

**Centro Universitário Carlos Drummond de Andrade**  
**Análise e desenvolvimento de sistemas**

**Carlos Eduardo da Silva: RA\_201000042**  
**Lucas Azevedo Meireles: RA\_192001161**  
**Thiago dos Santos Ribeiro: RA\_192001098**  
**Melissa Lopes Vieira: RA\_212001945**  
**Danilo Oliveira de Melo RA\_212001813**

**Análise do banco de dados “adult-census-income”**

**Projeto Integrador Orientado**

Cidade - SP  
2021

**Centro Universitário Carlos Drummond de Andrade**

**Análise do banco de dados “adult-census-income”**

Relatório Técnico-Científico apresentado na disciplina de Projeto Integrador para o curso de (Análise e desenvolvimento de sistemas) do Centro Universitário Carlos Drummond de Andrade (UNIDRUMMOND).

Orientador: Me Eduardo Palhares Júnior

São Paulo – SP  
2021

EDUARDO, Carlos; AZEVEDO, Lucas; SANTOS, Thiago; LOPES, Melissa; OLIVEIRA, Danilo. **Título do trabalho.** 00f. Relatório Técnico-Científico. Análise e desenvolvimento de sistema– **Centro Universitário Carlos Drummond de Andrade.** Tutor: (ME. Eduardo Palhares Júnior). Polo (Ponte rasa), 2021.

## **RESUMO**

Neste trabalho iremos analisar uma base de dados chamada *adult-census-income* que por meio da linguagem de programação Python vamos demonstrar vários aspectos da nossa base utilizando buscas, discretização, remoção de dados, analisar correlação etc. o objetivo é entender a implementação de códigos e funcionamento de cada estrutura, utilizando assim um método ágil de adaptação e observando fazendo testes a cada bloco de códigos, e assim observando o comportamento de cada estrutura montando um resultado sólido e funcional e de fácil entendimento respeitando as diretrizes.

**PALAVRAS-CHAVE:** Python, Análise Qualitativa, Análise Quantitativa

## **SUMÁRIO**

(Fonte: Arial ou Times 12; títulos em negrito/ subtítulo sem negrito)

<b>1. INTRODUÇÃO</b>	<b>1</b>
<b>2. DESENVOLVIMENTO</b>	<b>2</b>
2.1 PROBLEMA E OBJETIVOS	2
2.2. JUSTIFICATIVA	2
2. 3. FUNDAMENTAÇÃO TEÓRICA	2
2.4. APLICAÇÃO DAS DISCIPLINAS ESTUDADAS NO PROJETO INTEGRADOR	3
2.5. METODOLOGIA	3
<b>3. RESULTADOS</b>	<b>4</b>
3.1. SOLUÇÃO INICIAL	4
3.2. SOLUÇÃO FINAL	4
<b>4. CONSIDERAÇÕES FINAIS</b>	<b>5</b>
<b>REFERÊNCIAS</b>	<b>6</b>

## 1. INTRODUÇÃO

Atualmente, a tecnologia abrange diversas áreas de conhecimento e a informação está disponível em maior volume do que outrora, seja considerando aspectos pessoais, profissionais ou empresariais. No caso das empresas, esse aumento de volume de informação é significativo à medida que seus processos internos são automatizados, informatizados e que utilizam de alguma tecnologia para o armazenamento da informação. Os bancos de dados são ferramentas responsáveis pelo armazenamento de informações geradas por determinado software são estruturas importantes neste contexto.

Temos uma base de dados a ser analisada, o primeiro passo é demonstrar os códigos que utilizaremos para fazer as buscas, remoções de dados inúteis ou ausentes, ordenação a discretizações de dados e os resultados esperados, assim tratando e estruturando os códigos não funcionais que podem atrapalhar nossa análise. Utilizando o "google collab" uma ferramenta online indicada pelo orientador, onde codificando e debugando os resultados aparecem como qualquer outra IDE.

- Criar uma biblioteca.
- Importar database “*adult-census-income*”.
- Carregar os arquivos do database.
- Discretização do “income”.
- Análise de variáveis.
- Discretização das variáveis.
- Variável sexo “Gênero”.
- Dados faltantes.
- Dados da variável “Native country”(País de origem).
- Variável “Marital status” (Matrimônio).
- Variável “Relationship” (Relacionamento).
- Variável “Race” (Raça).
- Variável "Work Class" (Tipo de ocupação).
- Variável “Capital gain” (Ganho de capital).
- Variável “Capital loss” (Perda de Capital).
- Removendo variáveis não explicativas.
- Variável “Education” (Nível Educacional).

- Variável "Occupation" (Ocupação).
- Remoção.
- Variáveis não discretizadas.
- Variável "Hours per week" (Horas trabalhadas por semana).
- Variável "Education num" (Número de anos na escola).
- Variável "Age" (Idade).
- Analisando Correlação.
- Construção dos classificadores.
- Importando o data Split.
- Participando dos dados.
- Importando os classificadores.
- Treinamento.
- Importando os métodos de treinamento.
- Importando as bibliotecas de avaliação.

## **2. DESENVOLVIMENTO**

### **2.1 Objetivos**

O objetivo geral deste trabalho é mostrar com detalhes a forma de montagem e estruturação de análise do banco de dados, utilizando análise de dados qualitativos e quantitativas desde o início da escolha das bibliotecas utilizadas até a avaliação de resultados.

### **2.2. Justificativa e delimitação do problema**

Os bancos de dados estão cada vez mais presentes em nosso dia a dia, visto que a maioria das atividades que realizamos envolve, direta ou indiretamente, o uso de uma base de dados. Segundo Elmasri e Navathe (2011, p.3), um banco de dados “é uma coleção de dados relacionados”, ou seja, sempre que possuímos um grupo de informações que se relacionam entre si e tratam de um mesmo assunto, temos um banco de dados.

Banco de dados relacionais hoje são a tecnologia predominante para armazenar dados estruturados. “A partir de 1970 o armazenamento de dados baseado em cálculo relacional foi bastante utilizado e muitos pensaram ser a única alternativa para o armazenamento de dados acessível por vários clientes de uma forma consistente” (FERNANDES, 2013, p.2). Porém, com a proliferação das redes sociais e a intensa interação entre usuários da internet, cada dia mais e mais dados são inseridos e, recuperá-los para análise em alta velocidade é quase impossível.

Um dos desafios deste trabalho é encontrar erros nos códigos, que podem não trazer os resultados esperados, a equipe se reuniu e debatemos qual era a melhor solução, e chegamos em um comum acordo, que todos os integrantes participassem ativamente para um resultado positivo e de fato funcional.

### **2.3. Fundamentação teórica**



Segundo Elmasri e Navathe (2011, p. 2), “os bancos de dados e os sistemas de bancos de dados se tornaram componentes essenciais no cotidiano da sociedade moderna. No decorrer do dia, a maioria de nós se depara com atividades que envolvem alguma interação com os bancos de dados”.

A estabilidade dessa área é de grande valor. Os dados de uma organização perduram muito mais tempo do que seus programas. É importante termos um armazenamento de dados estável, que seja bem compreendido e que possa ser acessado a partir de muitas plataformas de programação de aplicativos, (SADALAGE, FOWELER, 2013, p.13)

Date, (2003, p.10) diz que, “um banco de dados é uma coleção de dados persistentes, usada pelos sistemas de aplicação de uma determinada empresa”. Em outras palavras, um banco de dados é um local onde são armazenados dados necessários ao suporte das atividades de determinada entidade, sendo este diretório a fonte de dados para os softwares atuais e os que vierem a existir. O crescimento do armazenamento virtual se dá de forma exponencial, após os anos 2000, preocupando vários desenvolvedores por conta da falta de espaço em armazenamento. Dados da IBM apontam que em 2015 “diariamente são criados 4,5 quintilhões de bytes de dados a partir de uma variedade de fontes de informação”, (OLIVEIRA, 2015). Este fato é decorrente da, cada vez mais frequente, conexão de empresas à internet, como redes sociais, companhias de telefonia, serviços de streaming, dentre outras.

