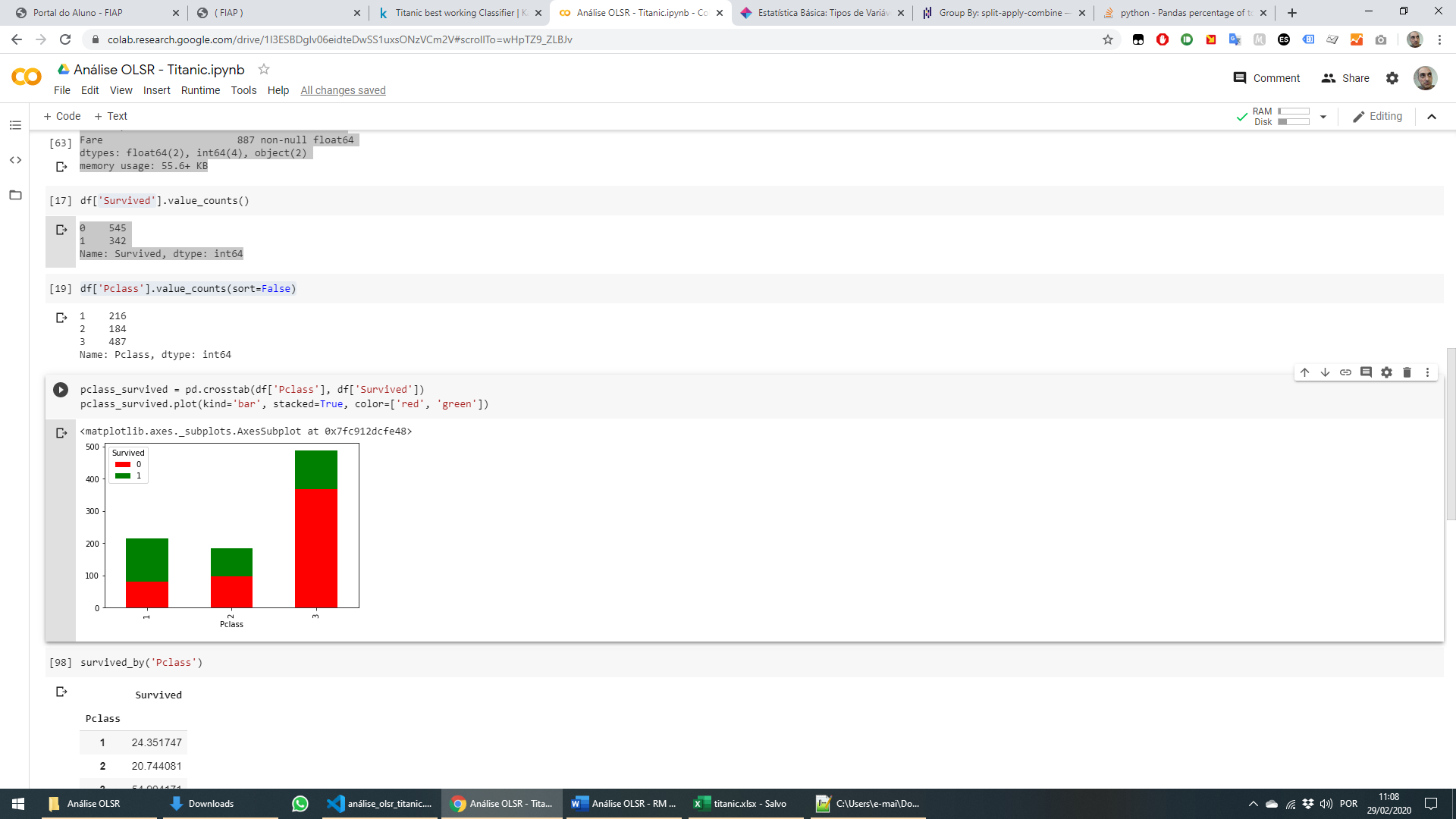
216 na 1ª classe, 184 na 2ª classe e 487 na 3ª classe.

Análise de sobreviventes:

pclass\_survived = pd.crosstab(df['Pclass'], df['Survived'])

pclass\_survived.apply(lambda x: x/x.sum()\*100, axis=1)

pclass\_survived.plot(kind='bar', stacked=True, color=['red', 'green'])



