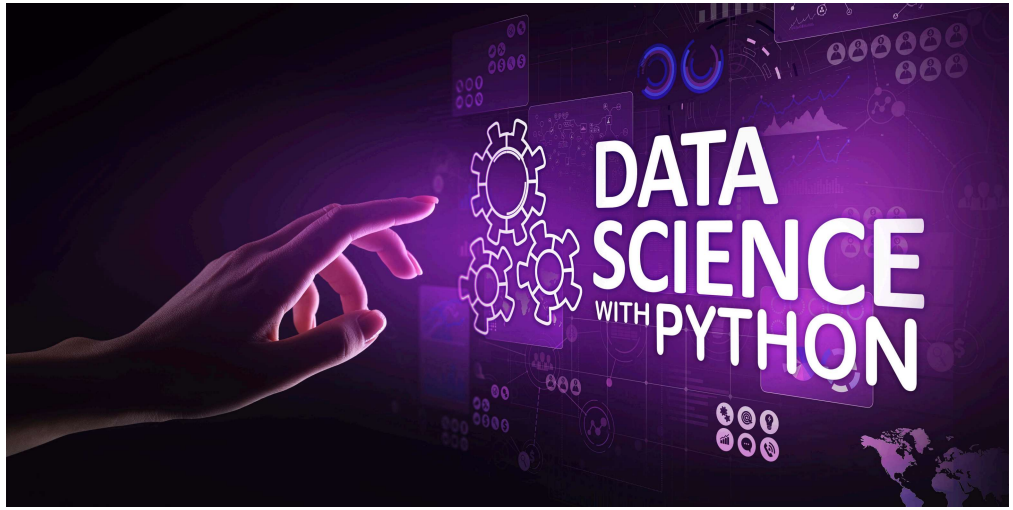
Week 1 Introduction to Data Science



Python for Data Science

Why is Python essential for Data Analysis?

This programming language is:

- flexible
- easy to learn
- open source
- well supported

Python is a widely used programming language. It is used by deep learning frameworks such as PyTorch and TensorFlow.

What we can do with data

1. Get or collect data
2. Manipulate and process data
3. Modeling and analysis
4. Visualize, evaluate, present, and communicate

Python review

Functions

In [1]:
```
def placesToVisit():
    someList = ["Germany", "China", "Saint Petersburg", "Kazakhstan"]
    return someList
print(f"An unordered list: \n {placesToVisit()}")
print(f"Ordered list: \n {sorted(placesToVisit())}")
```

```
An unordered list:
 ['Germany', 'China', 'Saint Petersburg', 'Kazakhstan']
Ordered list:
 ['China', 'Germany', 'Kazakhstan', 'Saint Petersburg']
```

Loops

In [2]:
```
sentence = ["This", "is", "list"]
for word in sentence:
    print(word)
```

```
This
is
list
```

List manipulation

In [3]:
```
sentence = ["This", "is", "awesome", "and", "amazing"]
startsWith_A = [word for word in sentence if word.startswith("a")]
startsWith_A
```

Out[3]:
```
['awesome', 'and', 'amazing']
```

This is the equivalent of:

In [4]:
```
sentence = ["This", "is", "awesome", "and", "amazing"]
startsWith_A = []
for word in sentence:
    if word.startswith("a"):
        startsWith_A.append(word)
startsWith_A
```

Out[4]:
```
['awesome', 'and', 'amazing']
```

Python libraries for Data Science

- **Numpy**: N-dimensional arrays, Matrices and Linear Algebra
- **Scipy**: Algorithms from linear algebra, optimization, statistics and signal processing
- **Pandas**: Data Manipulation and Analysis
- **Matplotlib**: Data Visualization
- **Scikit-learn**: Machine Learning

Data science platforms

- Anaconda
- Jupyter Notebook
- Google Colab

Pandas

- Pandas is a library for data manipulation and analysis
- Data structures: Series and DataFrame
- Data is loaded in-memory, hence super fast

In [5]:
```
import pandas as pd
```

Read file

In [6]:
```
# Load the CSV into a DataFrame and show the first 5 entries
# download from the link below Titanic-Dataset (train.csv)
# https://www.kaggle.com/datasets/hesh97/titanicdataset-traincsv?resource=download
df = pd.read_csv('train.csv')
df.head()
```

Out[6]:

	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

What does the data look like?

In [7]:
```
df.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
```

In [8]:
```
# Print the number of rows and columns in the data
df.shape
```

Out[8]:
```
(891, 12)
```

In [9]:
```
# Peek at the first two rows
df.head(2)
```

Out[9]:

	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

In [10]:
```
# from the peeked row print the 'Name' and 'Age' columns
```

In [11]:
```
df[['Name', 'Age']][:2]
```

Out[11]:

	Name	Age
0	Braund, Mr. Owen Harris	22.0
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0

In [12]:
```
# As you can see from the result above, the DataFrame is like a table with rows and columns.
# Pandas use the **loc** attribute to return one or more specified row(s)
# to locate the second row
df.loc[1]
```

Out[12]:
```
PassengerId                                     2
Survived                                        1
Pclass                                          1
Name           Cumings, Mrs. John Bradley (Florence Briggs Th...
Sex                                       female
Age                                         38.0
SibSp                                          1
Parch                                          0
Ticket                                  PC 17599
Fare                                     71.2833
Cabin                                        C85
Embarked                                       C
Name: 1, dtype: object
```