O paradigma sobre o armazenamento de dados em aplicações, traz à tona estudos sobre as características necessárias em modelos de banco de dados para que esse crescente volume de informações seja gerenciado de forma adequada, (OLIVEIRA, 2015). Oliveira (2015), et al, diz que: Um modelo de banco de dados é uma descrição dos tipos de informações que estão armazenados em um banco de dados. Um dos modelos de banco de dados amplamente usados até os dias de hoje é o modelo relacional. Porém, constatou-se que os modelos de banco de dados relacionais apresentam limitações ao trabalhar com grandes volumes de dados. A difusão do tema “modelos de bancos de dados não-relacionais” provém desse aumento do volume de dados gerados na web, e da percepção de que o modelo relacional pode se mostrar ineficiente quando utilizado para administrar grande quantidade de dados, (OLIVEIRA, 2014).

## **2.4. Aplicação das disciplinas estudadas no Projeto Integrador**

Este item do referencial teórico deve indicar os conteúdos das disciplinas estudadas no curso que foram abordados no projeto. Espera-se que os estudantes relacionem, de forma clara e coerente, o conteúdo estudado à solução desenvolvida durante o projeto.

Para isso, indique as disciplinas estudadas, materiais e conteúdo específicos que foram usados na construção do trabalho.

**R:** Usamos como parâmetro para o desenvolvimento desse projeto o vídeo que foi nos deixado, com esse vídeo conseguimos baixar o banco de dados e conseguimos estruturar os dados. Neste trabalho desenvolvemos códigos em python, usamos o banco de dados, a matéria usada para o desenvolvimento deste trabalho foi estrutura de dados

## **2.5. Metodologia**

Metodologia utilizada neste projeto foi a mineração de dados utilizando as análises Qualitativa e Quantitativa.

Assim, indique as estratégias adotadas em cada etapa do projeto:

- Ouvir e Interpretar o Contexto:

- Descrição do contexto em que o projeto foi realizado;
- Em nosso trabalho usamos reuniões para organização, usamos de base às aulas e vídeo no youtube.
- Perfil dos sujeitos participantes, se for o caso;
- Carlos, Lucas, Thiago, Melissa e Danilo.
- Como as informações iniciais foram coletadas: observação, entrevista, formulário, questionário etc.

Coletamos as informações através de observação, através de reuniões conseguimos coletar todas as informações que achamos necessárias para construirmos o trabalho mais coerente possível com que aprendemos com as aulas

- **Criar / Prototipar:**

Análise dos dados, por exemplo, estratégias referentes à pesquisa qualitativa ou quantitativa.

**R:** Priorizamos uma pesquisa qualitativa por suma importância de qualidade em um código limpo e adequado para obtermos uma estrutura linear e totalmente clara, Podemos observar que na imagem acima usamos uma estratégia para podermos organizar melhor a estrutura, e então podermos dar o fim nos erros e seguirmos em frente a nossa meta de finalização.

Descrição das soluções encontradas ou desenvolvidas para o problema investigado.

**R:** Bom tivemos muitos problemas para identificar os erros apresentados no sistema da plataforma, uma delas foi analisar o princípio do erro e como o consertar de forma simples e correta, a solução que tivemos em grupo foi elaborar uma pesquisa desejada para implementar na estrutura em que ocorreu o erro, após identificá-lo analisamos como corrigi-lo e então entramos em um acordo, daí em diante conseguimos facilmente identificar os erros que vieram a seguir.

#### **- Implementar / Testar:**

Como a solução foi testada? Que devolutivas sobre a solução o grupo conseguiu coletar?

**R: a.** Como tivemos bastante dificuldade com as implementações e com os erros dos códigos, conseguimos de certa forma utilizar conteúdos explicativos para podermos dar continuação no contexto do protótipo, podendo de certa forma analisar o código e submeter as melhores ideias para solucionar os erros;

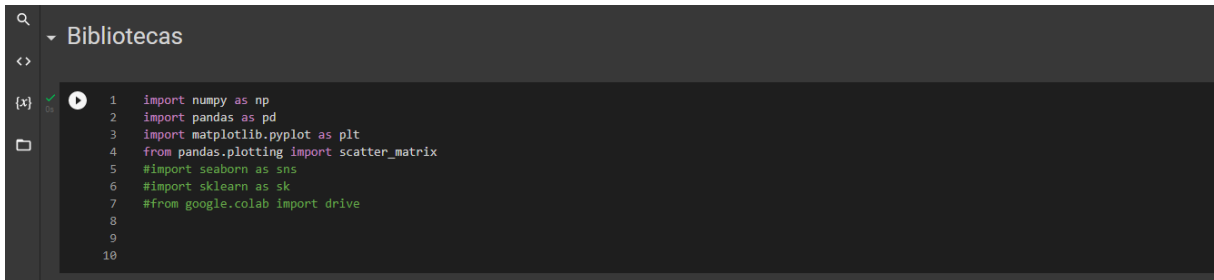
**b.** A partir do vídeo de orientação conseguimos alienar uma solução para os problemas interligados ao código, podendo então continuar com o desenvolvimento da base de estrutura e assim conseguindo ter uma análise melhor do conteúdo.

### **3. RESULTADOS**

Foi relatado um problema no código "plot correlação" que não funcionou e com pesquisa e reiniciado todo sistema ele voltou a funcionar, foi constatado que o erro foi no ide "google colab".

### 3.1. Solução inicial

- Criar uma biblioteca.



```
[x] 1 import numpy as np
    2 import pandas as pd
    3 import matplotlib.pyplot as plt
    4 from pandas.plotting import scatter_matrix
    5 #import seaborn as sns
    6 #import sklearn as sk
    7 #from google.colab import drive
    8
    9
   10
```

O primeiro passo do projeto foi a utilização das bibliotecas, são coleções de fontes de informação. No entanto, em vez de livros, nas bibliotecas Python você recupera módulos que você aplicará durante o processo de programação.

- Importar database “*adult-census-income*”.

A importação da base de dados foi feita através de link redirecionando e carregando do google drive, esta base de dados foi indicada pelo orientador do curso.

- Carregar os arquivos do database.

```
+ Código + Texto
[2] 1 from google.colab import drive
1 drive.mount("/content/drive")
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
1 df = pd.read_csv("/content/drive/MyDrive/Documentos_faculdade/adult.csv",1,",")
2 data = [df]
3 print("Inicial dataset \n\n", df.head(10), "\n")
4
Inicial dataset
  age  workclass  fnlwtg  ...  hours.per.week  native.country  income
0  90         ?    77053  ...             40  United-States  <=50K
1  82   Private  132870  ...             18  United-States  <=50K
2  66         ?   186061  ...             40  United-States  <=50K
3  54   Private  140359  ...             40  United-States  <=50K
4  41   Private  264663  ...             40  United-States  <=50K
5  34   Private  216864  ...             45  United-States  <=50K
6  38   Private  150601  ...             40  United-States  <=50K
7  74  State-gov   88638  ...             20  United-States  >50K
8  68  Federal-gov 422013  ...             40  United-States  <=50K
9  41   Private   70037  ...             60         ?         >50K

[10 rows x 15 columns]
0s conclusão: 18:35
```

Carregando os arquivos da base de dados, já podemos ver a primeira consulta que foi as dez primeiras linhas onde se mostra o conteúdo a ser utilizado.

- Discretização do “income”.

