welcome-to-colaboratory

September 3, 2024

Welcome to Colab!

(New) Try the Gemini API

Generate a Gemini API key

Talk to Gemini with the Speech-to-Text API

Gemini API: Quickstart with Python

Gemini API code sample

Compare Gemini with ChatGPT

More notebooks

If you’re already familiar with Colab, check out this video to learn about interactive tables, the executed code history view and the command palette.

[ ]:
```

```

What is Colab?

Colab, or ‘Colaboratory’, allows you to write and execute Python in your browser, with - Zero configuration required - Access to GPUs free of charge - Easy sharing

Whether you’re a student, a data scientist or an AI researcher, Colab can make your work easier. Watch Introduction to Colab to find out more, or just get started below!

0.1 Getting started

The document that you are reading is not a static web page, but an interactive environment called a Colab notebook that lets you write and execute code.

For example, here is a code cell with a short Python script that computes a value, stores it in a variable and prints the result:

[ ]:
```
seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

[ ]:
```
86400
```

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut ‘Command/Ctrl+Enter’. To edit the code, just click the cell and start editing.

Variables that you define in one cell can later be used in other cells:

```
[ ]: seconds_in_a_week = 7 * seconds_in_a_day
     seconds_in_a_week
```

```
[ ]: 604800
```

Colab notebooks allow you to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more. When you create your own Colab notebooks, they are stored in your Google Drive account. You can easily share your Colab notebooks with co-workers or friends, allowing them to comment on your notebooks or even edit them. To find out more, see Overview of Colab. To create a new Colab notebook you can use the File menu above, or use the following link: Create a new Colab notebook.

Colab notebooks are Jupyter notebooks that are hosted by Colab. To find out more about the Jupyter project, see jupyter.org.

0.2 Data science

With Colab you can harness the full power of popular Python libraries to analyse and visualise data. The code cell below uses numpy to generate some random data, and uses matplotlib to visualise it. To edit the code, just click the cell and start editing.

```
[ ]: import numpy as np
     import IPython.display as display
     from matplotlib import pyplot as plt
     import io
     import base64

     ys = 200 + np.random.randn(100)
     x = [x for x in range(len(ys))]

     fig = plt.figure(figsize=(4, 3), facecolor='w')
     plt.plot(x, ys, '-')
     plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)
     plt.title("Sample Visualization", fontsize=10)

     data = io.BytesIO()
     plt.savefig(data)
     image = F"data:image/png;base64,{base64.b64encode(data.getvalue()).decode()}"
     alt = "Sample Visualization"
     display.display(display.Markdown(F"""![{alt}]({image})"""))
     plt.close(fig)
```

You can import your own data into Colab notebooks from your Google Drive account, including from spreadsheets, as well as from GitHub and many other sources. To find out more about importing data, and how Colab can be used for data science, see the links below under Working with data.

0.3 Machine learning

With Colab you can import an image dataset, train an image classifier on it, and evaluate the model, all in just a few lines of code. Colab notebooks execute code on Google's cloud servers, meaning you can leverage the power of Google hardware, including GPUs and TPUs, regardless of the power of your machine. All you need is a browser.

Colab is used extensively in the machine learning community with applications including: - Getting started with TensorFlow - Developing and training neural networks - Experimenting with TPUs - Disseminating AI research - Creating tutorials

To see sample Colab notebooks that demonstrate machine learning applications, see the machine learning examples below.

0.4 More resources

0.4.1 Working with notebooks in Colab

- Overview of Colaboratory
- Guide to markdown
- Importing libraries and installing dependencies
- Saving and loading notebooks in GitHub
- Interactive forms
- Interactive widgets

Working with data

- Loading data: Drive, Sheets and Google Cloud Storage
- Charts: visualising data
- Getting started with BigQuery

0.4.2 Machine learning crash course

These are a few of the notebooks from Google's online machine learning course. See the full course website for more. - Intro to Pandas DataFrame - Linear regression with tf.keras using synthetic data

Using accelerated hardware

- TensorFlow with GPUs
- TensorFlow with TPUs

0.4.3 Featured examples

- NeMo voice swap: Use Nvidia NeMo conversational AI toolkit to swap a voice in an audio fragment with a computer-generated one.

- Retraining an Image Classifier: Build a Keras model on top of a pre-trained image classifier to distinguish flowers.

- Text Classification: Classify IMDB film reviews as either positive or negative.
- Style Transfer: Use deep learning to transfer style between images.
- Multilingual Universal Sentence Encoder Q&A: Use a machine-learning model to answer questions from the SQuAD dataset.
- Video Interpolation: Predict what happened in a video between the first and the last frame.

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv("/content/titanic_dataset.csv")
df
```