In [13]:
```
# Make the "Ticket" column become the index of the DataFrame:
df2 = df.set_index('Ticket')
```

In [14]:
```
# iloc property get or set the values of a group of elements in the specified positions
df2.iloc[2]
```

Out[14]:
```
PassengerId                          3
Survived                             1
Pclass                               3
Name           Heikkinen, Miss. Laina
Sex                            female
Age                              26.0
SibSp                               0
Parch                               0
Fare                            7.925
Cabin                             NaN
Embarked                            S
Name: STON/O2. 3101282, dtype: object
```

In [15]:
```
# loc property get or set the value of a group of elements specified using their labels
df2.loc['PC 17599']
```

Out[15]:
```
PassengerId                                                 2
Survived                                                    1
Pclass                                                      1
Name           Cumings, Mrs. John Bradley (Florence Briggs Th...
Sex                                                    female
Age                                                      38.0
SibSp                                                       1
Parch                                                       0
Fare                                                  71.2833
Cabin                                                     C85
Embarked                                                    C
Name: PC 17599, dtype: object
```

In [16]:
```
# Get row with the index==6
df.loc[6]
```

Out[16]:
```
PassengerId                          7
Survived                             0
Pclass                               1
Name           McCarthy, Mr. Timothy J
Sex                               male
Age                               54.0
SibSp                                0
Parch                                0
Ticket                           17463
Fare                           51.8625
Cabin                              E46
Embarked                             S
Name: 6, dtype: object
```

In [17]:
```
# Check datatype of each column
df.dtypes
```

Out[17]:
```
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
```

In [18]:
```
# Print summary statistics for *numerical* columns
df.describe()
```

Out[18]:

	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

In [19]:
```
# Count number of rows, for every value of `Sex`

df['Sex'].value_counts()
```

Out[19]:
```
male      577
female    314
Name: Sex, dtype: int64
```

In [20]:
```
df['Age'].value_counts()
```

Out[20]:
```
24.00    30
22.00    27
18.00    26
19.00    25
28.00    25
         ..
36.50     1
55.50     1
0.92      1
23.50     1
74.00     1
Name: Age, Length: 88, dtype: int64
```

In [21]:
```
df[(df['Age'] > 80)  | (df['Sex'] =='female')].shape
```

Out[21]:
```
(314, 12)
```

In [22]:
```
# We can use the logical operators on column values to filter rows
# We've now selected the rows in which the value in the 'Age' column is greater than 80
df[(df['Age'] > 80)  | (df['Sex'] =='female')]
```

Out[22]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
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
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C
...	...	...	...	...	...	...	...	...	...	...	...	...
880	881	1	2	Shelley, Mrs. William (Imanita Parrish Hall)	female	25.0	0	1	230433	26.0000	NaN	S
882	883	0	3	Dahlberg, Miss. Gerda Ulrika	female	22.0	0	0	7552	10.5167	NaN	S
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.1250	NaN	Q
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S

314 rows × 12 columns

Sorting

In [23]:
```
df.sort_values(by=['Age', 'Sex'], ascending=[True, False])
```

Out[23]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
803	804	1	3	Thomas, Master. Assad Alexander	male	0.42	0	1	2625	8.5167	NaN	C
755	756	1	2	Hamalainen, Master. Viljo	male	0.67	1	1	250649	14.5000	NaN	S
469	470	1	3	Baclini, Miss. Helene Barbara	female	0.75	2	1	2666	19.2583	NaN	C
644	645	1	3	Baclini, Miss. Eugenie	female	0.75	2	1	2666	19.2583	NaN	C
78	79	1	2	Caldwell, Master. Alden Gates	male	0.83	0	2	248738	29.0000	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
727	728	1	3	Mannion, Miss. Margareth	female	NaN	0	0	36866	7.7375	NaN	Q
792	793	0	3	Sage, Miss. Stella Anna	female	NaN	8	2	CA. 2343	69.5500	NaN	S
849	850	1	1	Goldenberg, Mrs. Samuel L (Edwiga Grabowska)	female	NaN	1	0	17453	89.1042	C92	C
863	864	0	3	Sage, Miss. Dorothy Edith "Dolly"	female	NaN	8	2	CA. 2343	69.5500	NaN	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S

891 rows × 12 columns

Preprocessing

In [24]:
```
df.isnull().sum()
```

Out[24]:
```
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
```

In [25]:
```
df.shape
```

Out[25]:
```
(891, 12)
```

Replacing missing values

In [26]:
```
df.fillna({'Embarked':"Not specified", "Age":35})
```

Out[26]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	35.0	1	2	W./C. 6607	23.4500	NaN	S

