Curso de Ciência de Dados Aula2

Para receber por e-mail o(s) arquivo(s) utilizados na aula, preencha:

Seu melhor e-mail

Enviar

Introdução ao Pandas

Se você já tem conhecimento sobre o **pandas** essa aula vai ser mais uma revisão, caso contrário vem comigo que eu vou te mostrar alguns comandos e funcionalidades dessa biblioteca.

O pandas possui alta performance, é fácil de utilizar para trabalhar com dados, tem muita semelhança com o Excel, então se já sabe mexer no Excel fica mais fácil entender alguns processos.

O DataFrame é utilizado em várias outras bibliotecas de Ciência de Dados. O pandas vai desde a importação da base de dados, passando por toda a análise exploratória até a exibição de gráficos.

Caso tenha alguma dúvida de como baixar ou como utilizar algum recurso, basta acessar a documentação dessa biblioteca no link abaixo:

https://pandas.pydata.org/

Vamos iniciar com a importação dessa biblioteca.

```
# Importando o pandas
import pandas as pd
```

Importando a biblioteca pandas

Aqui você pode utilizar apenas o **import pandas**, só que quando colocamos o “as pd” isso facilita na hora de escrever o seu código.

Pois sempre que for utilizar algum comando dessa biblioteca você vai escrever **pandas.comando**, e isso não é interessante, então esse “as pd” troca o nome pandas apenas por “pd”.

Então diminui a quantidade de informação que você escreve, então passa a escrever **pd.comando**, que fica muito mais rápido e fácil.

pd.read_csv("arquivo_excel.xlsx")

- O arquivo que vamos usar é o `titanic_train.csv`

```
# Importando a base em csv
base = pd.read_csv('titanic_train.csv')
```

Lendo a base de dados em CSV

Depois de importar a base de dados já podemos começar a utilizá-la e o primeiro comando que vamos aprender é o **pd.read** que é para ler um arquivo, nesse caso estamos utilizando o **pd.read_csv**, pois a extensão do nosso arquivo é csv.

Além disso já estamos lendo essas informações e armazenando na variável base, com isso podemos utilizar essa base de dados para visualizar os dados e fazer os tratamentos.

Não adianta apenas ler o arquivo se não vamos salvar de alguma maneira para podermos utilizar essas informações.

E também visualizar todas as informações sobre essa base

```
# Visualizando as 5 primeiras linhas
base.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
# Visualizando as 5 últimas linhas
base.tail()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

Visualizando os dados da base

Aqui temos dois comandos que nos permitem visualizar as 5 primeiras linhas (**base.head**) e visualizar as últimas 5 linhas (**base.tail**).

Esses comandos são interessantes para que você consiga visualizar como está a estrutura dos seus dados, como eles estão organizados, sem que tenha que trazer toda a base de dados.

Além disso, também podemos ver as informações da base

```
# Mostrando somente o tipo dos dados
base.dtypes
```

```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

Verificando o tipo dos dados

Você pode visualizar qual é o tipo de informação que temos em cada uma das nossas colunas, o que facilita na hora de fazer as operações, pois já vamos saber se é possível ou não fazer essas operações.

É possível que você tenha números em uma coluna, mas o formato pode ser de texto, então você já sabe que se for fazer algum cálculo não vai funcionar, pois não conseguimos fazer cálculos com textos.

```
# Mostrando o tipo de dados e os valores vazios
base.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

Tipos de dados e valores vazios

Outra verificação bem importante além de visualizar o tipo dos dados é verificar os valores vazios, pois isso pode ser um problema.

Muitas das vezes precisamos tratar esses valores vazios antes de começar com a análise de dados, então é importante verificar que tudo está certo com a base de dados antes de prosseguir.

Para verificar melhor a quantidade de valores vazios nós podemos utilizar o **base.isnull().sum()**, assim vamos fazer a soma de todos os valores vazios para que possamos tratar da melhor forma.

Agora nós vamos aos principais conceitos estatísticos que vão te ajudar na análise dos seus dados.