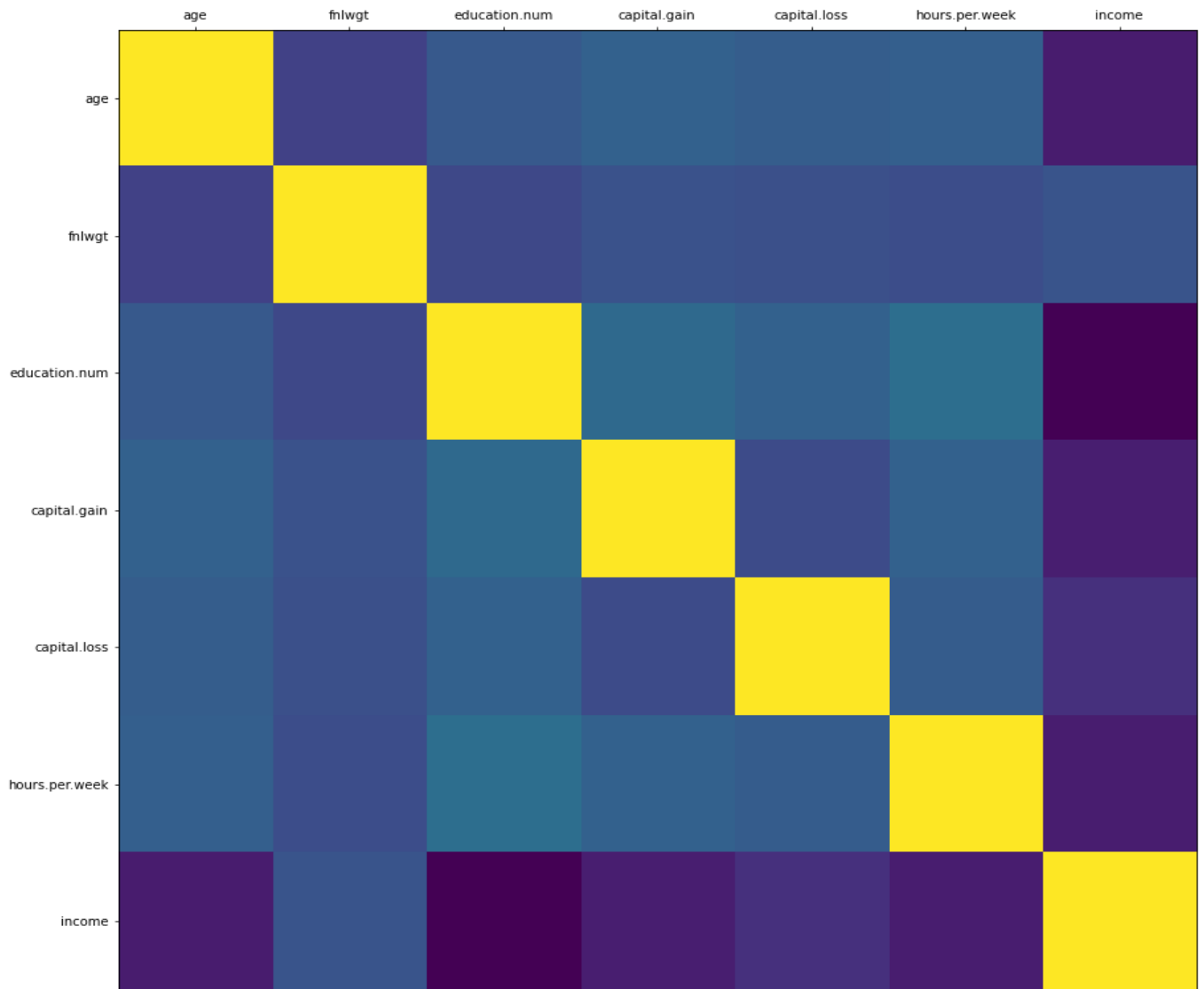
```
Discretização "income"
[ ] income_map = {'<=50K':1,'>50K':0}
#df['income'] = df['income'].map(income_map).astype(int)
df['income']=df['income'].map({'<=50K': 1,'>50K': 0, '<=50K.': 1,'>50K.': 0})
print("income classified by binary \n\n", df.head(10), "\n")
```

Essa parte do código mostra um mapeamento dos valores de salários de quem ganha um valor menor ou igual a 50.000 ou maior que 50.000, transformando em binário.

- Análise de variáveis.

```
projeto_pio 2/2021.ipynb
Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda Todas as alterações foram salvas
+ Código + Texto
Analize das variáveis
Função que plota a matriz de correlação
+ Código + Texto
[7] 1 def plot_correlation(df, size=10):
2     corr= df.corr()
3     fig, ax =plt.subplots(figsize=(15,15))
4     ax.matshow(corr)
5     plt.xticks(range(len(corr.columns)),corr.columns)
6     plt.yticks(range(len(corr.columns)),corr.columns)
7     plt.show()
```

Nessa parte do código temos variáveis sendo analisadas, e temos também uma função que plota a matriz de correlação.



- Variável sexo “Gênero”.

```

+ Código + Texto
RAM
Disco
Editar
Variável "sex (gênero)"
1 print(df[['sex', 'income']].groupby(['sex']).mean(), "\n")
sex
income
Female 0.890539
Male 0.694263

[11] 1 sex_map = {'Male':1, 'Female':0}
2 df['sex'] = df['sex'].map({'Male':1, 'Female':0})

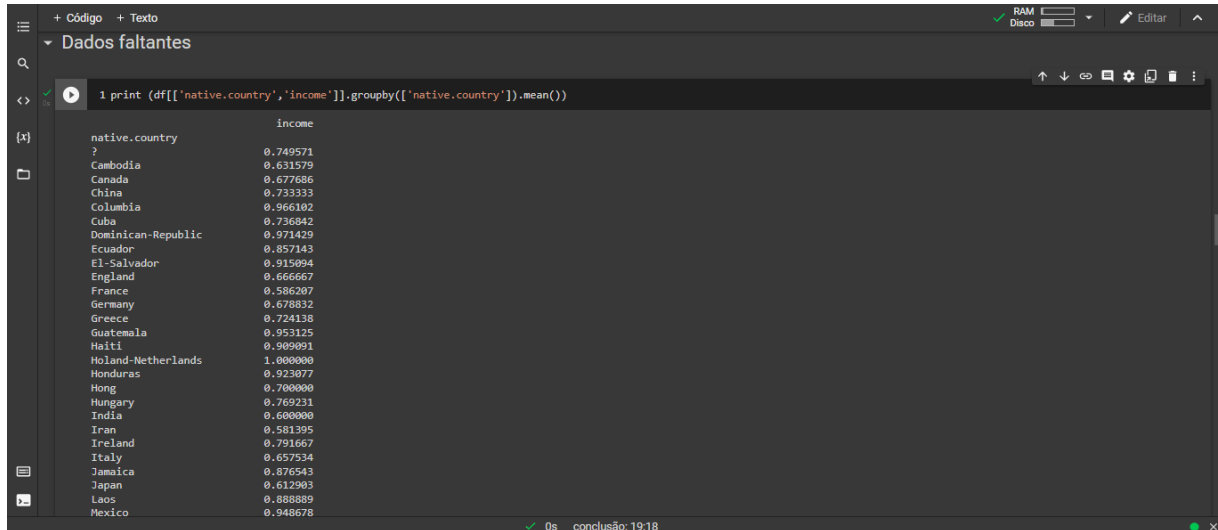
1 print("Sex classified binary \n", df.head(10), "\n")
Sex classified binary
age workclass fnlwgt ... hours.per.week native.country income
0 90 ? 77053 ... 40 United-States 1
1 82 Private 132870 ... 18 United-States 1
2 66 ? 186061 ... 40 United-States 1
3 54 Private 140359 ... 40 United-States 1
4 41 Private 284663 ... 40 United-States 1
5 34 Private 216864 ... 45 United-States 1
6 38 Private 150501 ... 40 United-States 1
7 74 State-gov 88638 ... 20 United-States 0
8 68 Federal-gov 422013 ... 40 United-States 1
9 41 Private 70037 ... 60 ? 0

[18 rows x 15 columns]
0s conclusão: 19:12

```

Neste passo, criei uma lista com a variável “sex” sexo, discretizando os gêneros para números binários, 0 para mulher e para 1 homem.