[2]:
```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           …         …     …
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                  Name     Sex   Age  SibSp  \
0                              Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                               Heikkinen, Miss. Laina  female  26.0      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                             Allen, Mr. William Henry    male  35.0      0
..                                                 …     …   …    …
886                              Montvila, Rev. Juozas    male  27.0      0
887                       Graham, Miss. Margaret Edith  female  19.0      0
888           Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                              Behr, Mr. Karl Howell    male  26.0      0
890                                Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare Cabin Embarked
0        0         A/5 21171   7.2500   NaN        S
1        0          PC 17599  71.2833   C85        C
2        0  STON/O2. 3101282   7.9250   NaN        S
3        0            113803  53.1000  C123        S
4        0            373450   8.0500   NaN        S
..     …               …      …   …      …
```

```
886       0           211536  13.0000    NaN          S
887       0           112053  30.0000    B42          S
888       2       W./C. 6607  23.4500    NaN          S
889       0           111369  30.0000   C148          C
890       0           370376   7.7500    NaN          Q

[891 rows x 12 columns]
```

[8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 891 entries, 1 to 891
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Name      891 non-null    object
 3   Sex       891 non-null    object
 4   Age       714 non-null    float64
 5   SibSp     891 non-null    int64
 6   Parch     891 non-null    int64
 7   Ticket    891 non-null    object
 8   Fare      891 non-null    float64
 9   Cabin     204 non-null    object
 10  Embarked  889 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 83.5+ KB
```

[9]:
```
df.set_index("PassengerId",inplace=True)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-9-b742e7ac24b7> in <cell line: 1>()
----> 1 df.set_index("PassengerId",inplace=True)

/usr/local/lib/python3.10/dist-packages/pandas/core/frame.py in set_index(self,
  ↪keys, drop, append, inplace, verify_integrity)
   5868
   5869         if missing:
-> 5870             raise KeyError(f"None of {missing} are in the columns")
   5871
   5872         if inplace:

KeyError: "None of ['PassengerId'] are in the columns"
```

[10]:
```
df
```

[10]:
```
             Survived  Pclass  \
PassengerId
1                   0       3
2                   1       1
3                   1       3
4                   1       1
5                   0       3
…                 …     …
887                 0       2
888                 1       1
889                 0       3
890                 1       1
891                 0       3

                                                          Name     Sex   Age  \
PassengerId
1                                      Braund, Mr. Owen Harris    male  22.0
2            Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0
3                                       Heikkinen, Miss. Laina  female  26.0
4                 Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0
5                                     Allen, Mr. William Henry    male  35.0
…                                                          …     …   …
887                                      Montvila, Rev. Juozas    male  27.0
888                               Graham, Miss. Margaret Edith  female  19.0
889                   Johnston, Miss. Catherine Helen "Carrie"  female   NaN
890                                      Behr, Mr. Karl Howell    male  26.0
891                                        Dooley, Mr. Patrick    male  32.0

             SibSp  Parch            Ticket     Fare Cabin Embarked
PassengerId
1                1      0         A/5 21171   7.2500   NaN        S
2                1      0          PC 17599  71.2833   C85        C
3                0      0  STON/O2. 3101282   7.9250   NaN        S
4                1      0            113803  53.1000  C123        S
5                0      0            373450   8.0500   NaN        S
…              …    …               …      …   …      …
887              0      0            211536  13.0000   NaN        S
888              0      0            112053  30.0000   B42        S
889              1      2        W./C. 6607  23.4500   NaN        S
890              0      0            111369  30.0000  C148        C
891              0      0            370376   7.7500   NaN        Q