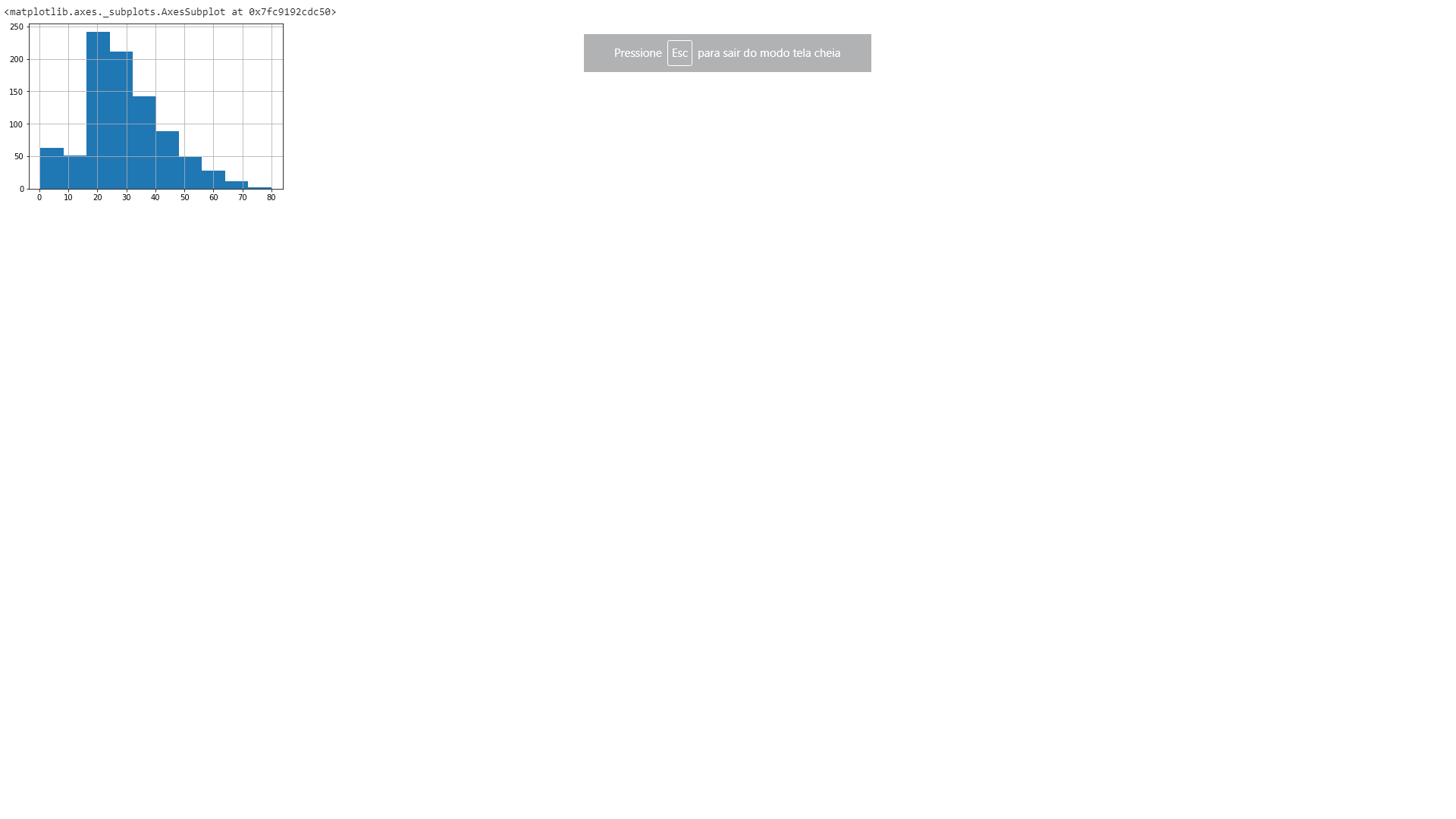
Podemos já constatar que houveram mais sobreviventes na primeira class. Com a terceira classe com maior número de mortos proporcionamente.

#### Idade (Age):

Variável quantitativa contínua, com valores de 0,42 a 80.

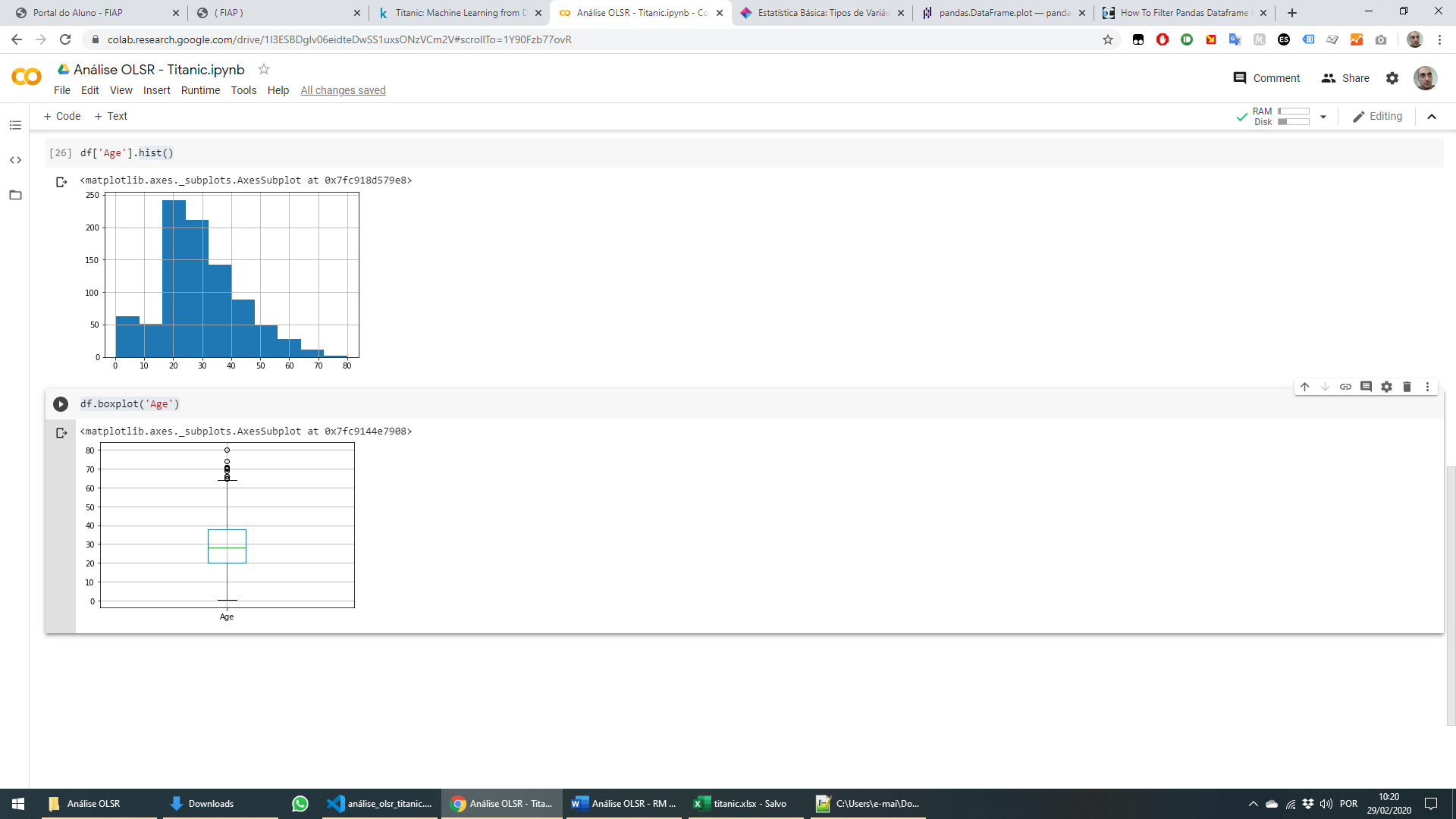
df['Age'].hist()