### Dados faltantes.

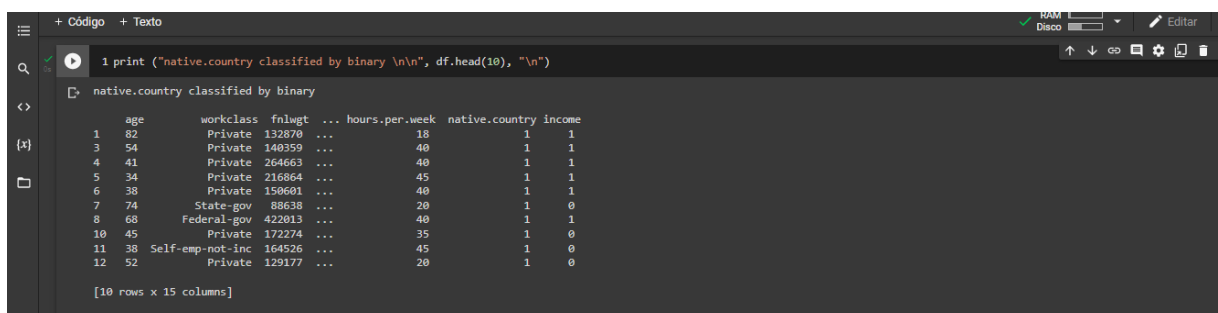


```
1 print (df[['native.country', 'income']].groupby(['native.country']).mean())
```

native.country	income
?	0.749571
Cambodia	0.631579
Canada	0.677686
China	0.733333
Columbia	0.966182
Cuba	0.736842
Dominican-Republic	0.971429
Ecuador	0.857143
EL-Salvador	0.915094
England	0.666667
France	0.586287
Germany	0.678832
Greece	0.724138
Guatemala	0.953125
Haiti	0.909091
Holand-Netherlands	1.000000
Honduras	0.923077
Hong	0.700000
Hungary	0.762231
India	0.600000
Iran	0.581395
Ireland	0.791667
Italy	0.657534
Jamaica	0.876543
Japan	0.612903
Laos	0.888889
Mexico	0.948678

Nesta parte do código temos uma função que nos traz alguns dados, das rendas de alguns países.

- Dados da variável “Native country” (País de origem).



```
1 print ("native.country classified by binary \n\n", df.head(10), "\n")
```

```
native.country classified by binary
```

	age	workclass	fnlwt	...	hours.per.week	native.country	income
1	82	Private	132870	...	18	1	1
3	54	Private	140359	...	40	1	1
4	41	Private	264663	...	40	1	1
5	34	Private	216864	...	45	1	1
6	38	Private	150601	...	40	1	1
7	74	State-gov	88638	...	20	1	0
8	68	Federal-gov	422813	...	40	1	1
10	45	Private	172274	...	35	1	0
11	38	Self-emp-not-inc	164526	...	45	1	0
12	52	Private	129177	...	20	1	0

[10 rows x 15 columns]

Nesta etapa, iremos discretizar os códigos para pessoas nativas e não nativas (United States) e classificadas em números binários, nativos ficam 1 e não nativos 0.

- Variável “Marital status” (Matrimônio).

```

[27] 1 print(df[['marital.status','relationship','income']].groupby\
      2      ([['marital.status','relationship']],.mean())

marital.status relationship      income
Couple      Husband      0.544331
            Not-in-family 0.714286
            Other-relative 0.857143
            Own-child      0.821429
            Wife           0.506401
Single      Not-in-family 0.893802
            Other-relative 0.976623
            Own-child      0.988818
            Unmarried      0.933686

[28] 1 print(df[['marital.status','relationship','income']].groupby\
      2      ([['relationship','marital.status']],.mean())

relationship marital.status      income
Husband      Couple      0.544331
Not-in-family Couple      0.714286
Other-relative Couple      0.857143
Own-child    Couple      0.821429
Wife         Single      0.976623
Unmarried    Couple      0.988818
Wife         Single      0.933686

```

```

[29] 1 marital_status_map={'Single':1,'Couple':0}
      2 df['marital.status'] = df['marital.status'].map(marital_status_map)

1 print("\nmarital.status classified by binary values \n\n", df.head(10), "\n")

marital.status classified by binary values
   age  workclass  fnlwt  ...  hours.per.week  native.country  income
1   82   Private  132878  ...         18         1         1
3   54   Private  140359  ...         40         1         1
4   41   Private  264663  ...         40         1         1
5   34   Private  216864  ...         45         1         1
6   38   Private  150601  ...         40         1         1
7   74  State-gov  88638  ...         20         1         0
8   68  Federal-gov 422013 ...         40         1         1
10  45   Private  172274  ...         35         1         0
11  38  Self-emp-not-inc 164926 ...         45         1         0
12  52   Private  129177  ...         20         1         0

[10 rows x 15 columns]

```

Na variável “Marital Status” ou “Matrimônio” nós agrupamos os dados de quem é divorciado, casado no civil, casado sem declaração, casado com cônjuge ausente, nunca casado, separado ou viúva, classificando em binário.

- Variável “Relationship” (Relacionamento).



```

+ Código + Texto
[31] 1 print(df[['relationship', 'income']].groupby(['relationship']).mean(), "\n")

relationship    income
Husband         0.544331
Not-in-family   0.893477
Other-relative  0.960630
Own-child       0.985670
Unmarried       0.933686
Wife            0.596401

[32] 1 relationship_map = {'Unmarried':0, 'Wife':1, 'Husband':2, 'Not-in-family':3,\
2 'Own-child':4, 'Other-relative':5}
3 df['relationship'] = df['relationship'].map(relationship_map)

1 print("relationship classified by values\n\n", df.head(10), "\n")

relationship classified by values
   age  workclass  fnlgt  ...  hours.per.week  native.country  income
1   82   Private  132870  ...             18                1         1
3   54   Private  148359  ...             40                1         1
4   41   Private  264663  ...             40                1         1
5   34   Private  216864  ...             45                1         1
6   38   Private  150601  ...             40                1         1
7   74  State.gov  88638  ...             20                1         0
8   68  Federal.gov 422013  ...             40                1         1
10  45   Private  172274  ...             35                1         0
11  38  Self-emp-not-inc 164526  ...             45                1         0
12  52   Private  129177  ...             20                1         0

```

Nesta variável, iremos também discretizar-lá para números, sendo classificadas como: ‘Solteiro/a’:0, ‘Esposa’:1, ‘Marido’:2, ‘Não em família’:3, ‘Uma criança’:4, ‘Outro relativo’:5.

- Variável “Race” (Raça).

```

+ Código + Texto
[34] 1 print(df[['race', 'income']].groupby('race').mean())

race    income
Amer-Indian-Eskimo  0.881119
Asian-Pac-Islander  0.722965
Black                0.870075
Other                0.969891
White               0.736282

[35] 1 race_map = {'White':0, 'Amer-Indian-Eskimo':1, 'Asian-Pac-Islander':2, 'Black':3,\
2 'Other':4}
3 df['race'] = df['race'].map(race_map)

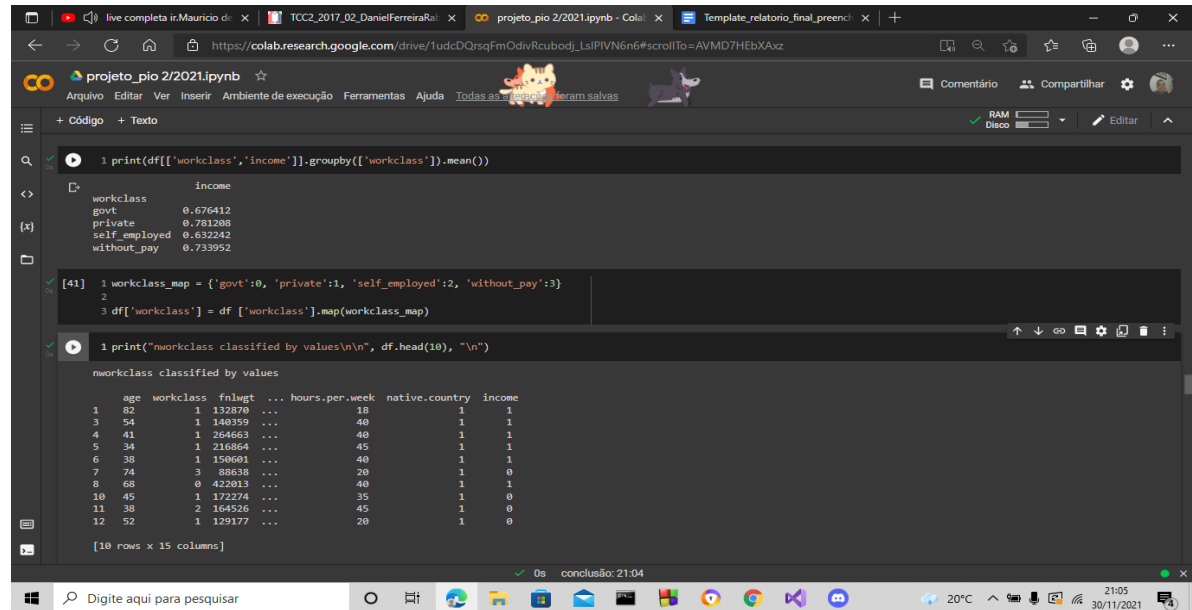
1 print("race classified by values\n\n", df.head(10), "\n")

race classified by values
   age  workclass  fnlgt  ...  hours.per.week  native.country  income
1   82   Private  132870  ...             18                1         1
3   54   Private  148359  ...             40                1         1
4   41   Private  264663  ...             40                1         1
5   34   Private  216864  ...             45                1         1
6   38   Private  150601  ...             40                1         1
7   74  State.gov  88638  ...             20                1         0
8   68  Federal.gov 422013  ...             40                1         1
10  45   Private  172274  ...             35                1         0
11  38  Self-emp-not-inc 164526  ...             45                1         0
12  52   Private  129177  ...             20                1         0

```

Nesta parte do código temos uma função que define um grupo, e dentro desta função temos uma variável “Race” (Raça) onde mapeia raça, também temos classificação através da idade de cada indivíduo.