[891 rows x 11 columns]
```

[11]:
```
df.nunique()
```

[11]:
```
Survived      2
Pclass        3
Name        891
Sex           2
Age          88
SibSp         7
Parch         7
Ticket      681
Fare        248
Cabin       147
Embarked      3
dtype: int64
```

[12]:
```
df["Survived"].value_counts()
```

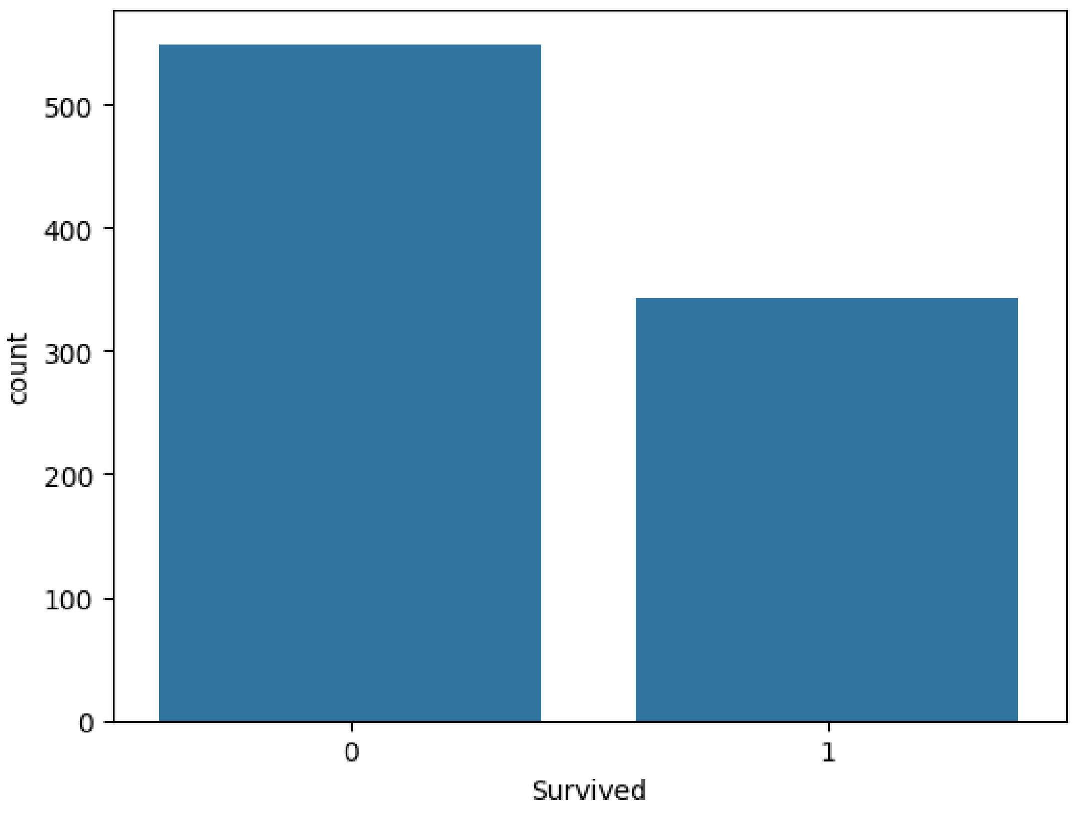
[12]:
```
Survived
0    549
1    342
Name: count, dtype: int64
```

[13]:
```
per=(df["Survived"].value_counts()/df.shape[0]*100).round(2)
per
```

[13]:
```
Survived
0    61.62
1    38.38
Name: count, dtype: float64
```

[14]:
```
sns.countplot(data=df,x="Survived")
```

[14]:
```
<Axes: xlabel='Survived', ylabel='count'>
```



[15]:
```
df.Pclass.unique()
```

[15]:
```
array([3, 1, 2])
```

[17]:
```
df.rename(columns={"Sex":"Gender"},inplace=True)
df
```

[17]:
```
             Survived  Pclass  \
PassengerId
1                   0       3
2                   1       1
3                   1       3
4                   1       1
5                   0       3
…                 …     …
887                 0       2