In [27]:
```
temp = df['Age'].fillna(35)
```

In [28]:
```
temp.isnull().sum()
```

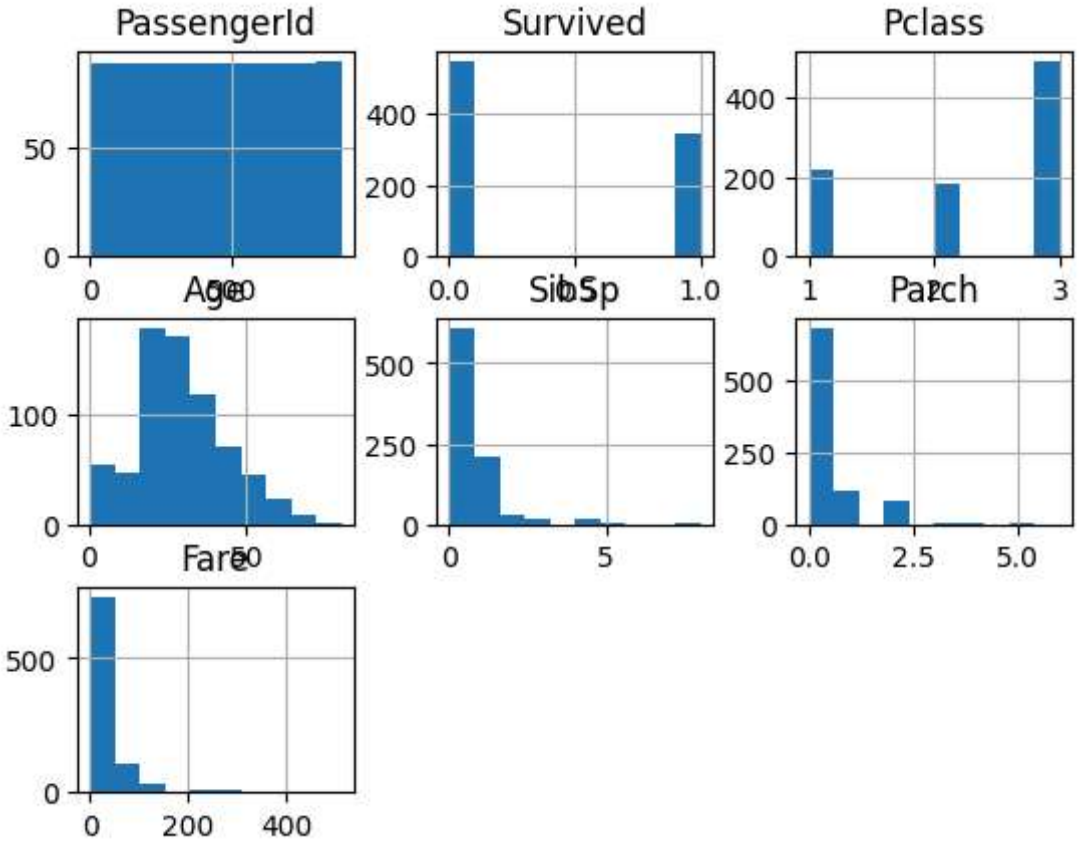
Out[28]:
```
0
```

In [29]:
```
df.shape
```

Out[29]:
```
(891, 12)
```

Let's visualize

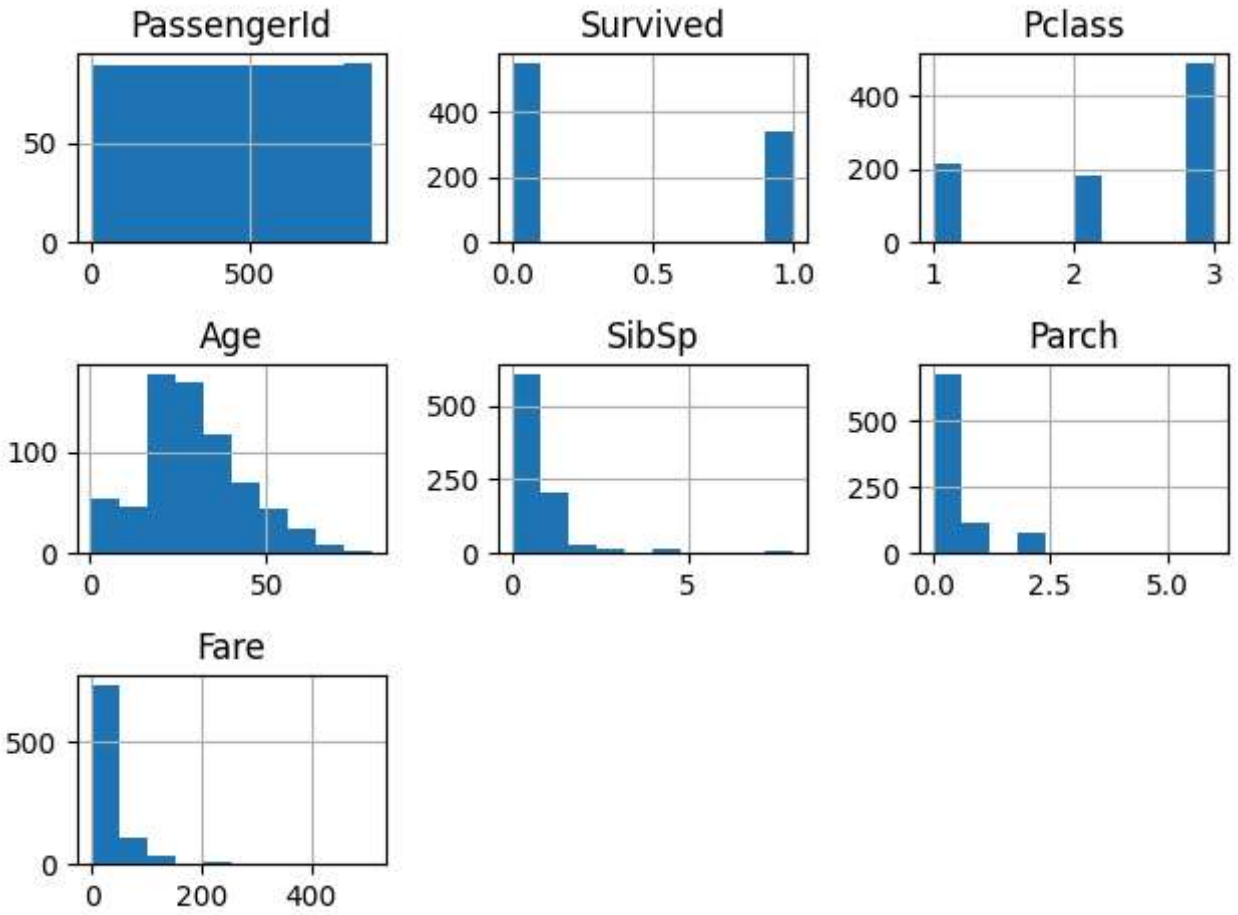
In [30]:
```
hists = df.hist()