## Variável “WorkClass” (Tipo de ocupação).



```
1 print(df[['workclass','income']].groupby(['workclass']).mean())
```

workclass	income
govt	0.676412
private	0.781208
self_employed	0.632242
without_pay	0.733952

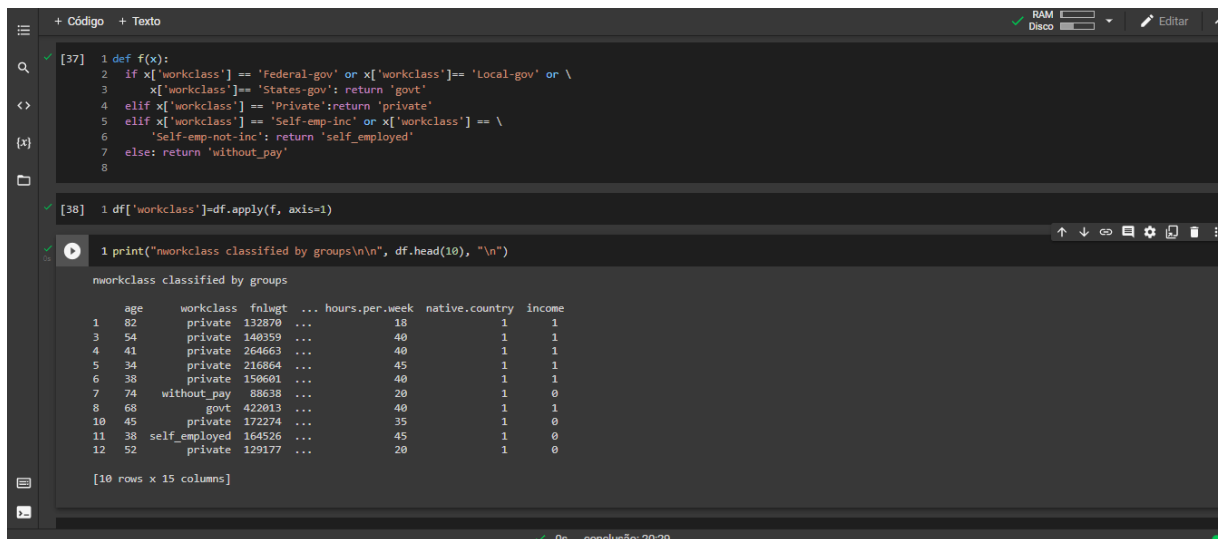
```
[41] 1 workclass_map = {'govt':0, 'private':1, 'self_employed':2, 'without_pay':3}
2
3 df['workclass'] = df['workclass'].map(workclass_map)
```

```
1 print("workclass classified by values\n\n", df.head(10), "\n")
```

workclass classified by values

	age	workclass	fnlwtg	...	hours.per.week	native.country	income
1	82	1	132870	...	18	1	1
3	54	1	140359	...	40	1	1
4	41	1	264663	...	40	1	1
5	34	1	216864	...	45	1	1
6	38	1	150601	...	40	1	1
7	74	3	88638	...	20	1	0
8	68	0	422013	...	40	1	1
10	45	1	172274	...	35	1	0
11	38	2	164526	...	45	1	0
12	52	1	129177	...	20	1	0

[10 rows x 15 columns]



```
[37] 1 def f(x):
2   if x['workclass'] == 'Federal-gov' or x['workclass']=='Local-gov' or \
3       x['workclass'] == 'States-gov': return 'govt'
4   elif x['workclass'] == 'Private':return 'private'
5   elif x['workclass'] == 'Self-emp-inc' or x['workclass'] == \
6       'Self-emp-not-inc': return 'self_employed'
7   else: return 'without_pay'
8
```

```
[38] 1 df['workclass']=df.apply(f, axis=1)
```

```
1 print("workclass classified by groups\n\n", df.head(10), "\n")
```

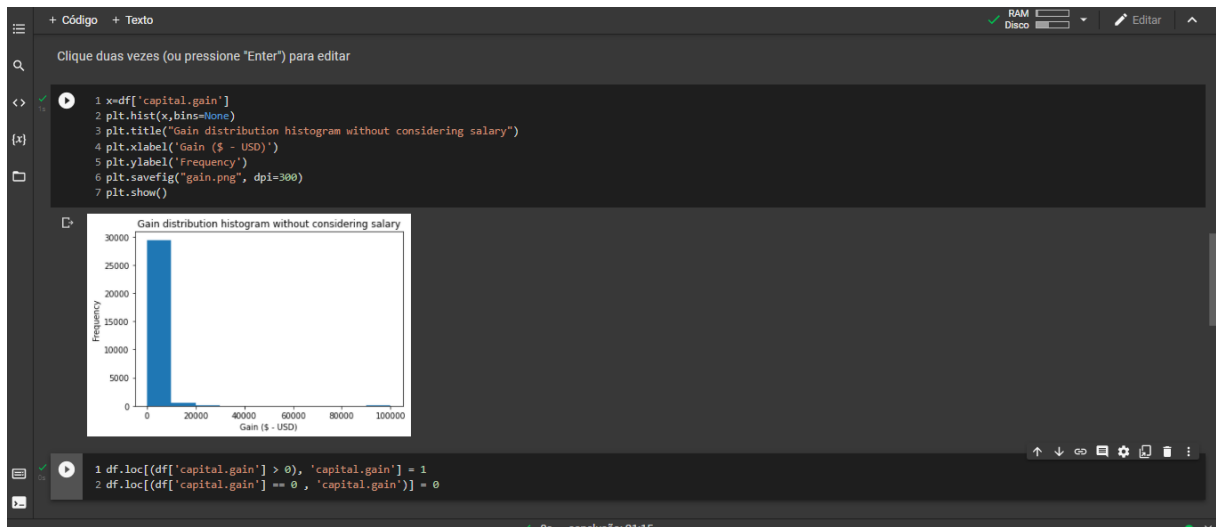
workclass classified by groups

	age	workclass	fnlwtg	...	hours.per.week	native.country	income
1	82	private	132870	...	18	1	1
3	54	private	140359	...	40	1	1
4	41	private	264663	...	40	1	1
5	34	private	216864	...	45	1	1
6	38	private	150601	...	40	1	1
7	74	without_pay	88638	...	20	1	0
8	68	govt	422013	...	40	1	1
10	45	private	172274	...	35	1	0
11	38	self_employed	164526	...	45	1	0
12	52	private	129177	...	20	1	0

[10 rows x 15 columns]