888                 1       1
889                 0       3
890                 1       1
891                 0       3
```

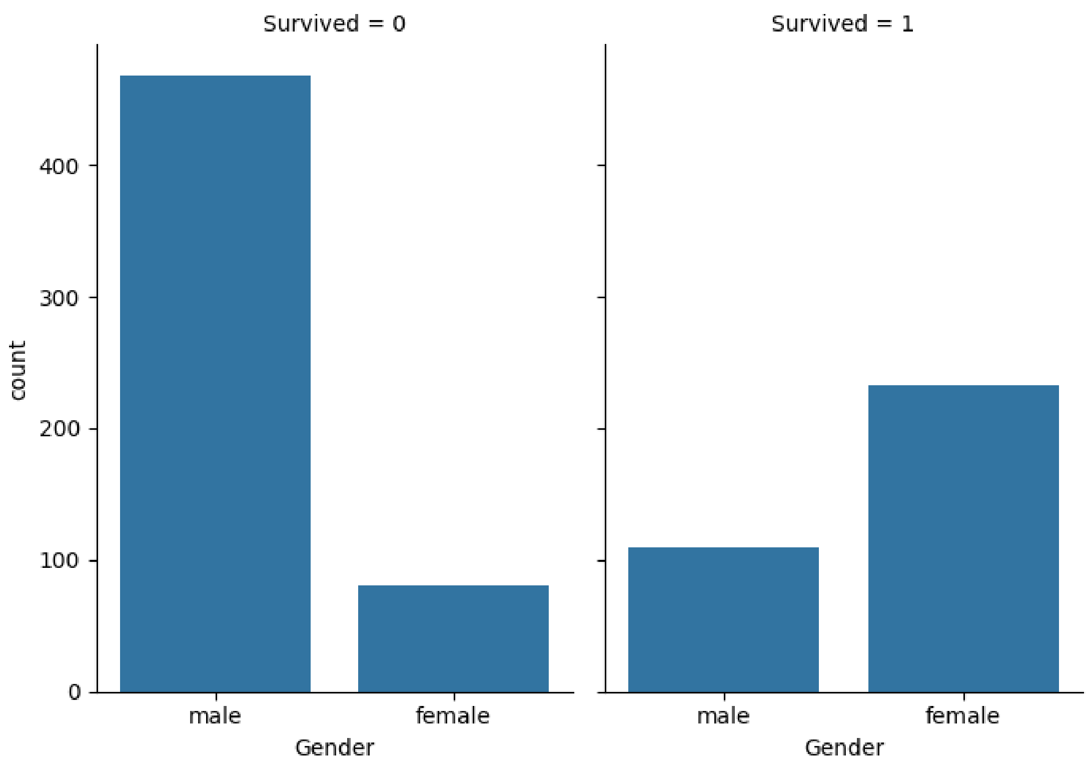
```
                                                       Name  Gender   Age  \
PassengerId
1                                   Braund, Mr. Owen Harris    male  22.0
2            Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0
3                                    Heikkinen, Miss. Laina  female  26.0
4              Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0
5                                  Allen, Mr. William Henry    male  35.0
…                                                        …      …    …
887                                   Montvila, Rev. Juozas    male  27.0
888                            Graham, Miss. Margaret Edith  female  19.0
889                Johnston, Miss. Catherine Helen "Carrie"  female   NaN
890                                   Behr, Mr. Karl Howell    male  26.0
891                                     Dooley, Mr. Patrick    male  32.0