```
    'X': [1,2,3,4,5,6,7,8,9,10,11]
}

dados = pd.DataFrame(dados)
```

```
# Mostrando a média
dados.mean()
```

```
X    6.0
dtype: float64
```

```
# Mostrando a contagem de registros
dados.count()
```

```
X    11
dtype: int64
```

```
# Mediana
dados.median()
```

```
X    6.0
dtype: float64
```

```
# Desvio Padrão
dados.std()
```

```
X    3.316625
dtype: float64
```

```
# Trazendo todo o resumo estatístico utilizando o describe
dados.describe()
```

	X
count	11.000000
mean	6.000000
std	3.316625
min	1.000000
25%	3.500000
50%	6.000000
75%	8.500000
max	11.000000

Conceitos estatísticos

Aqui temos a média, contagem, mediana, desvio padrão e o resumo estatístico de uma base de dados que vai de 1 a 11.

Agora nós podemos utilizar esse resumo estatístico dentro da nossa base de dados que tínhamos inicialmente.

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Resumo estatístico da base de dados

Podemos também trazer as informações de uma única coluna para trabalhar somente com esses dados.

Buscando por uma coluna:

```
# Podemos usar o nome da coluna entre aspas
base['Survived']
```

```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

```
# Ou usar o ponto
base.Survived
```

```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

Buscando por colunas dentro da base de dados

Podemos fazer a contagem de cada um dos termos dentro de uma coluna utilizando o **value_counts**.

```
0    549
1    342
Name: Survived, dtype: int64
```

```
# Selecionando um conjunto de colunas
base[['Survived','Sex','Age']]
```

	Survived	Sex	Age
0	0	male	22.0
1	1	female	38.0
2	1	female	26.0
3	1	female	35.0
4	0	male	35.0
...	...	...	...
886	0	male	27.0
887	1	female	19.0
888	0	female	NaN
889	1	male	26.0
890	0	male	32.0

891 rows × 3 columns

Contando valores da coluna e selecionando um conjunto de colunas

Além disso você pode selecionar mais de uma coluna de uma vez, para que possa separá-las e trabalhar apenas com o que precisa.

Então para análises mais detalhadas nós podemos fazer alguns filtros na nossa base de dados.

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
# Verificando clientes que pagaram mais de 500 libras
base[base.Fare > 500]
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
258	259	1	1	Ward, Miss. Anna	female	35.0	0	0	PC 17755	512.3292	NaN	C
679	680	1	1	Cardeza, Mr. Thomas Drake Martinez	male	36.0	0	1	PC 17755	512.3292	B51 B53 B55	C
737	738	1	1	Lesurer, Mr. Gustave J	male	35.0	0	0	PC 17755	512.3292	B101	C

```
# Verificando se tiveram clientes que pagaram menos de 5 libras
base[base.Fare < 5]
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
179	180	0	3	Leonard, Mr. Lionel	male	36.0	0	0	LINE	0.0000	NaN	S
263	264	0	1	Harrison, Mr. William	male	40.0	0	0	112059	0.0000	B94	S
271	272	1	3	Tornquist, Mr. William Henry	male	25.0	0	0	LINE	0.0000	NaN	S
277	278	0	2	Parkes, Mr. Francis "Frank"	male	NaN	0	0	239853	0.0000	NaN	S
302	303	0	3	Johnson, Mr. William Cahoone Jr	male	19.0	0	0	LINE	0.0000	NaN	S
378	379	0	3	Betros, Mr. Tannous	male	20.0	0	0	2648	4.0125	NaN	C
413	414	0	2	Cunningham, Mr. Alfred Fleming	male	NaN	0	0	239853	0.0000	NaN	S
466	467	0	2	Campbell, Mr. William	male	NaN	0	0	239853	0.0000	NaN	S
481	482	0	2	Frost, Mr. Anthony Wood "Archie"	male	NaN	0	0	239854	0.0000	NaN	S
597	598	0	3	Johnson, Mr. Alfred	male	49.0	0	0	LINE	0.0000	NaN	S
633	634	0	1	Parr, Mr. William Henry Marsh	male	NaN	0	0	112052	0.0000	NaN	S
674	675	0	2	Watson, Mr. Ennis Hastings	male	NaN	0	0	239856	0.0000	NaN	S
732	733	0	2	Knight, Mr. Robert J	male	NaN	0	0	239855	0.0000	NaN	S
806	807	0	1	Andrews, Mr. Thomas Jr	male	39.0	0	0	112050	0.0000	A36	S
815	816	0	1	Fry, Mr. Richard	male	NaN	0	0	112058	0.0000	B102	S
822	823	0	1	Reuchlin, Jonkheer. John George	male	38.0	0	0	19972	0.0000	NaN	S

Filtrando informações na base de dados

Aqui estamos filtrando a coluna **Fare** (que seria um valor de taxa), então no primeiro exemplo estamos filtrando apenas o valores **maiores do que 500** e no segundo exemplo valores **menores do que 5**.