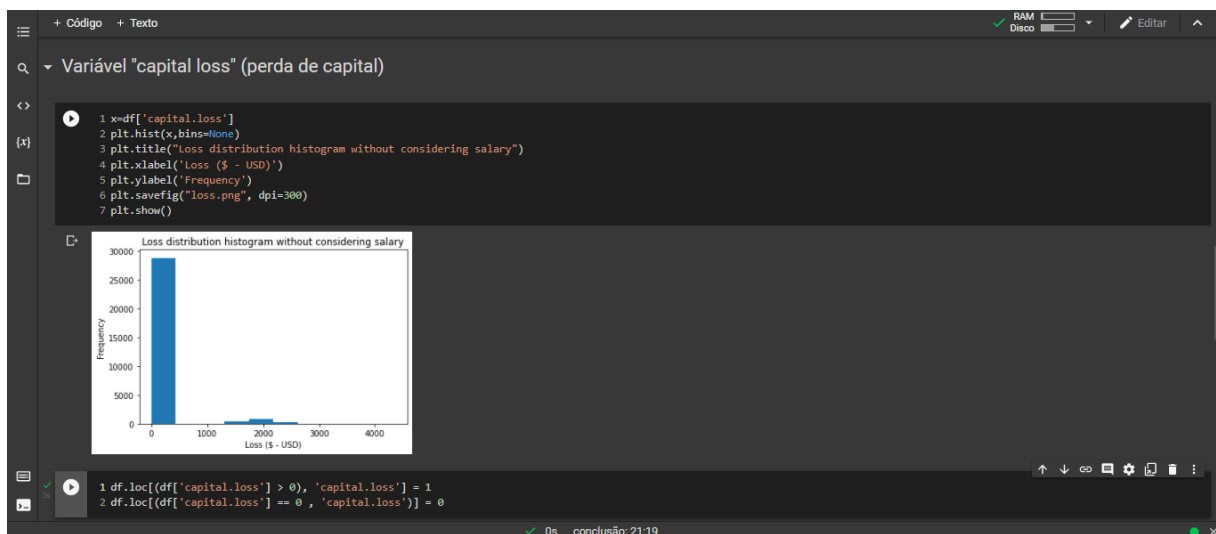
Na variável “work class” Tipo de ocupação fica a disposição a informação sobre os indivíduos que trabalham em empresas privadas, autônomas e quem não tem renda.

- Variável “Capital gain” (Ganho de capital).



Nesta parte do código temos uma variável com função de mapear o ganho capital, criando um histograma e logo depois fazendo a conversão binária.

- Variável “Capital loss” (Perda de Capital).



Esta parte do código temos uma variável que recebe uma função, esta função tem nome perda de capital, temos um gráfico que mostra perda sem considerar o salário.

- Variável “Education” (Nível Educacional).

```
Variável "education" (nível educacional)
Variável redundante em relação à education.num
1 print(df[['education', 'income']].groupby(['education']).mean())
```

education	income
10th	0.928049
11th	0.943702
12th	0.923077
1st-4th	0.960265
5th-6th	0.958333
7th-8th	0.937163
9th	0.945855
Assoc-acdm	0.746032
Assoc-voc	0.736802
Bachelors	0.578509
Doctorate	0.253333
HS-grad	0.835671
Masters	0.435771
Preschool	1.000000
Prof-school	0.250923
Some-college	0.799940

Nessa variável, recebemos os dados de pessoas sobre o seu nível de educação, ensino fundamental/médio, faculdade, mestrado etc.

- Variável “Occupation” (Ocupação).

```
+ Código + Texto
RAM
Disco
Editor
Variável "occupation" (ocupação)
Variável muito pulverizada, que a torna pouco explicativa
1 print(df[['occupation', 'income']].groupby(['occupation']).mean())
```

occupation	income
Adm-clerical	0.866165
Armed-Forces	0.888889
Craft-repair	0.774690
Exec-managerial	0.514780
Farming-fishing	0.883721
Handlers-cleaners	0.938519
Machine-op-inspct	0.875381
Other-service	0.958904
Priv-house-serv	0.990007
Prof-specialty	0.551511
Protective-serv	0.673913
Sales	0.729353
Tech-support	0.695175
Transport-moving	0.797074

Nesta variável se apresenta as diferentes profissões, como administrador, operador de telemarketing, vendedor etc.

- Remoção.

```
+ Código + Texto
1 print("\neducation and occupation are removed because are not significant " \
2       "explain variables \n\n", df.head(10), "\n")
```

```
education and occupation are removed because are not significant explain variables
```

	age	workclass	fnlwt	...	hours.per.week	native.country	income
1	82	1	132879	...	18	1	1
3	54	1	140359	...	40	1	1
4	41	1	264663	...	40	1	1
5	34	1	216864	...	45	1	1
6	38	1	150601	...	40	1	1
7	74	3	88638	...	20	1	0
8	68	0	422813	...	40	1	1
10	45	1	172274	...	35	1	0
11	38	2	164526	...	45	1	0
12	52	1	129177	...	20	1	0

[10 rows x 13 columns]

Neste quesito, foi feita a remoção de dados como de educação e ocupação porque não são variáveis explicações significativas.

- Variável “Hours per week” (Horas trabalhadas por semana).

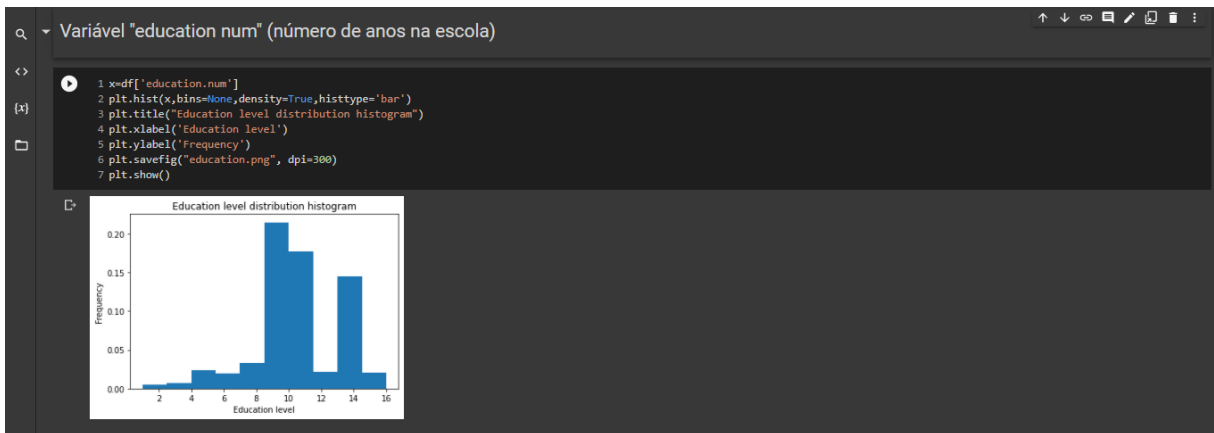
```
Variável "hours per week" (horas trabalhadas por semana)
```

```
1 x=df['hours_per_week']
2 plt.hist(x,bins=None,density=True,histtype='bar')
3 plt.title("Hour worked distribution histogram")
4 plt.xlabel("Hours worked per week (Total hours)")
5 plt.ylabel("Frequency")
6 plt.savefig("hours.png", dpi=300)
7 plt.show()
```



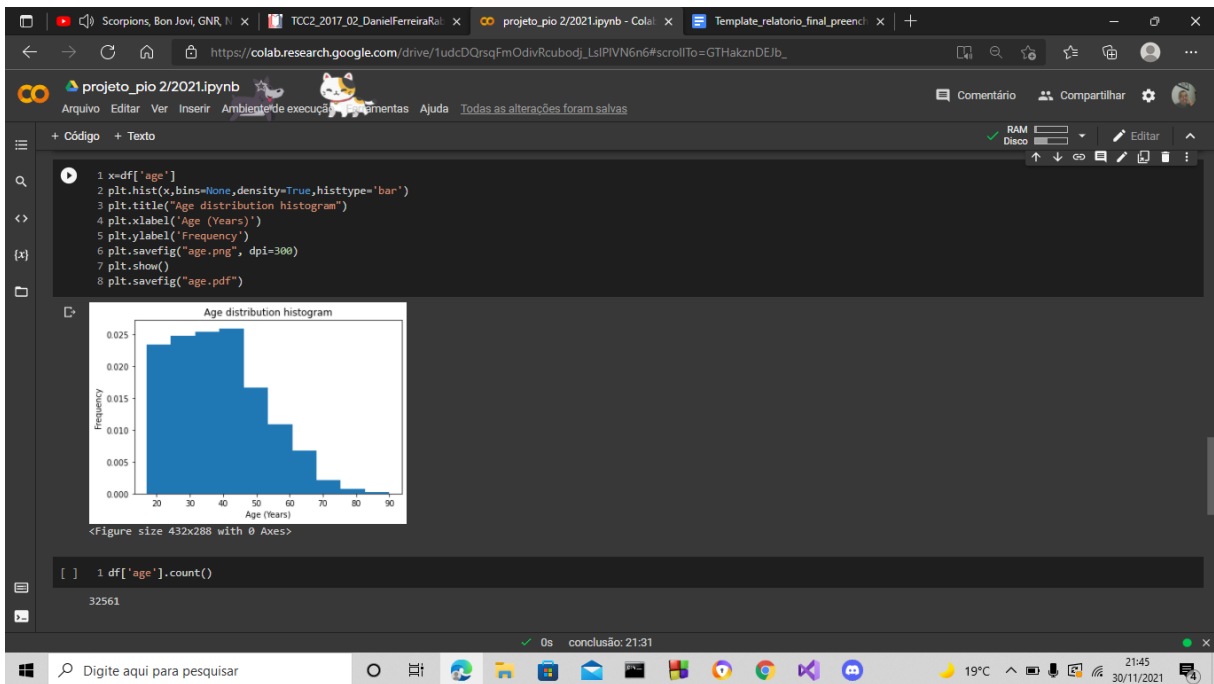
Neste, temos essa variável com a função de mapear as horas trabalhadas por semana, distribuindo e criando um histograma com as informações dadas.