             SibSp  Parch            Ticket     Fare Cabin Embarked
PassengerId
1                1      0         A/5 21171   7.2500   NaN        S
2                1      0          PC 17599  71.2833   C85        C
3                0      0  STON/O2. 3101282   7.9250   NaN        S
4                1      0            113803  53.1000  C123        S
5                0      0            373450   8.0500   NaN        S
…              …    …               …      …    …
887              0      0            211536  13.0000   NaN        S
888              0      0            112053  30.0000   B42        S
889              1      2        W./C. 6607  23.4500   NaN        S
890              0      0            111369  30.0000  C148        C
891              0      0            370376   7.7500   NaN        Q

[891 rows x 11 columns]
```

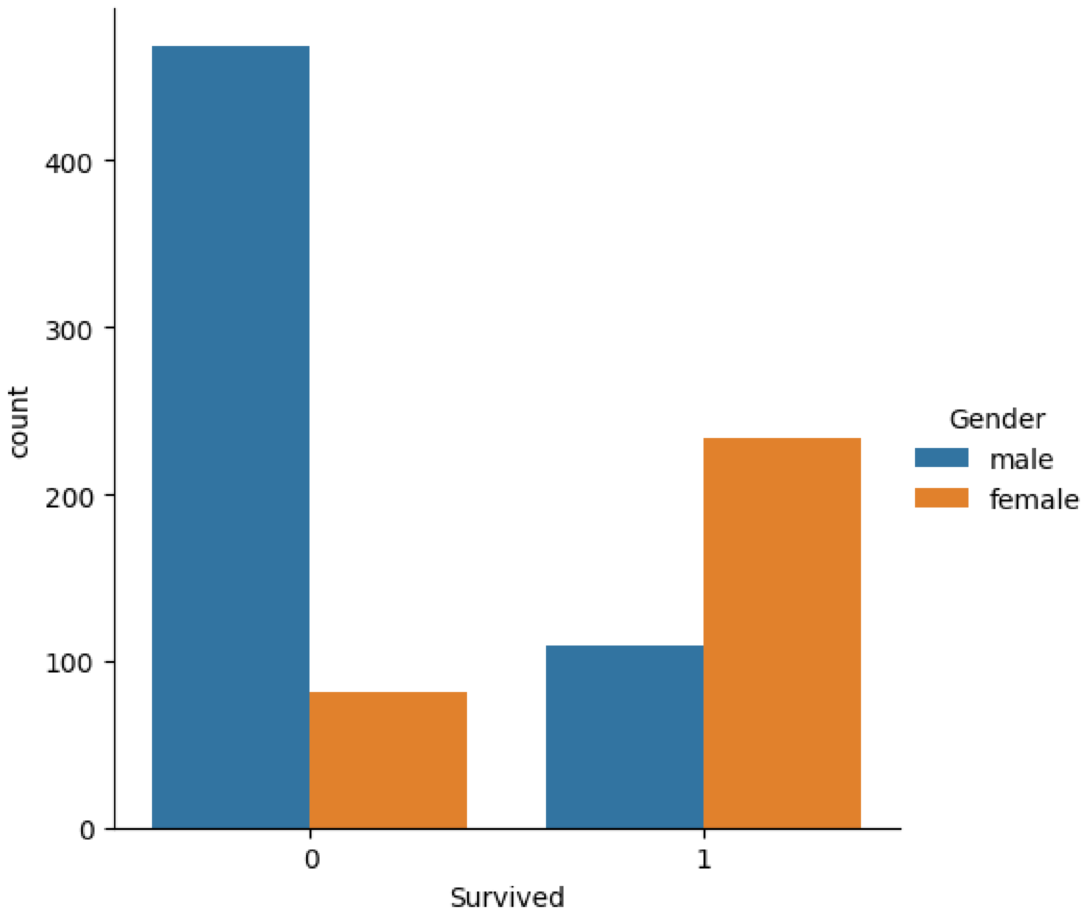
[19]:
```python
sns.catplot(x="Gender",col="Survived",kind="count",data=df,height=5,aspect=.7)
```

[19]:
```
<seaborn.axisgrid.FacetGrid at 0x79bb46d03970>
```



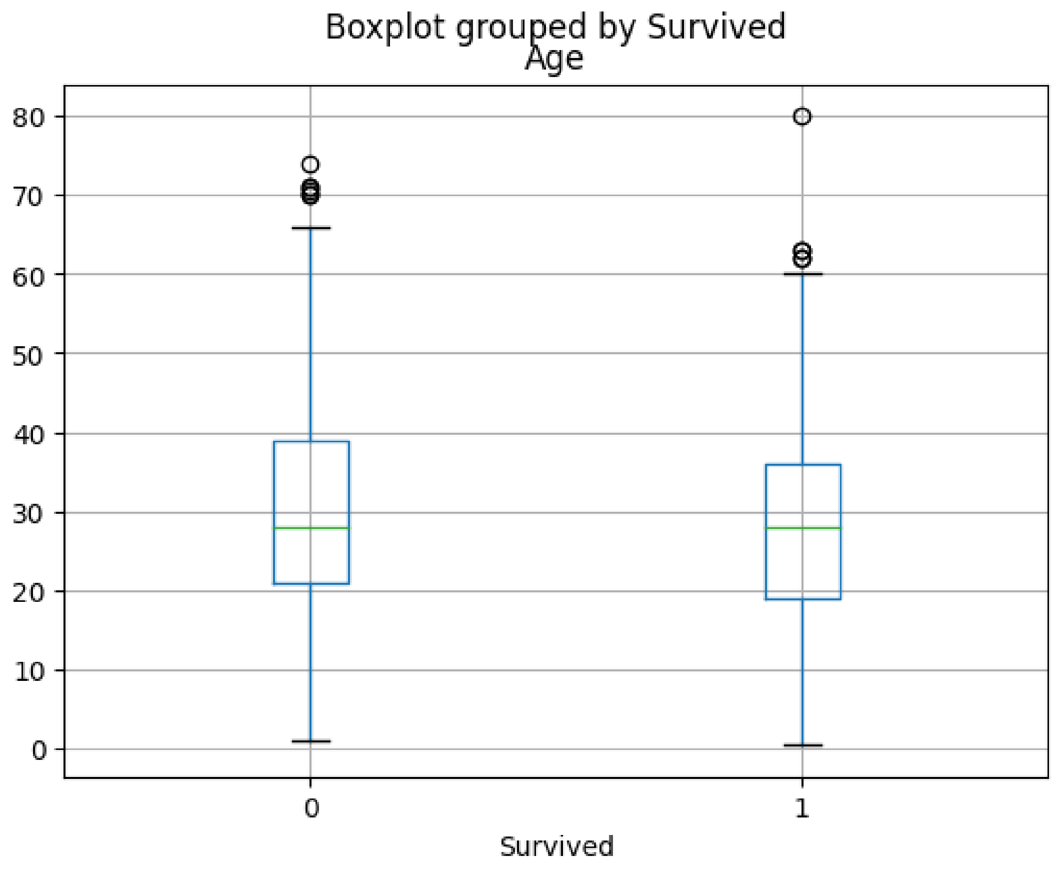
```
[21]: sns.catplot(x="Survived",hue="Gender",data=df,kind="count")
```