```



Matplotlib helps keep things cleaner!

In [31]:
```
import matplotlib.pyplot as plt
```

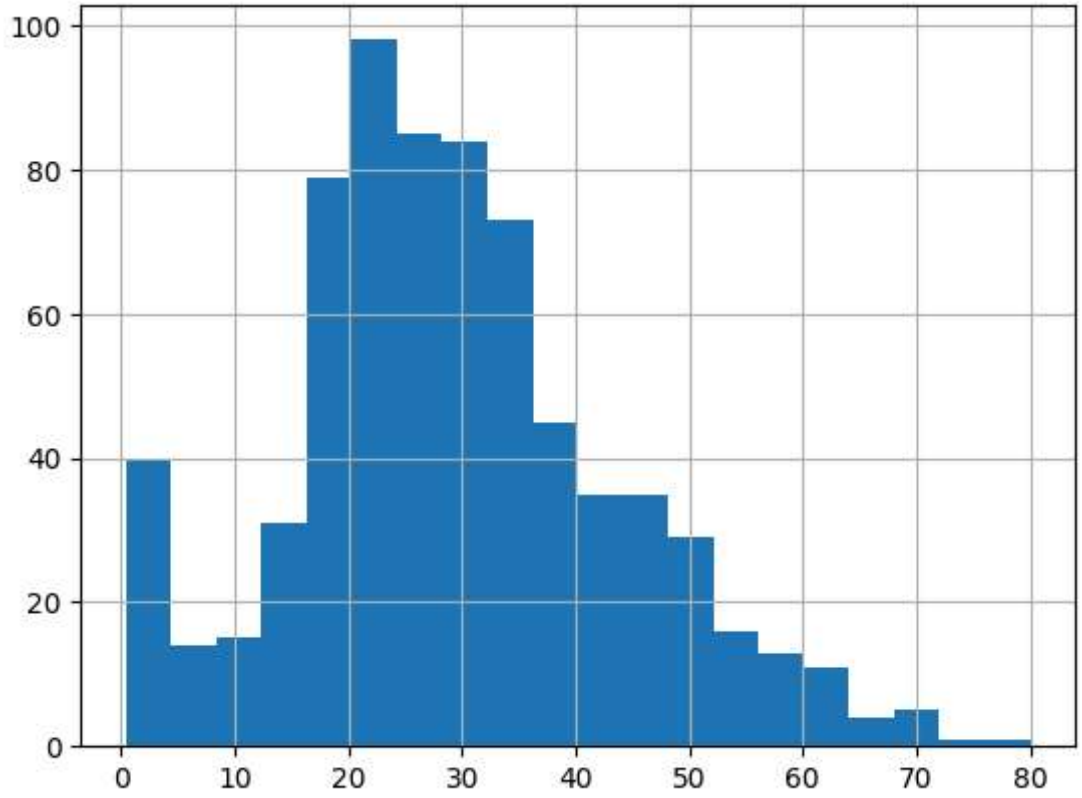
In [32]:
```
hists = df.hist()
plt.tight_layout()
plt.show()
plt.savefig('plot.png')
```




```
<Figure size 640x480 with 0 Axes>
```