Output:



Distribuição normal com concentração na faixa entre 18 e 40 anos.

df.boxplot('Age')



Alguns outliers acima dos 60 anos.

#### Número de irmãos/cônjuge abordo (Siblings/Spouses Aboard):

Variável quantitativa discreta, com valores de 0 a 8.

df['Siblings/Spouses Aboard'].value\_counts(sort=False)

Output:

0 604

1 209

2 28

3 16

4 18

5 5

8 7

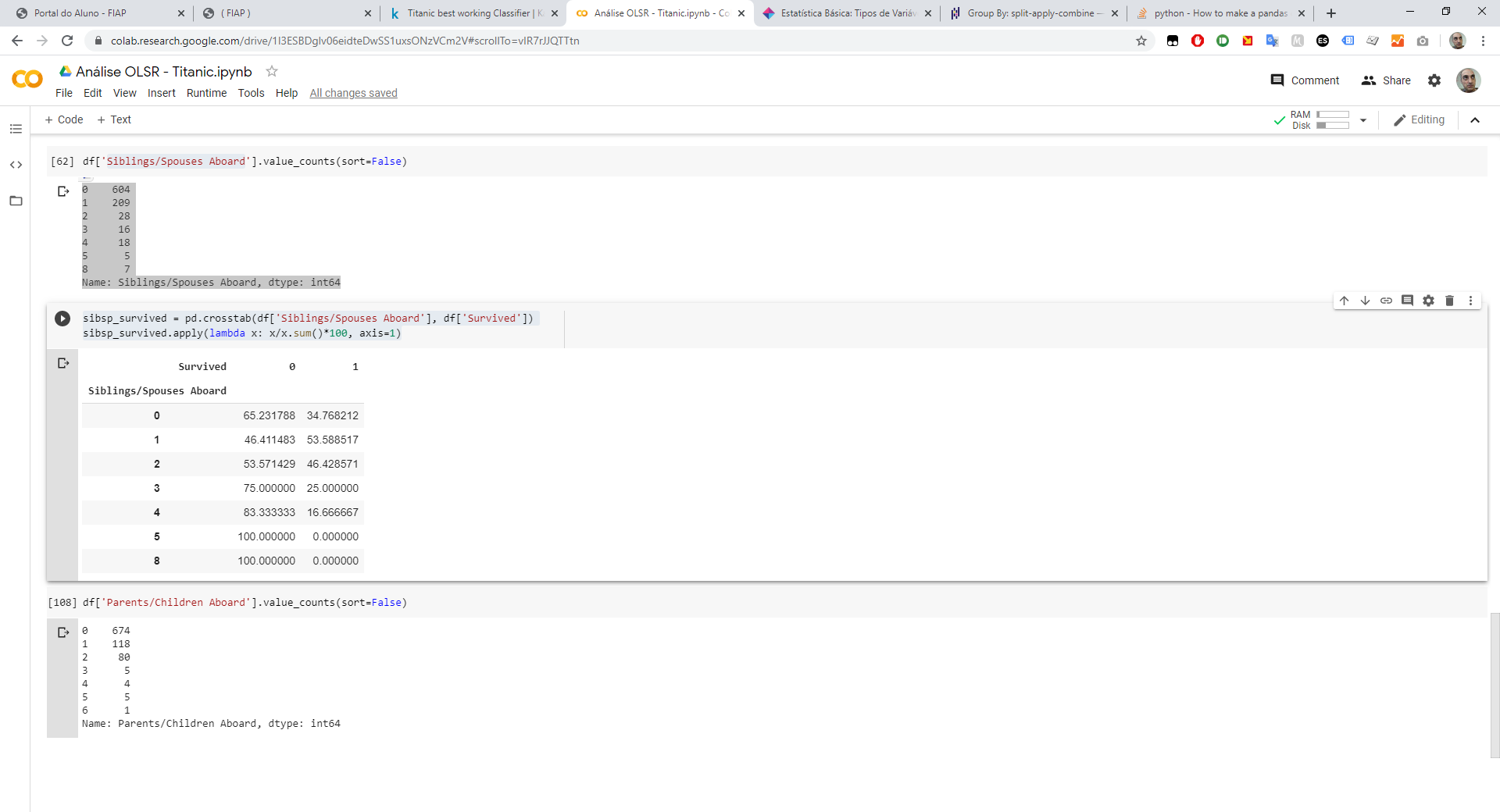
Name: Siblings/Spouses Aboard, dtype: int64

A maioria dos passageiros (604) não tem irmãos ou cônjuge a bordo.