```
[21]: <seaborn.axisgrid.FacetGrid at 0x79bb46ce1c60>
```



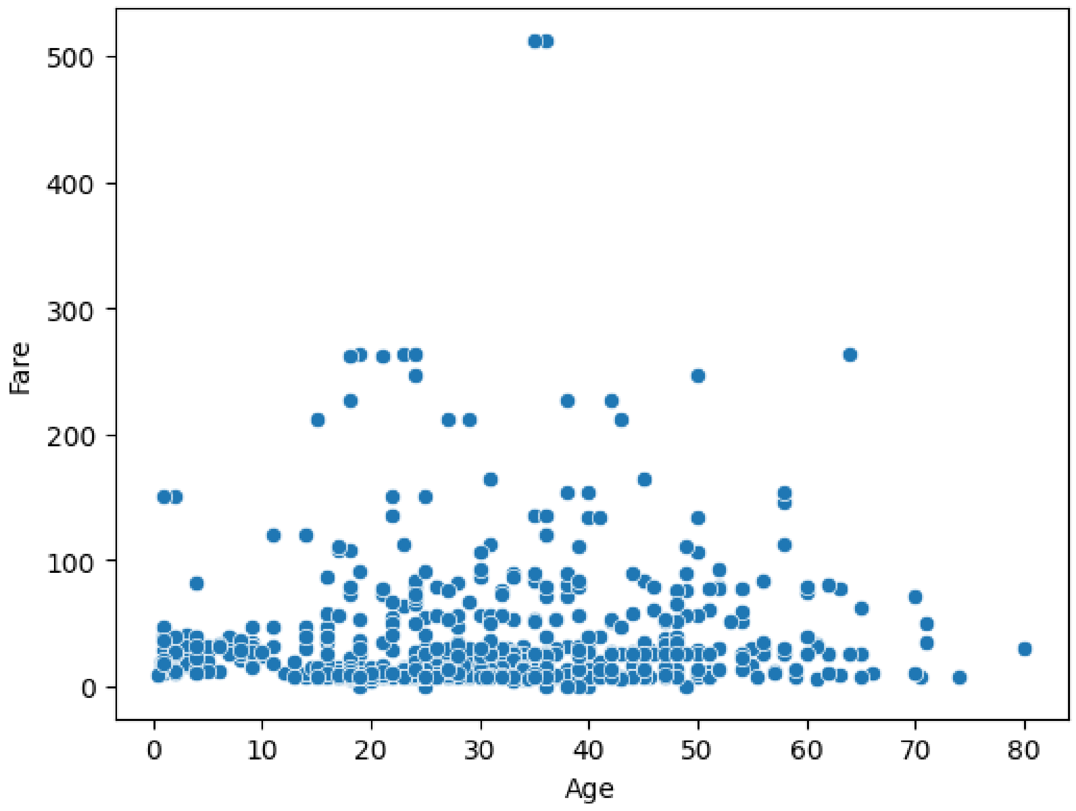
```
[23]: df.boxplot(column="Age",by="Survived")
```