In [33]:
```
df['Age'].hist(bins=20)
```

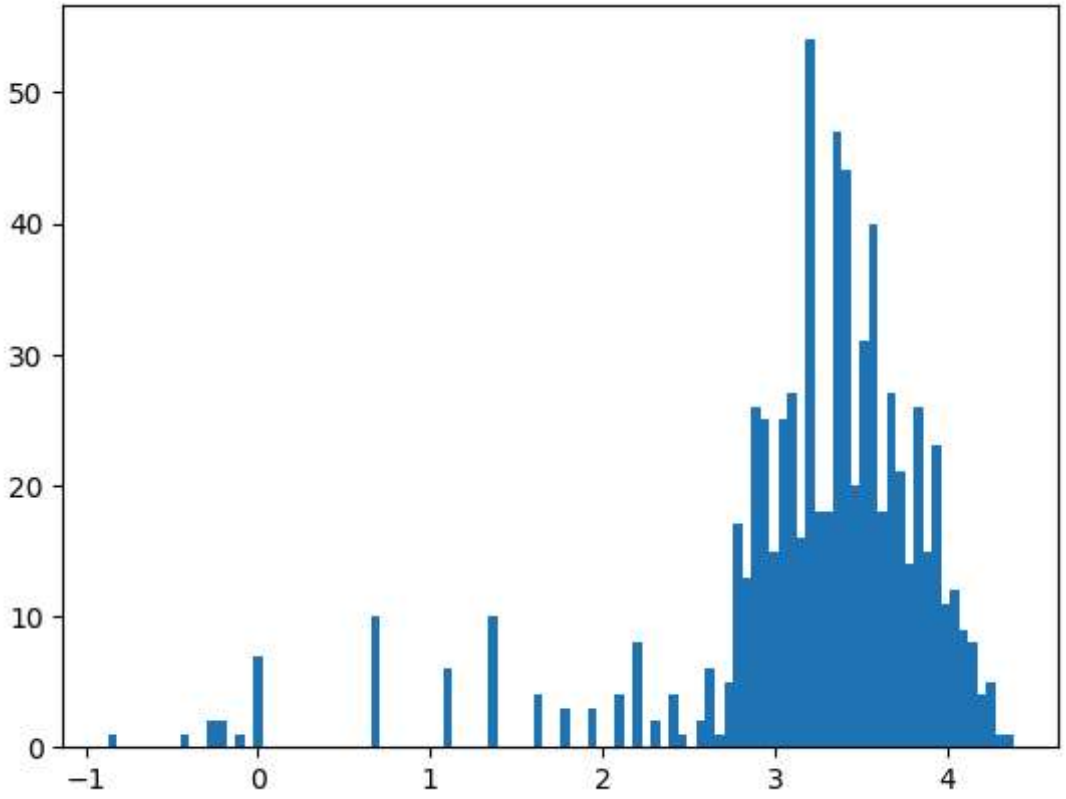
Out[33]: <AxesSubplot:>



In [34]:
```
import numpy as np
```

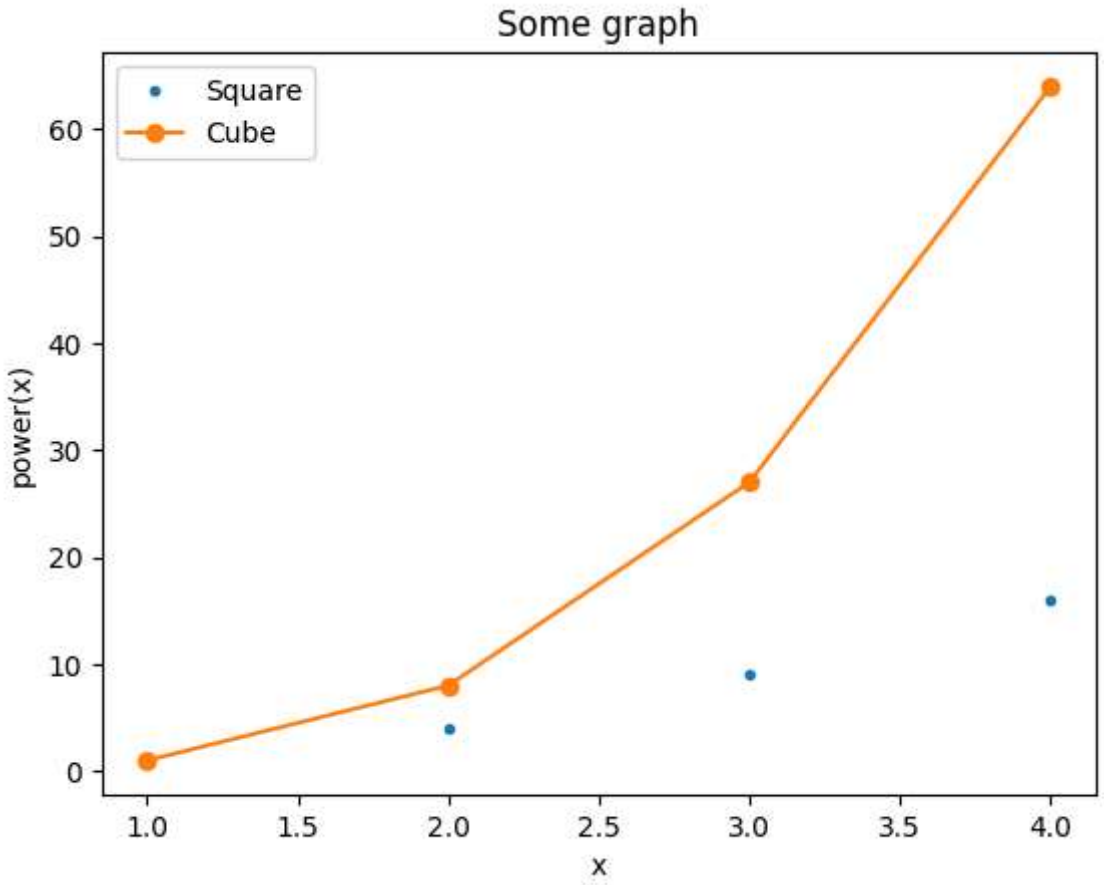
In [35]:
```
plt.hist(np.log([x for x in df['Age'] if x]), bins=100)
```