- Variável “Education num” (Número de anos na escola).



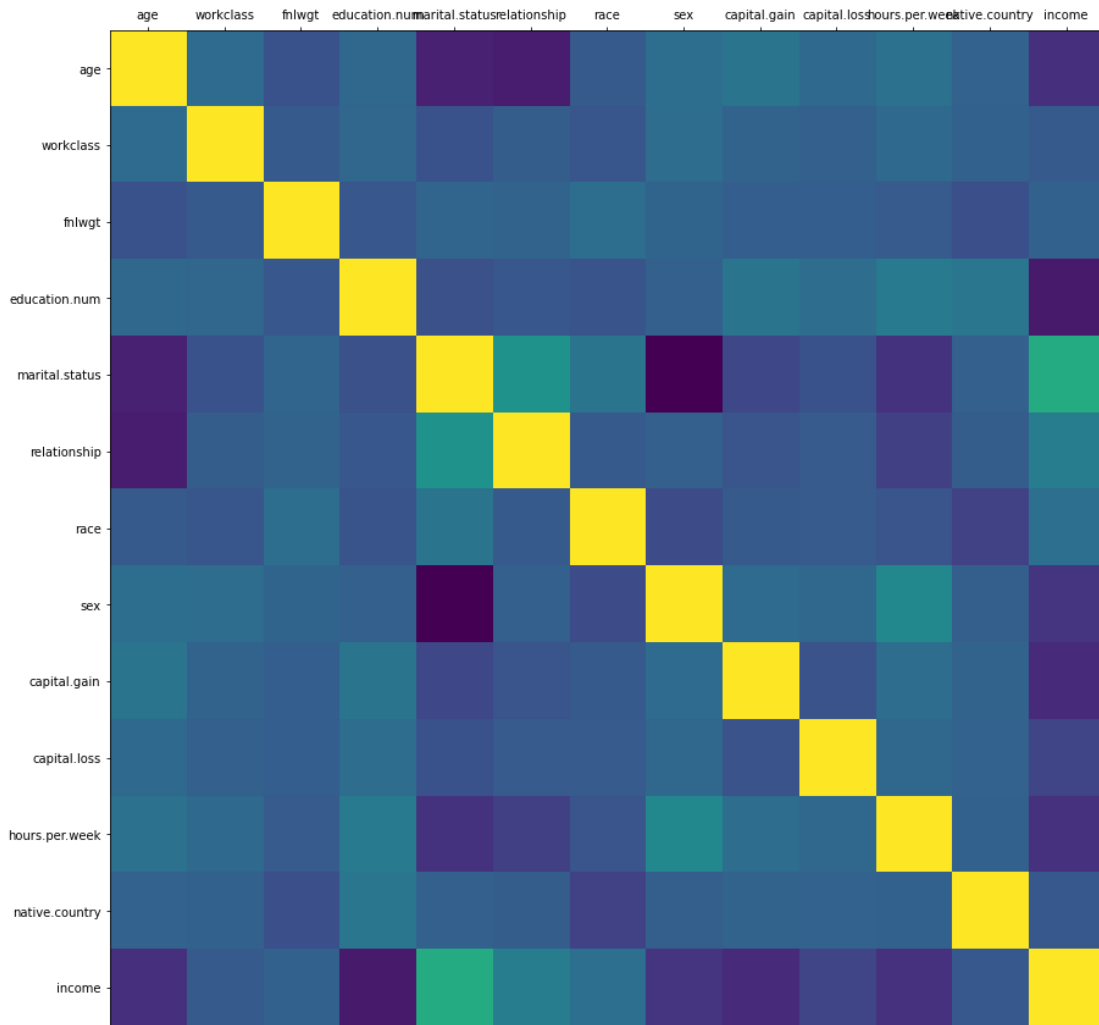
Nesta variável temos a função de mapear o tempo de educação, criando um histograma com os dados.

- Variável “Age” (Idade).



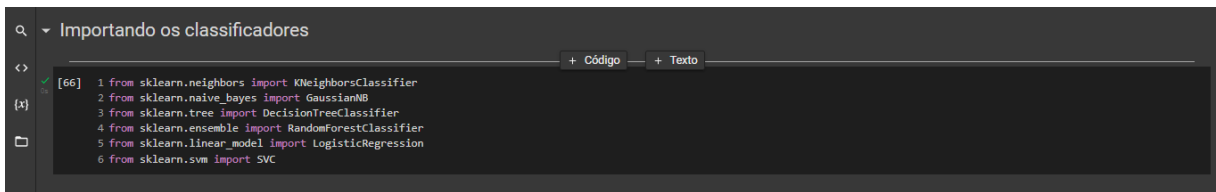
Nesta variável temos a função de separação e redistribuição das idades, criando um histograma com os dados, e logo depois fazendo a conta dos registros.

● Analisando Correlação.



Esta análise de correlação verifica diversos dados do nosso banco

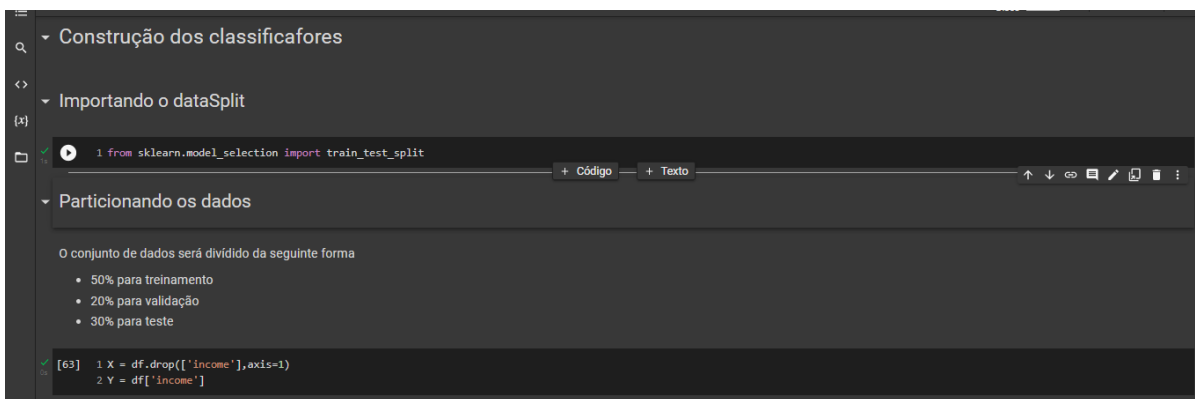
- Construção dos classificadores.



```
[66] 1 from sklearn.neighbors import KNeighborsClassifier
      2 from sklearn.naive_bayes import GaussianNB
      3 from sklearn.tree import DecisionTreeClassifier
      4 from sklearn.ensemble import RandomForestClassifier
      5 from sklearn.linear_model import LogisticRegression
      6 from sklearn.svm import SVC
```

Nesta parte do código temos importação de alguns agrupamentos.

- Importando o Data Split.