```
[23]: <Axes: title={'center': 'Age'}, xlabel='Survived'>
```



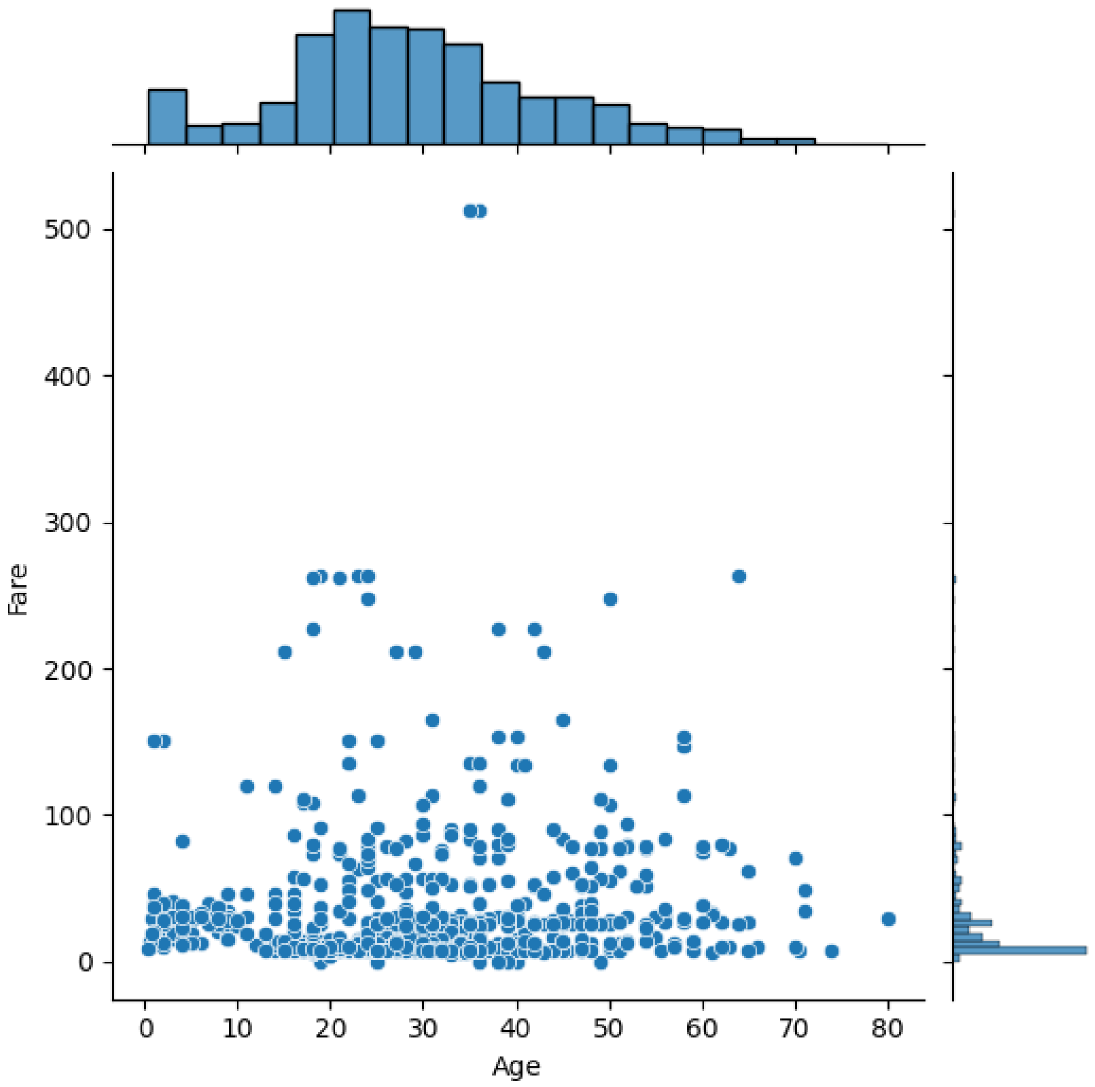
```
[24]: sns.scatterplot(x=df["Age"],y=df["Fare"])
```