Além de fazer alguns filtros de forma separada, nós vamos poder utilizar **E** e **OU** para fazer isso, ou seja, podemos juntar filtros para trazer informações mais detalhadas.

```
base[(base.Parch > 1) & (base.SibSp > 1)].head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.000	C23 C25 C27	S
59	60	0	3	Goodwin, Master. William Frederick	male	11.0	5	2	CA 2144	46.900	NaN	S
63	64	0	3	Skoog, Master. Harald	male	4.0	3	2	347088	27.900	NaN	S
68	69	1	3	Andersson, Miss. Erna Alexandra	female	17.0	4	2	3101281	7.925	NaN	S
71	72	0	3	Goodwin, Miss. Lillian Amy	female	16.0	5	2	CA 2144	46.900	NaN	S

Uma forma muito útil de fazer seleção de dados é usando o .loc() ou o .iloc()

```
# O .loc() permite fazer a busca da mesma forma acima
base.loc[(base.Parch > 1) & (base.SibSp > 1)].head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.000	C23 C25 C27	S
59	60	0	3	Goodwin, Master. William Frederick	male	11.0	5	2	CA 2144	46.900	NaN	S
63	64	0	3	Skoog, Master. Harald	male	4.0	3	2	347088	27.900	NaN	S
68	69	1	3	Andersson, Miss. Erna Alexandra	female	17.0	4	2	3101281	7.925	NaN	S
71	72	0	3	Goodwin, Miss. Lillian Amy	female	16.0	5	2	CA 2144	46.900	NaN	S

```
# Ele também permite usar argumentos lógicos
base.loc[(base.Parch > 1) | (base.SibSp > 1)].head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
13	14	0	3	Andersson, Mr. Anders Johan	male	39.0	1	5	347082	31.2750	NaN	S
16	17	0	3	Rice, Master. Eugene	male	2.0	4	1	382652	29.1250	NaN	Q
24	25	0	3	Palsson, Miss. Torborg Danira	female	8.0	3	1	349909	21.0750	NaN	S

```
# Ele permite filtrar colunas de forma muito prática
base.loc[(base.Parch > 1) | (base.SibSp > 1),['PassengerId','Parch','SibSp']].head()
```

	PassengerId	Parch	SibSp
7	8	1	3
8	9	2	0
13	14	5	1
16	17	1	4
24	25	1	3

Filtros com E e OU

No primeiro exemplo estamos filtrando as informações em que a coluna **Parch** é maior do que 1 e ao mesmo tempo (aplicando **E** com o símbolo de **&**) filtrando as informações onde a coluna **SubSp** é maior do que 1.

Isso quer dizer que estamos querendo que ambas as colunas só tragam as informações quando as duas condições forem satisfeitas.

Além disso podemos fazer um filtro com o **OU** que é representado pela barra (**|**), então nesse caso apenas uma das condições precisa ser satisfeita, ou o **Parch** ser maior do que 1, ou o **SibSp** ser maior do que 1.

Nesse primeiro exemplo fizemos o filtro normal e com o comando **.loc()**, já para o próximo exemplo podemos utilizar o **.iloc()** onde ele vai utilizar o índice para fazer esse filtro.

O índice é essa primeira coluna que fica em negrito para marcar a linha de cada registro.

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
30	31	0	1	Uruchurtu, Don. Manuel E	male	40.0	0	0	PC 17601	27.7208	NaN	C
31	32	1	1	Spencer, Mrs. William Augustus (Marie Eugenie)	female	NaN	1	0	PC 17569	146.5208	B78	C
32	33	1	3	Glynn, Miss. Mary Agatha	female	NaN	0	0	335677	7.7500	NaN	Q
33	34	0	2	Wheadon, Mr. Edward H	male	66.0	0	0	C.A. 24579	10.5000	NaN	S
34	35	0	1	Meyer, Mr. Edgar Joseph	male	28.0	1	0	PC 17604	82.1708	NaN	C
35	36	0	1	Holverson, Mr. Alexander Oskar	male	42.0	1	0	113789	52.0000	NaN	S
36	37	1	3	Mamee, Mr. Hanna	male	NaN	0	0	2677	7.2292	NaN	C
37	38	0	3	Cann, Mr. Ernest Charles	male	21.0	0	0	A./5. 2152	8.0500	NaN	S
38	39	0	3	Vander Planke, Miss. Augusta Maria	female	18.0	2	0	345764	18.0000	NaN	S
39	40	1	3	Nicola-Yarred, Miss. Jamila	female	14.0	1	0	2651	11.2417	NaN	C