Out[35]:
```
(array([ 1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  2.,  2.,
         0.,  1.,  0.,  7.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  0., 10.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  6.,  0.,
         0.,  0.,  0., 10.,  0.,  0.,  0.,  0.,  4.,  0.,  0.,  3.,  0.,
         0.,  3.,  0.,  0.,  4.,  0.,  8.,  0.,  2.,  0.,  4.,  1.,  0.,
         2.,  6.,  1.,  5., 17., 13., 26., 25., 15., 25., 27., 16., 54.,
        18., 18., 47., 44., 20., 31., 40., 18., 27., 21., 14., 26., 15.,
        23., 11., 12.,  9.,  8.,  4.,  5.,  1.,  1.]),
 array([-0.86750057, -0.8150053 , -0.76251002, -0.71001475, -0.65751948,
        -0.60502421, -0.55252894, -0.50003366, -0.44753839, -0.39504312,
        -0.34254785, -0.29005258, -0.2375573 , -0.18506203, -0.13256676,
        -0.08007149, -0.02757622,  0.02491906,  0.07741433,  0.1299096 ,
         0.18240487,  0.23490014,  0.28739542,  0.33989069,  0.39238596,
         0.44488123,  0.4973765 ,  0.54987178,  0.60236705,  0.65486232,
         0.70735759,  0.75985287,  0.81234814,  0.86484341,  0.91733868,
         0.96983395,  1.02232923,  1.0748245 ,  1.12731977,  1.17981504,
         1.23231031,  1.28480559,  1.33730086,  1.38979613,  1.4422914 ,
         1.49478667,  1.54728195,  1.59977722,  1.65227249,  1.70476776,
         1.75726303,  1.80975831,  1.86225358,  1.91474885,  1.96724412,
         2.01973939,  2.07223467,  2.12472994,  2.17722521,  2.22972048,
         2.28221575,  2.33471103,  2.3872063 ,  2.43970157,  2.49219684,
         2.54469211,  2.59718739,  2.64968266,  2.70217793,  2.7546732 ,
         2.80716847,  2.85966375,  2.91215902,  2.96465429,  3.01714956,
         3.06964483,  3.12214011,  3.17463538,  3.22713065,  3.27962592,
         3.33212119,  3.38461647,  3.43711174,  3.48960701,  3.54210228,
         3.59459755,  3.64709283,  3.6995881 ,  3.75208337,  3.80457864,
         3.85707391,  3.90956919,  3.96206446,  4.01455973,  4.067055  ,
         4.11955027,  4.17204555,  4.22454082,  4.27703609,  4.32953136,
         4.38202663]),
 <BarContainer object of 100 artists>)
```


50
40
30
20
10
0
−1
0
1
2
3
4

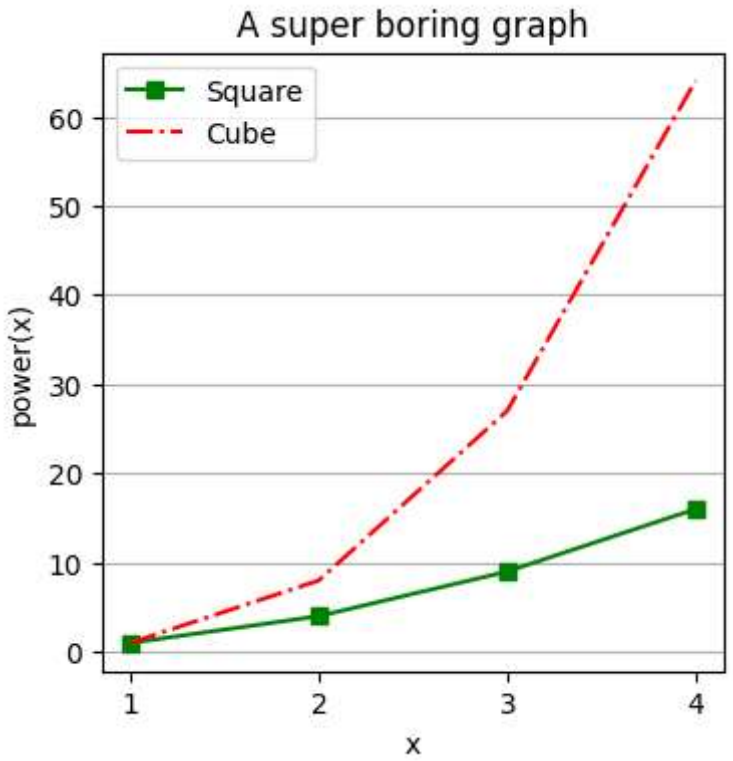
In [36]:
```
nums = [1,2,3,4]
plt.plot(nums, [x * x for x in nums], '.');
plt.plot(nums, [x ** 3 for x in nums], '-o');
plt.legend(['Square', 'Cube']);
plt.xlabel('x');
plt.ylabel('power(x)');
plt.title('Some graph')
```

Out[36]:
```
Text(0.5, 1.0, 'Some graph')
```



In [37]:
```
nums = [1,2,3,4]
plt.figure(figsize=(4,4))
plt.plot(nums, [x * x for x in nums], '-s', color='Green');
plt.plot(nums, [x * x * x for x in nums], '-.', color='Red');
plt.legend(['Square', 'Cube'])
plt.xlabel('x');
plt.ylabel('power(x)');

plt.xticks(nums)
plt.title('A super boring graph')
plt.grid(axis='y')
# plt.show()
```