Análise de sobreviventes:

sibsp\_survived = pd.crosstab(df['Siblings/Spouses Aboard'], df['Survived'])

sibsp\_survived.apply(lambda x: x/x.sum()\*100, axis=1)



Não é muito esclarecedor, a maioria sem irmãos ou cônjuge não sobreviveram, assim como quem tem 3 ou mais parentes. Com 1 ou 2 está próximo a 50%.

#### Número de pais/filhos abordo (Parents/Children Aboard):

Variável quantitativa discreta, com valores de 0 a 6.

df['Parents/Children Aboard'].value\_counts(sort=False)

0 674

1 118

2 80

3 5

4 4

5 5

6 1

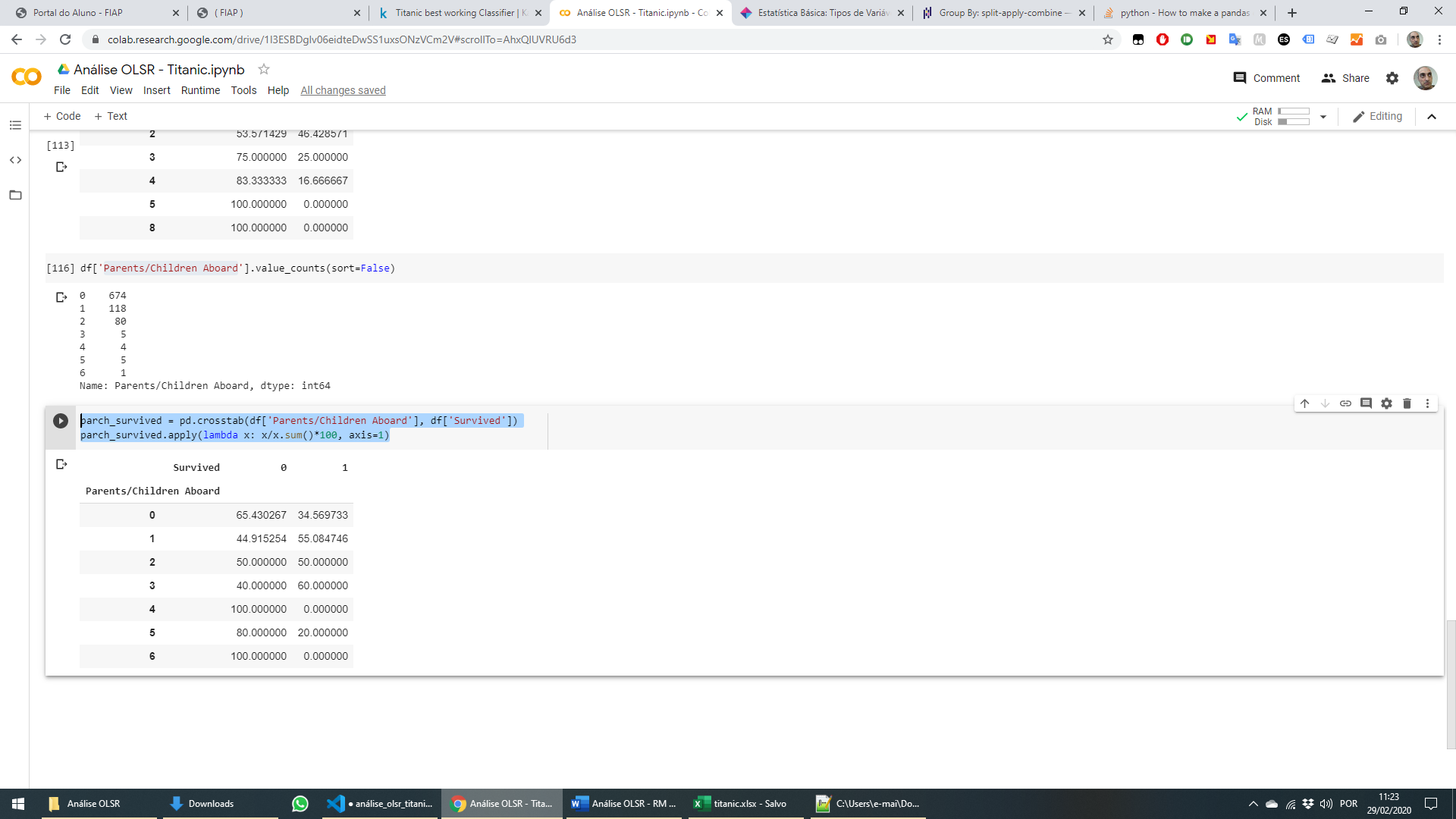
Name: Parents/Children Aboard, dtype: int64

A maioria dos passageiros (674) não tem pais ou filhos a bordo.

Análise de sobreviventes:

parch\_survived = pd.crosstab(df['Parents/Children Aboard'], df['Survived'])

parch\_survived.apply(lambda x: x/x.sum()\*100, axis=1)



Também não é muito esclarecedor, a maioria sem pais ou filhos não sobreviveram, assim como quem tem 4 ou mais parentes. Com 1, 2 ou 3 está próximo a 50%.