```
# Também posso usar para buscar apenas colunas específicas
base.iloc[30:40,3:6]
```

	Name	Sex	Age
30	Uruchurtu, Don. Manuel E	male	40.0
31	Spencer, Mrs. William Augustus (Marie Eugenie)	female	NaN
32	Glynn, Miss. Mary Agatha	female	NaN
33	Wheadon, Mr. Edward H	male	66.0
34	Meyer, Mr. Edgar Joseph	male	28.0
35	Holverson, Mr. Alexander Oskar	male	42.0
36	Mamee, Mr. Hanna	male	NaN
37	Cann, Mr. Ernest Charles	male	21.0
38	Vander Planke, Miss. Augusta Maria	female	18.0
39	Nicola-Yarred, Miss. Jamila	female	14.0

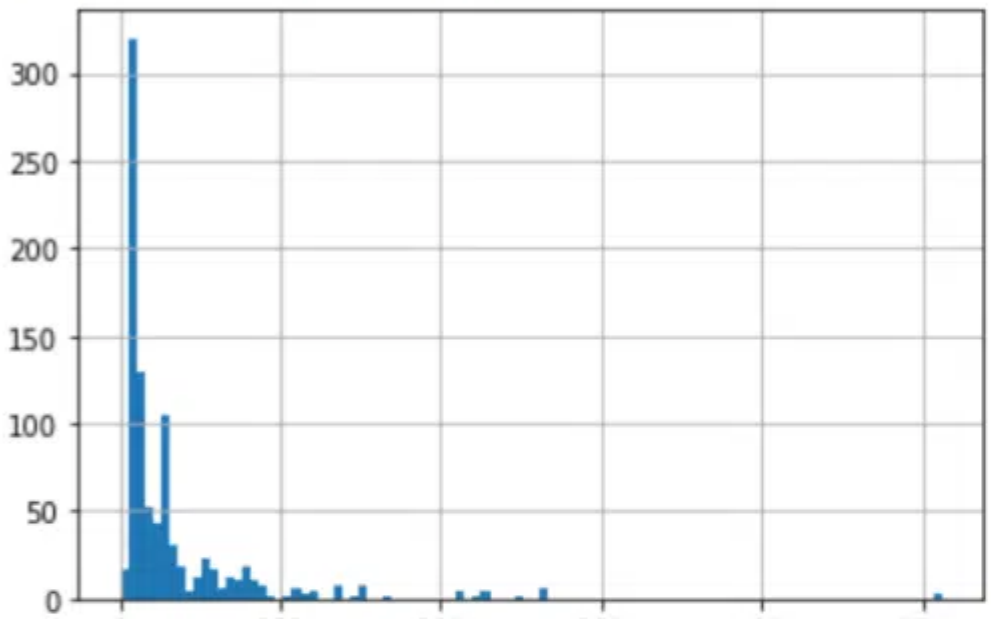
Filtrando informações através do índice

Então podemos filtrar apenas por linhas específicas, ou linhas e colunas ao mesmo tempo.

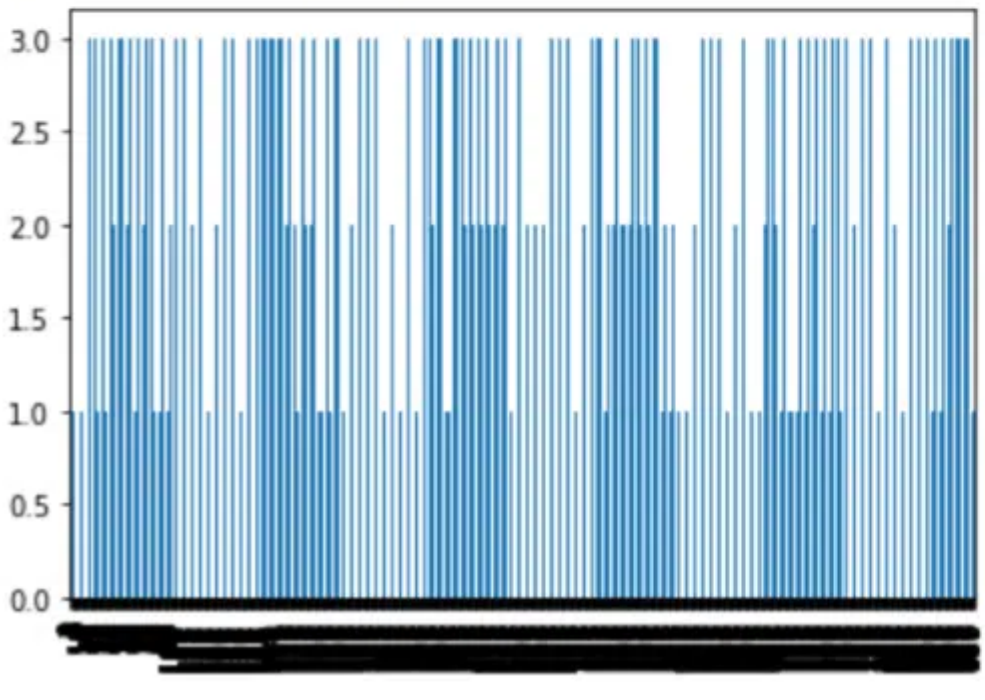
Para finalizar essas análises nada melhor do que um gráfico para representar esses dados de uma forma mais visual não é mesmo?