In [38]:
```
df.head(5)
```

Out[38]:

	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

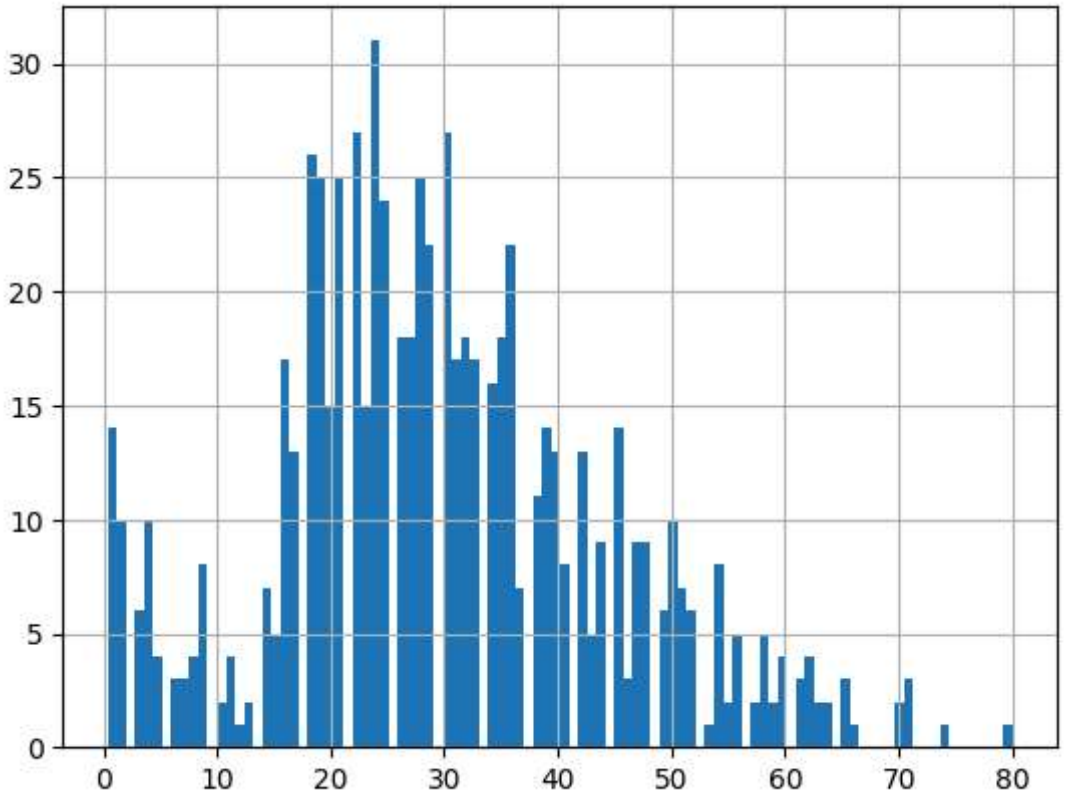
In [39]:
```
df.corr()
```

Out[39]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

Visualizing distributions

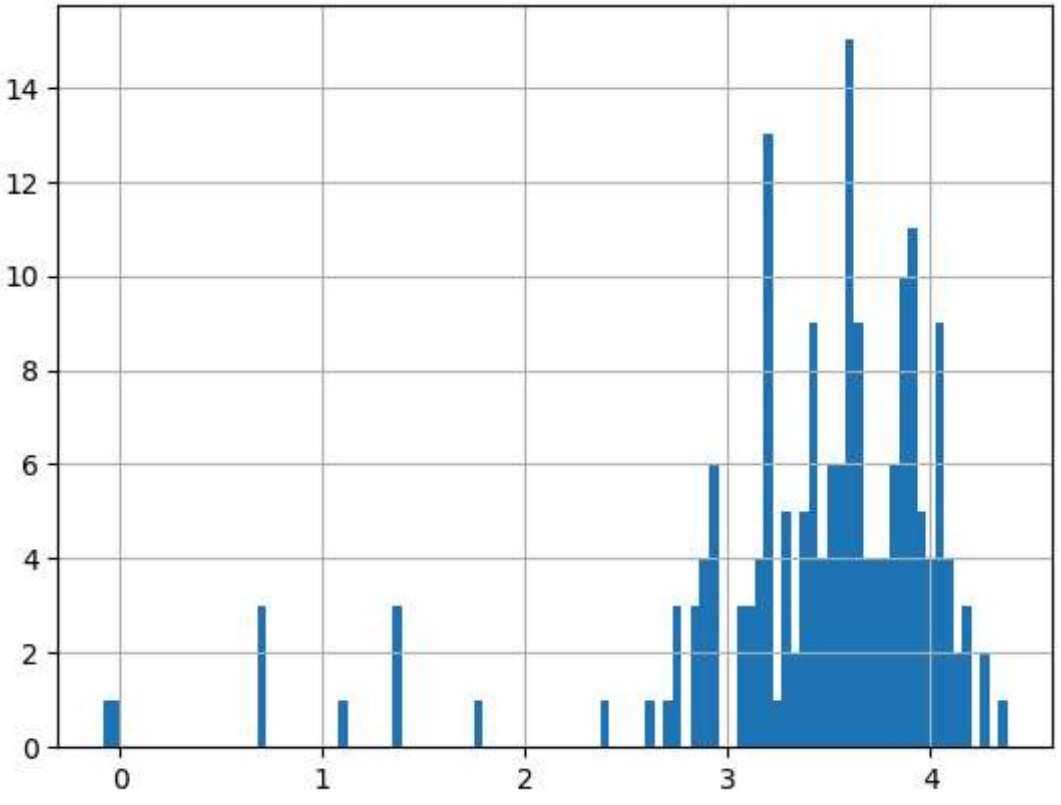
In [40]:
```
df['Age'].hist(bins=100);
```