#### Família (Siblings/Spouses Aboard + Parents/Children Aboard):

Fiz um estudo criando um novo campo somando os valores de irmãos, cônjuges, pais e filhos.

df['Family'] = df['Siblings/Spouses Aboard'] + df['Parents/Children Aboard']  
df['Family'].value\_counts(sort=False)

0 533

1 161

2 102

3 29

4 15

5 22

6 12

7 6

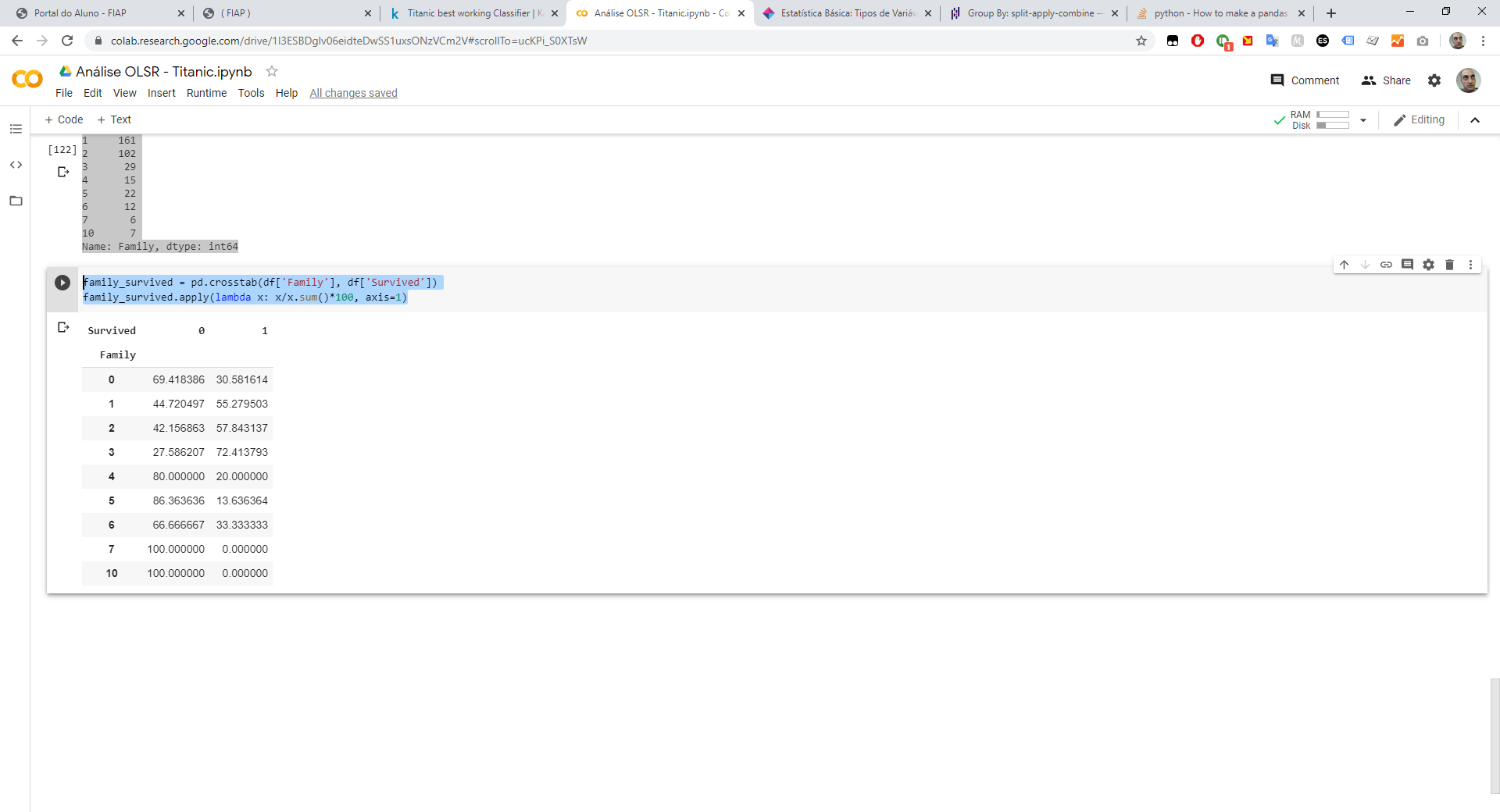
10 7

Name: Family, dtype: int64

A maioria dos passageiros (533) não tem parentes a bordo.

family\_survived = pd.crosstab(df['Family'], df['Survived'])

family\_survived.apply(lambda x: x/x.sum()\*100, axis=1)