```
import matplotlib.pyplot as plt
```

```
# É possível fazer um histograma simples
base.Fare.hist(bins=100);
```

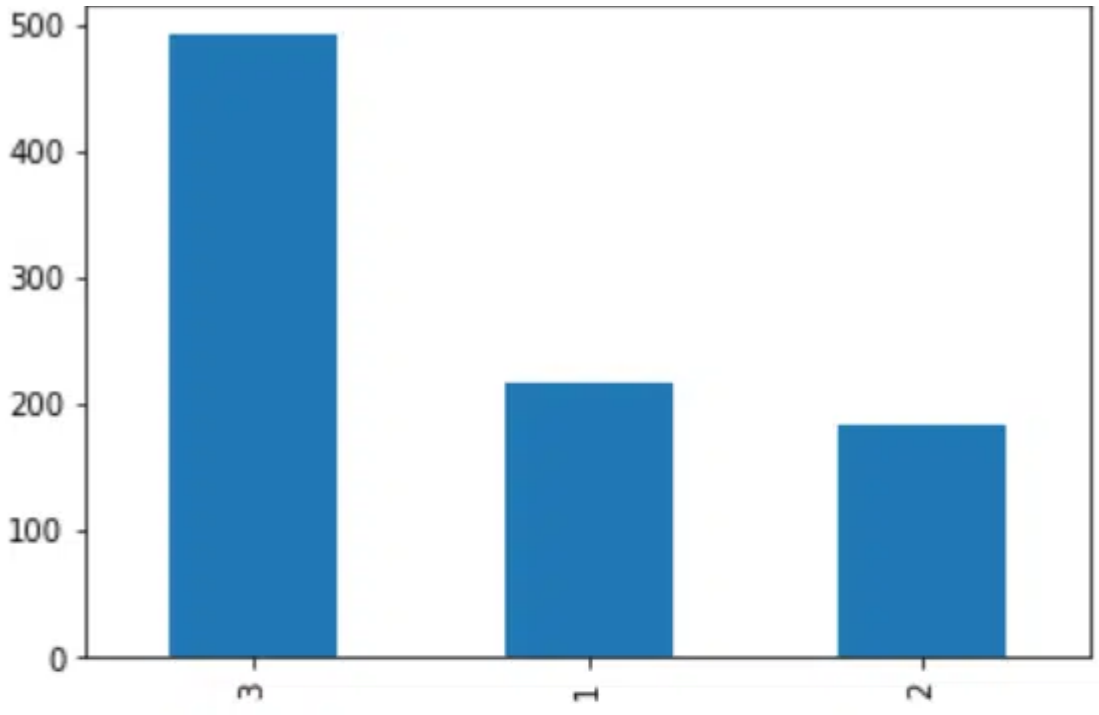



```
# Gráficos de barras
base.Pclass.plot.bar();
```



Histograma e gráfico de barras

Aqui temos um histograma e um gráfico de barras representando a coluna de Pclass, então você pode fazer uma análise melhor quando tem gráficos até para verificar se tem algum padrão, se está crescendo...