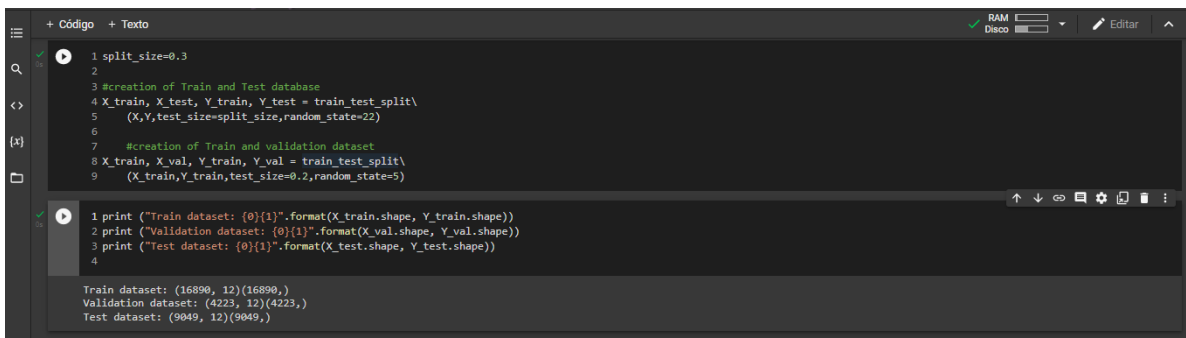


```
1 from sklearn.model_selection import train_test_split
```

O conjunto de dados será dividido da seguinte forma

- 50% para treinamento
- 20% para validação
- 30% para teste

```
[63] 1 X = df.drop(['income'],axis=1)
      2 Y = df['income']
```



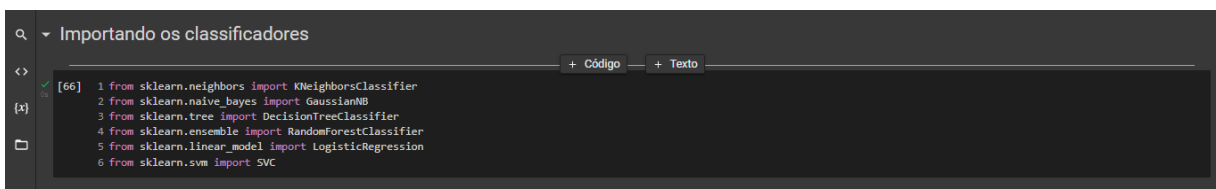
```
1 split_size=0.3
2
3 #creation of Train and Test database
4 X_train, X_test, Y_train, Y_test = train_test_split\
5 (X,Y,test_size=split_size,random_state=22)
6
7 #creation of Train and validation dataset
8 X_train, X_val, Y_train, Y_val = train_test_split\
9 (X_train,Y_train,test_size=0.2,random_state=5)
```

```
1 print ("Train dataset: {}".format(X_train.shape, Y_train.shape))
2 print ("Validation dataset: {}".format(X_val.shape, Y_val.shape))
3 print ("Test dataset: {}".format(X_test.shape, Y_test.shape))
4
```

Train dataset: (16890, 12)(16890,)  
Validation dataset: (4223, 12)(4223,)  
Test dataset: (9049, 12)(9049,)

Função que converte e separa o conjunto dos dados para treinamento, validação e testes, usando estes parâmetros, que são aplicados em todas as variáveis para prever o “income”.

- Importando os classificadores.



```
[66] 1 from sklearn.neighbors import KNeighborsClassifier
      2 from sklearn.naive_bayes import GaussianNB
      3 from sklearn.tree import DecisionTreeClassifier
      4 from sklearn.ensemble import RandomForestClassifier
      5 from sklearn.linear_model import LogisticRegression
      6 from sklearn.svm import SVC
```

Importando métodos classificadores, outros tipos de biblioteca que classificam outros tipos de dados.



- Treinamento.

```
Treinamento

[ ] models = []

names = ['Nearest Neighbors', 'Naive Bayes', 'Decision Tree', \
        'Random Forest', 'Logistic Regression', 'SVC']

[ ] models.append((KNeighborsClassifier(n_neighbors = 50)))
models.append((GaussianNB()))
models.append((DecisionTreeClassifier()))
models.append((RandomForestClassifier(n_estimators=100)))
models.append((LogisticRegression()))
models.append((SVC()))

[ ] print(models)

[KNeighborsClassifier(n_neighbors=50), GaussianNB(), DecisionTreeClassifier(), RandomForestClassifier(), LogisticRegression(), SVC()]
```

Criação e adição dos métodos a uma lista

- Importando os métodos de treinamento.

```
Importando os metodos de treinamento

[70] 1 from sklearn import model_selection
     2 from sklearn.metrics import accuracy_score

[77] 1 kfold = model_selection.KFold(n_splits=5, random_state=7)

2 for i in range(0, len(models)):
3     cv_result = model_selection.cross_val_score(
4         (models[i], X_train, Y_train, cv=kfold, scoring='accuracy')
5         score=models[i].fit(X_train, Y_train)
6         prediction = models[i].predict(X_val)
7         acc_score = accuracy_score(Y_val, prediction)
8         print ('-'*40)
9         print ('{0}: {1}'.format(names[i], acc_score))
```

Este código não funcionou, tentamos adaptar, mas não conseguimos, mas este código se funcionasse iria mostra o nível de exatidão.

- Importando as bibliotecas de avaliação.

```
Importando as bibliotecas de avaliação

1 from sklearn.metrics import classification_report
2 from sklearn.metrics import confusion_matrix
3
4 from sklearn.model_selection import GridSearchCV
```

Estas bibliotecas são para classificar e apontar as métricas utilizadas.

### **3.2. Solução Final**

Descrição detalhada, com imagens, de como se deu o processo de construção da solução final apresentada pelo grupo. Espera-se que o grupo demonstre quais foram as melhorias realizadas na solução final, a partir dos feedbacks coletados junto à comunidade ou local onde o projeto foi desenvolvido.

## **4. CONSIDERAÇÕES FINAIS**

Deve-se retomar os objetivos e o contexto em que o Projeto Integrador foi desenvolvido e apresentar:

Retomada dos resultados à luz das referências estudadas;

**R:** As referências para retomada dos resultados foram através do Vídeo que foi nos dado como exemplo para construção do trabalho em Si. Usamos o stackoverflow para conseguir resolver alguns erros que achamos ao decorrer do trabalho, e então dar a retomada aos resultados esperados.

Contribuições e limitações do trabalho realizado;

**R:** Conseguimos a contribuição do banco de dados “adult”, Através do vídeo, a limitações que tivemos foi o pouco tempo que tivemos para trabalhar e tivemos um pouco de dificuldade na execução dos códigos, mas na medida do possível conseguimos fazer um trabalho coerente ao que foi passado.

## REFERÊNCIAS

- ABNT – Associação Brasileira de Normas Técnicas. **NBR 14724**: Informação e documentação. Trabalhos Acadêmicos - Apresentação. Rio de Janeiro: ABNT, 2002.
- BOYER, C. B.; UTA, C. M. **História da Matemática** [Trad. Helena Castro]. 3 ed. São Paulo: Blucher, 2012.
- D'AMBRÓSIO, U. **Educação Matemática**: da teoria à prática. 23. ed. Campinas: Papirus, 2012.
- KUBO, OM.; BOTOMÉ, S. **Ensino e aprendizagem**: uma interação entre dois processos comportamentais. *Interação*, v.5, p.123-32, 2001.
- HART-DAVIS, A. **O Livro da Ciência**. 2. ed. São Paulo: Globo, 2016.
- PILETTI, C. **Didática geral**. São Paulo: Ática, 1995.
- RIBEIRO, J.L. , P. Áreas e Proporções nas Superquadradas de Brasília Usando o Google Maps. **Revista do Professor de Matemática**. Rio de Janeiro, n. 92, p. 12-15, jan-abr. 2017.
- SEVERINO, A. J. **Metodologia do trabalho científico**. 22. ed. rev. e ampl. São Paulo: Cortez, 2002.

O trabalho deverá ser redigido conforme recomendações das Diretrizes para confecção de teses e dissertações da Universidade de São Paulo (USP), disponíveis em: [http://www.teses.usp.br/index.php?option=com\\_content&view=article&id=52&Itemid=67](http://www.teses.usp.br/index.php?option=com_content&view=article&id=52&Itemid=67)>. Acesso em 24 jun.2021.