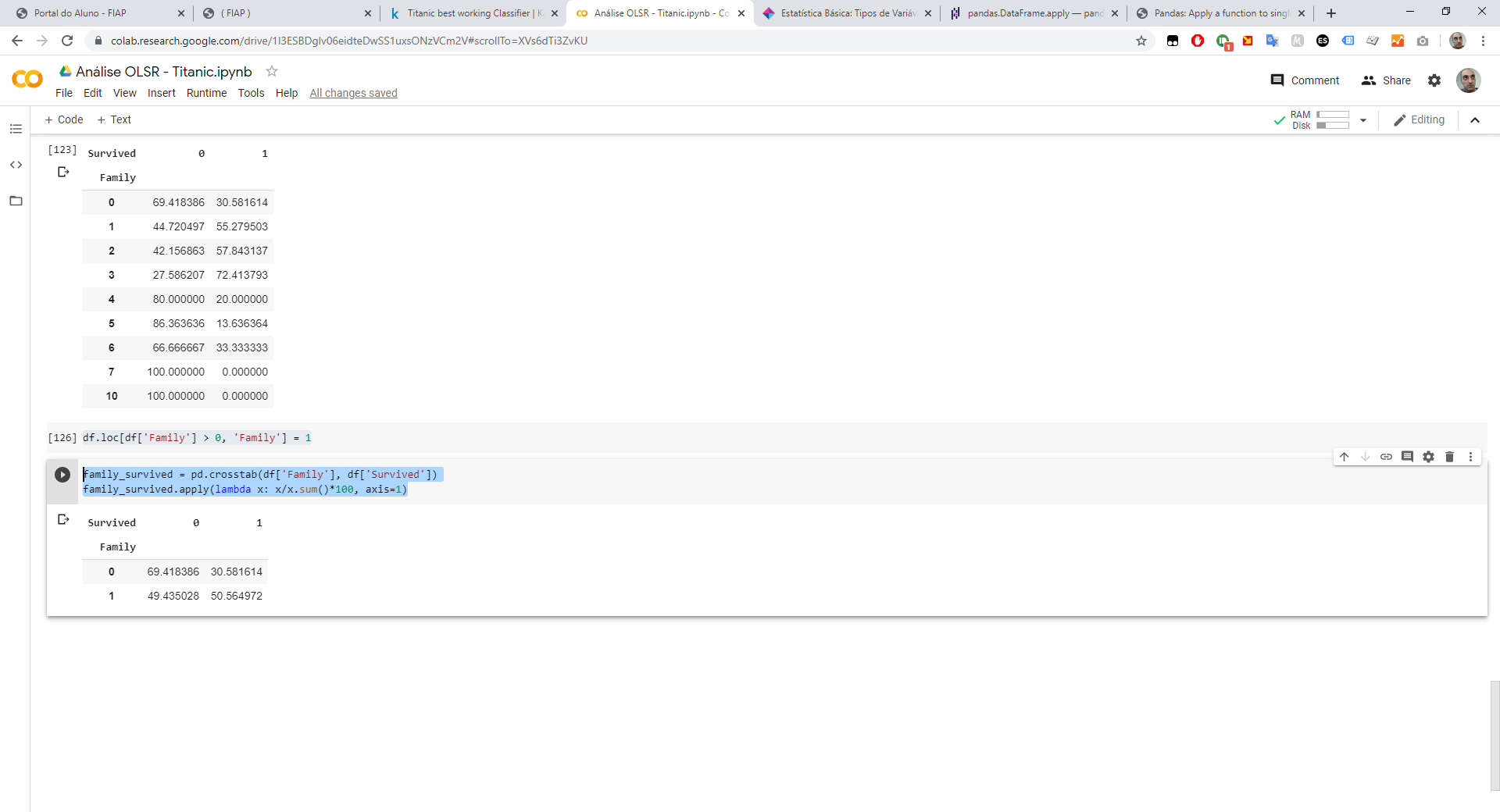
Novamente não ajuda muito, a maioria sem parentes não sobreviveram.

Experimento de transformação em uma variável boleana, 1 para qualquer número de membros.

df.loc[df['Family'] > 0, 'Family'] = 1

family\_survived = pd.crosstab(df['Family'], df['Survived'])

family\_survived.apply(lambda x: x/x.sum()\*100, axis=1)



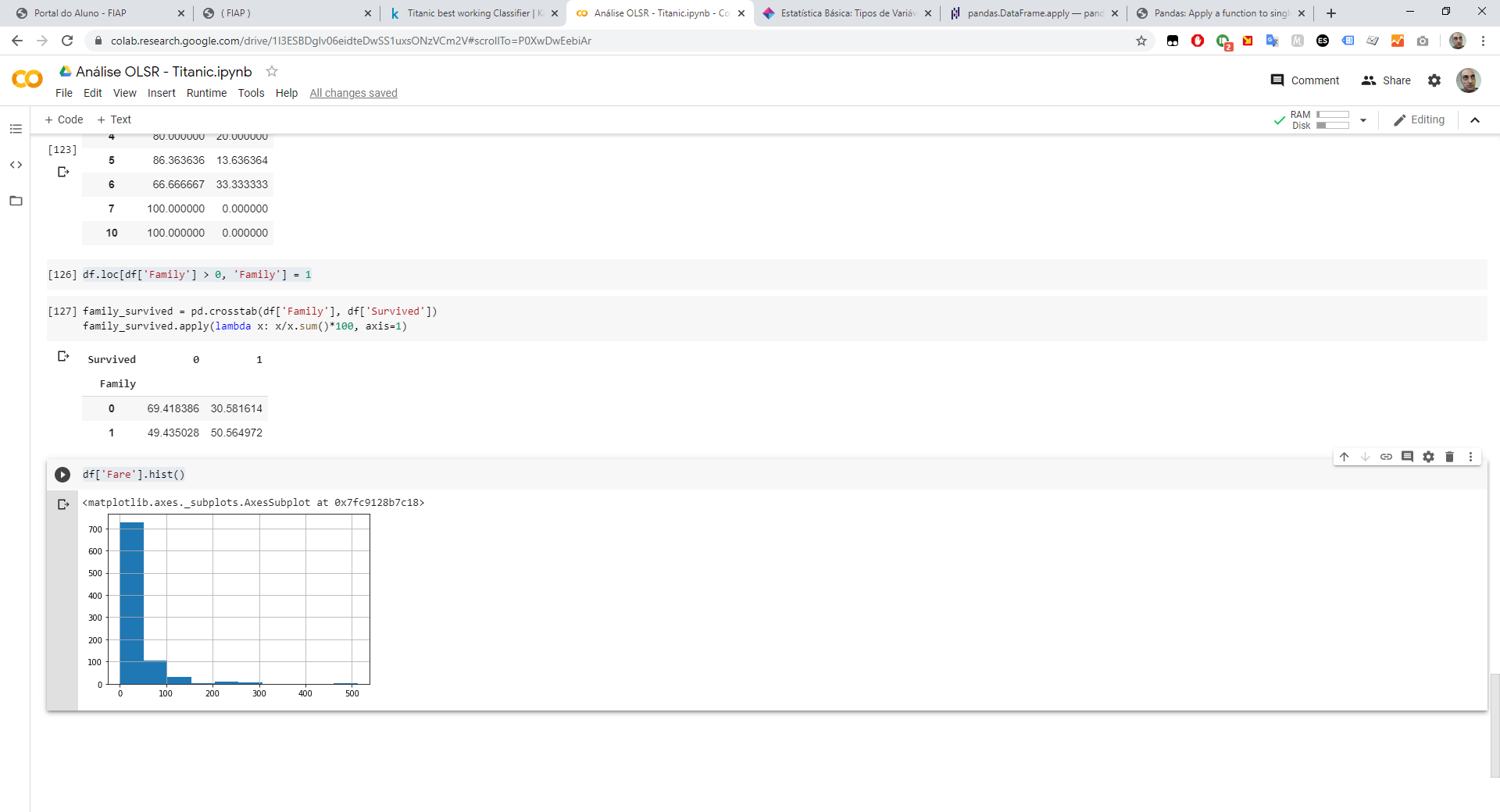
Desta maneira ficou mais claro, passageiros sozinhos morreram mais.