In [41]:
```
import numpy as np
```

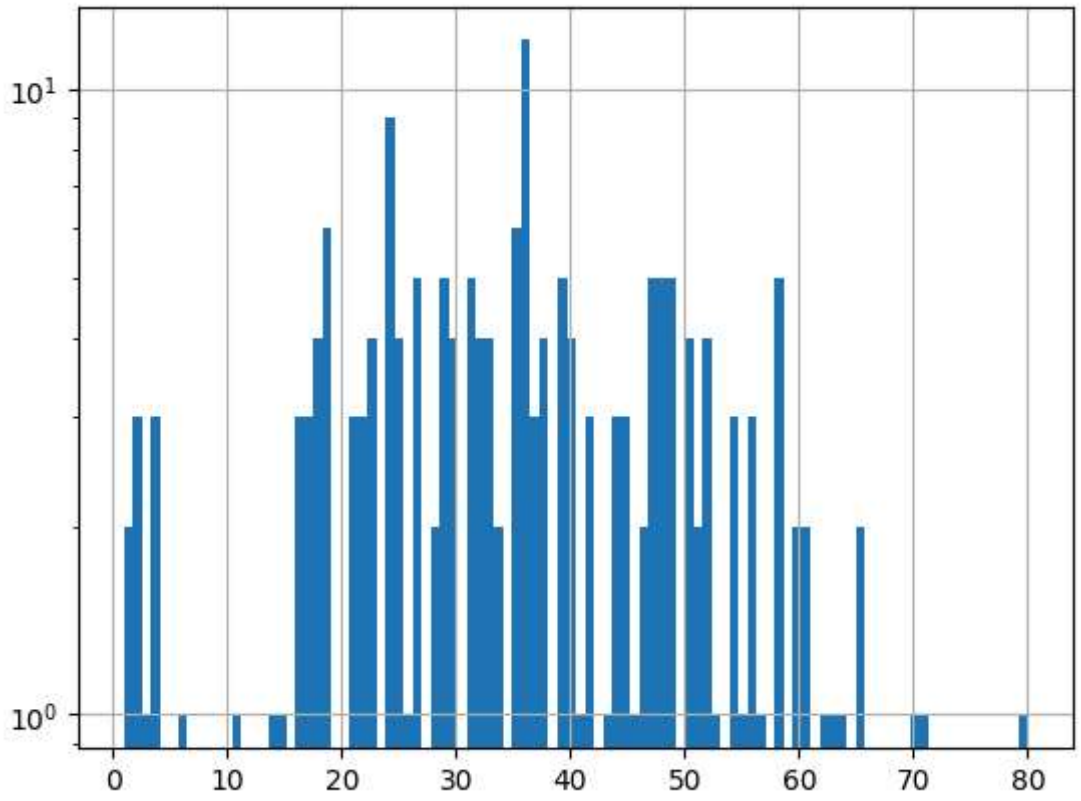
In [42]:
```
df.dropna(how='any', inplace=True)
```

In [43]:
```
np.log(df['Age']).hist(bins=100);
```



In [44]:
```
df['Age'].hist(log=True, bins=100)
```

Out[44]: <AxesSubplot:>



Predict label

In [45]:
```
df['Age'].value_counts()
```

Out[45]:
```
36.0    11
24.0     9
19.0     6
35.0     6
31.0     5
        ..
71.0     1
3.0      1
1.0      1
36.5     1
26.0     1
Name: Age, Length: 63, dtype: int64
```

In [46]:
```
middleAged = df['Age']
ages = list(middleAged.value_counts()[:50].index)
test = df['Age'].apply(\
  lambda x: x if x in ages else 'Other')
```

In [47]:
```
middleAged.value_counts()[:50].index
```

Out[47]:
```
Float64Index([36.0, 24.0, 19.0, 35.0, 31.0, 29.0, 49.0, 47.0, 27.0, 58.0, 48.0,
              39.0, 33.0, 52.0, 30.0, 40.0, 50.0],
             dtype='float64')
```

In [48]:
```
test.value_counts()
```

Out[48]:
```
Other    91
36.0     11
24.0      9
35.0      6
19.0      6
48.0      5
27.0      5
31.0      5
39.0      5
47.0      5
29.0      5
49.0      5
58.0      5
50.0      4
30.0      4
52.0      4
33.0      4
40.0      4
Name: Age, dtype: int64
```

In [49]:
```
test
```

Out[49]:
```
1      Other
3       35.0
6      Other
10     Other
11      58.0
       ...
871     47.0
872     33.0
879    Other
887     19.0
889    Other
Name: Age, Length: 183, dtype: object
```