```
# E até gráficos mais complexos como o de densidade
base['Fare'].plot.kde();
```

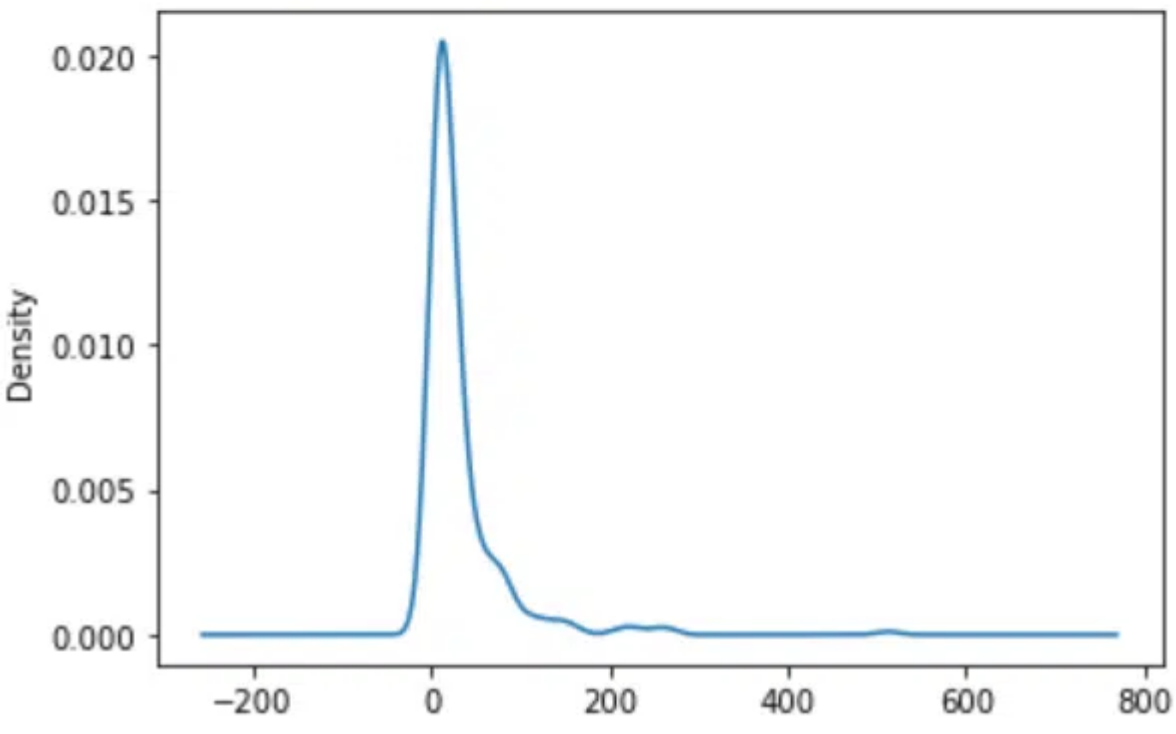


Gráfico de barras e gráfico de densidade

Aqui temos outro gráfico de barra, só que dessa vez com a soma dos valores, veja que fica muito mais fácil de entender essa visualização.

Nós temos apenas 3 informações nessa coluna (1, 2 e 3) e nesse gráfico estamos vendo a quantidade de cada um deles.

E por fim temos um gráfico mais complexo que é um gráfico de densidade, mas você tem diversos tipos de gráficos que pode criar, até porque pode utilizar outros gráficos para representar melhor seus dados.

Na documentação você vai conseguir visualizar diversos gráficos que consegue criar utilizando o pandas, então sempre que estiver com dúvidas ou dificuldades pode acessar a documentação para ver qual o código ou qual gráfico representa melhor seus dados.

https://pandas.pydata.org/docs/user_guide/visualization.html

Aula 3 – Análise Exploratória

Caso prefira esse conteúdo no formato de vídeo-aula, assista ao vídeo abaixo ou acesse o nosso canal do YouTube!