#### Valor pago pela passagem (Fare):

Variável quantitativa contínua, com valores de 0 a 512,3992.

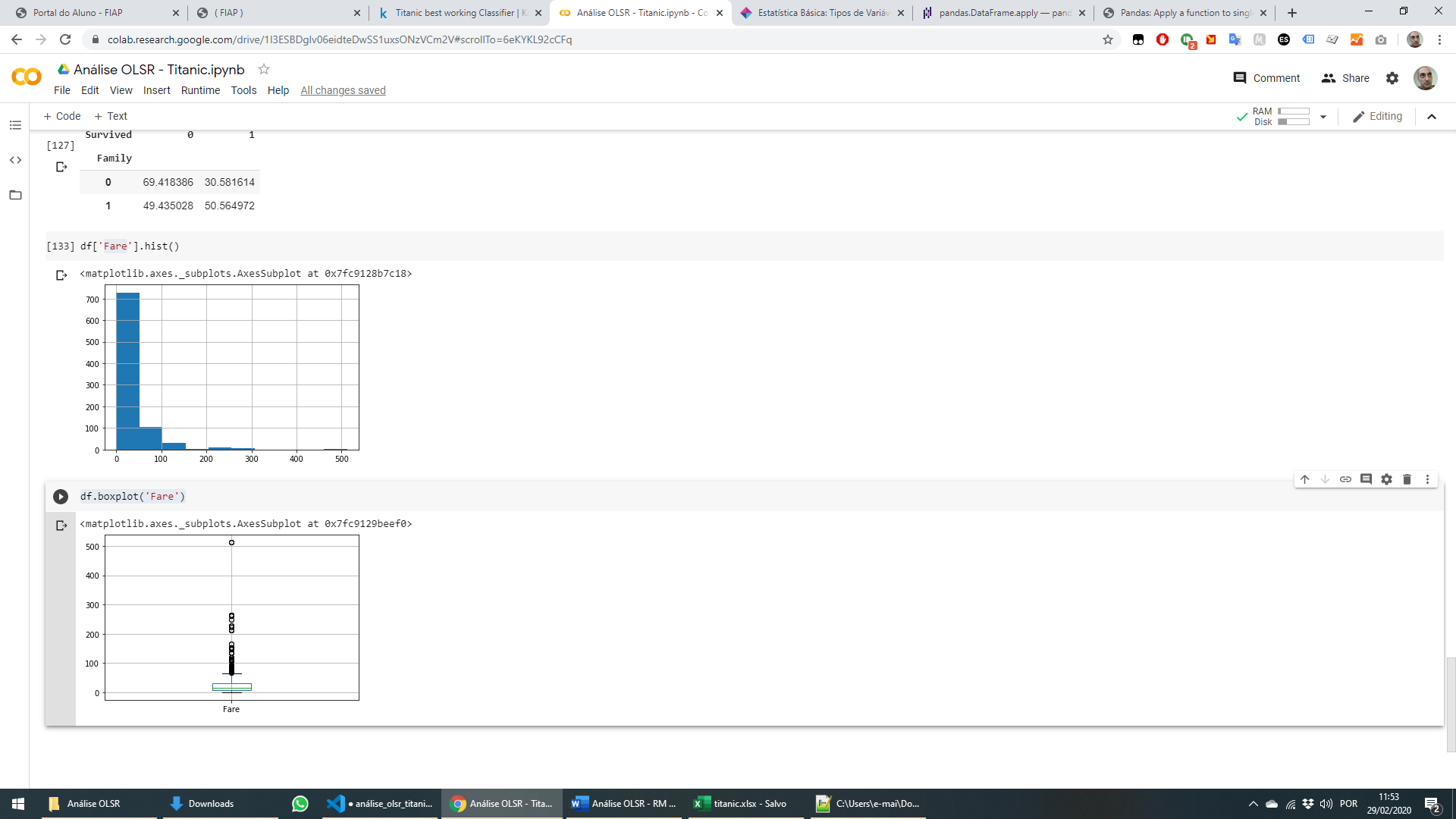
df['Fare'].hist()

Output:



A concentração maior está abaixo dos 50.

df.boxplot('Fare')



6) Regressão:

#### Preparação:

Substituição da variável “Sex” por 1 e 0.

df['Sex Category'] = df['Sex']

df['Sex Category'].replace(['female', 'male'], [0,1], inplace=True)

Categorização da variável “Age”, o desvio padrão é 14.12, separei em categoria de 16 em 16.

df['Age Category'] = pd.cut(df['Age'], bins=[0,16,32,48,64,80], labels=[0,1,2,3,4])

Categorização da variável “Fare”, separeis pelos quartis, 7.92, 14.45 e 31.13.

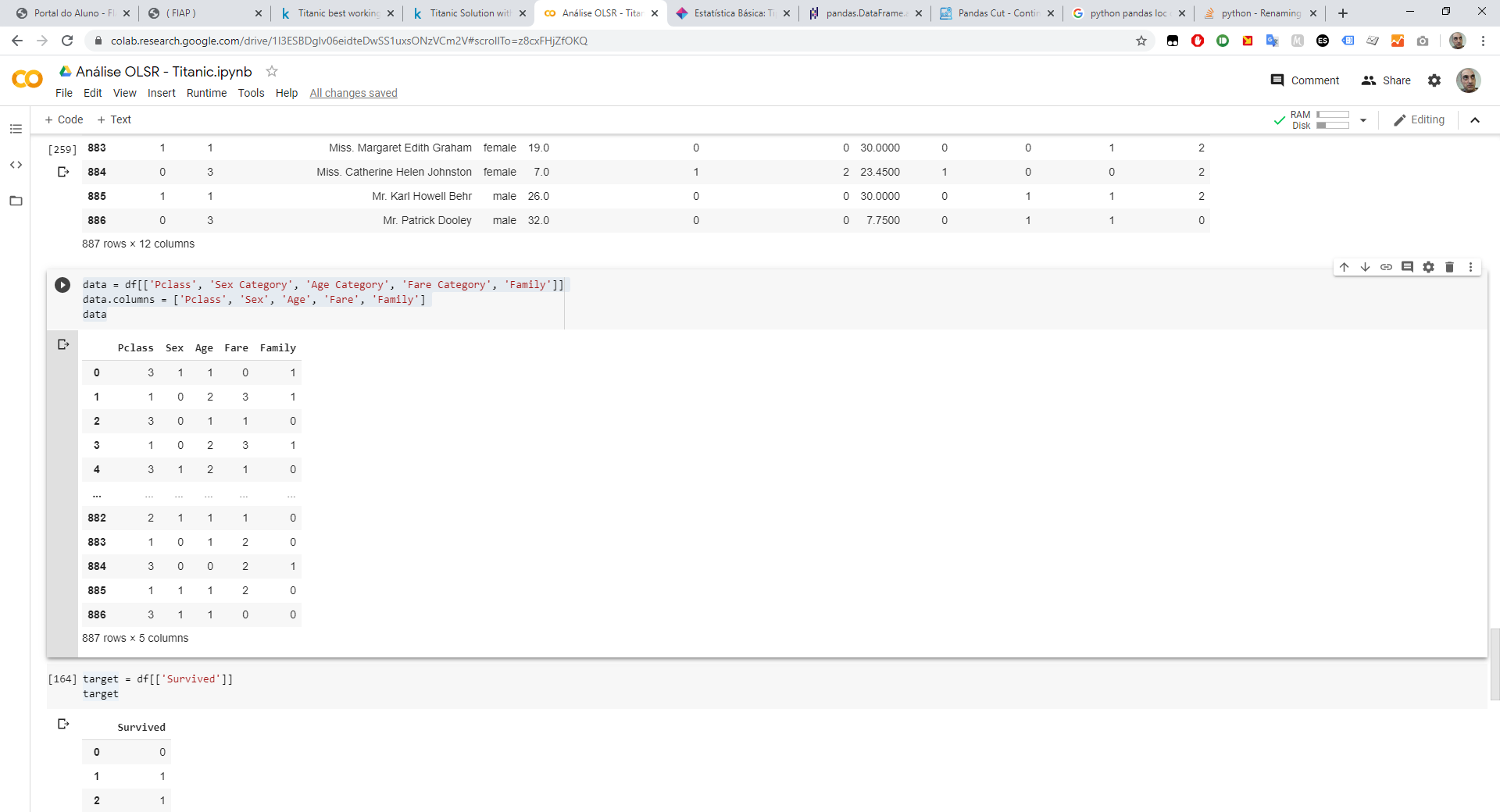
df['Fare Category'] = pd.cut(df['Fare'],bins=[-np.inf,7.92,14.45,31.13,np.inf], labels=[0,1,2,3])

Separação da base com os dados que serão utilizados.

data = df[['Pclass', 'Sex Category', 'Age Category', 'Fare Category', 'Family']]

data.columns = ['Pclass', 'Sex', 'Age', 'Fare', 'Family']

data

Data frame final:  


Variável target.

target = df['Survived']

target

0 0

1 1

2 1

3 1

4 0

..

882 0

883 1

884 0

885 1

886 0

Name: Survived, Length: 887, dtype: int64

Separação da base em treino e teste utilizando o train\_test\_split:

train\_data, test\_data, train\_target, test\_target = train\_test\_split(data, target, test\_size=0.2)

print(len(train\_data))

print(len(test\_data))

709 # para a base de treino

178 # para a base de teste

#### Predição:

Utilizei a **Regressão Logística** pois a variável target é boleana.

model = LogisticRegression().fit(train\_data, train\_target)

predicted = model.predict(test\_data)

score = accuracy\_score(test\_target, predicted)

print('Score: ', score)

Score: 0.7584269662921348

Como a base de treino é gerada randomicamente o score sofre variação, executando o mesmo algoritmo mil vez o score na média foi **0.78**.