```
[24]: <Axes: xlabel='Age', ylabel='Fare'>
```



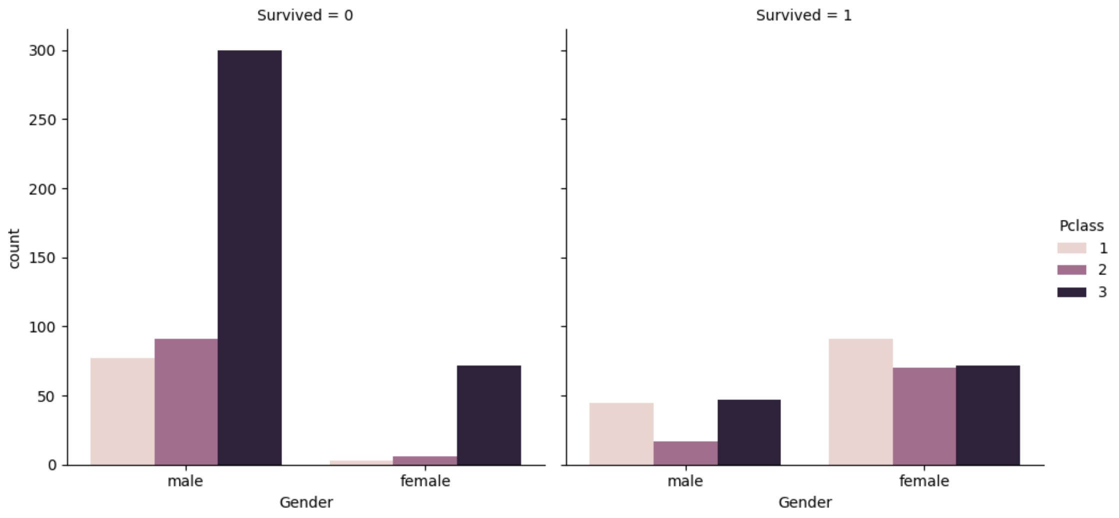
```
[25]: sns.jointplot(x="Age",y="Fare",data=df)
```

```
[25]: <seaborn.axisgrid.JointGrid at 0x79bb45bc4100>
```



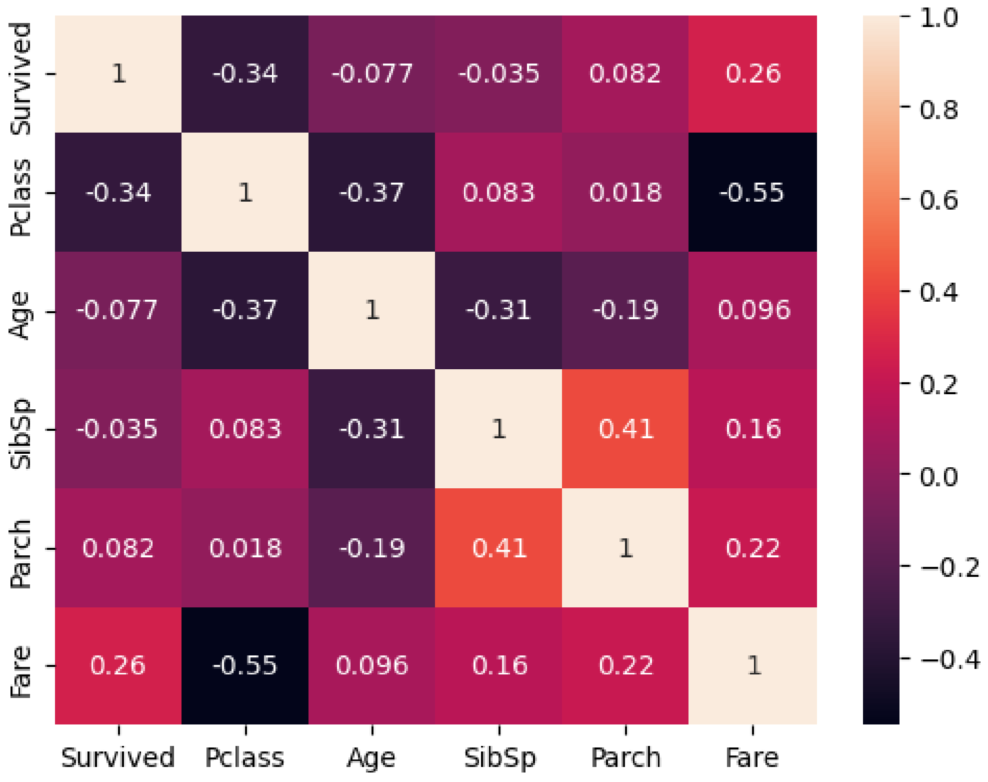
```
[27]: sns.catplot(col="Survived",x="Gender",hue="Pclass",data=df,kind="count")
```

```
[27]: <seaborn.axisgrid.FacetGrid at 0x79bb458d2d40>
```



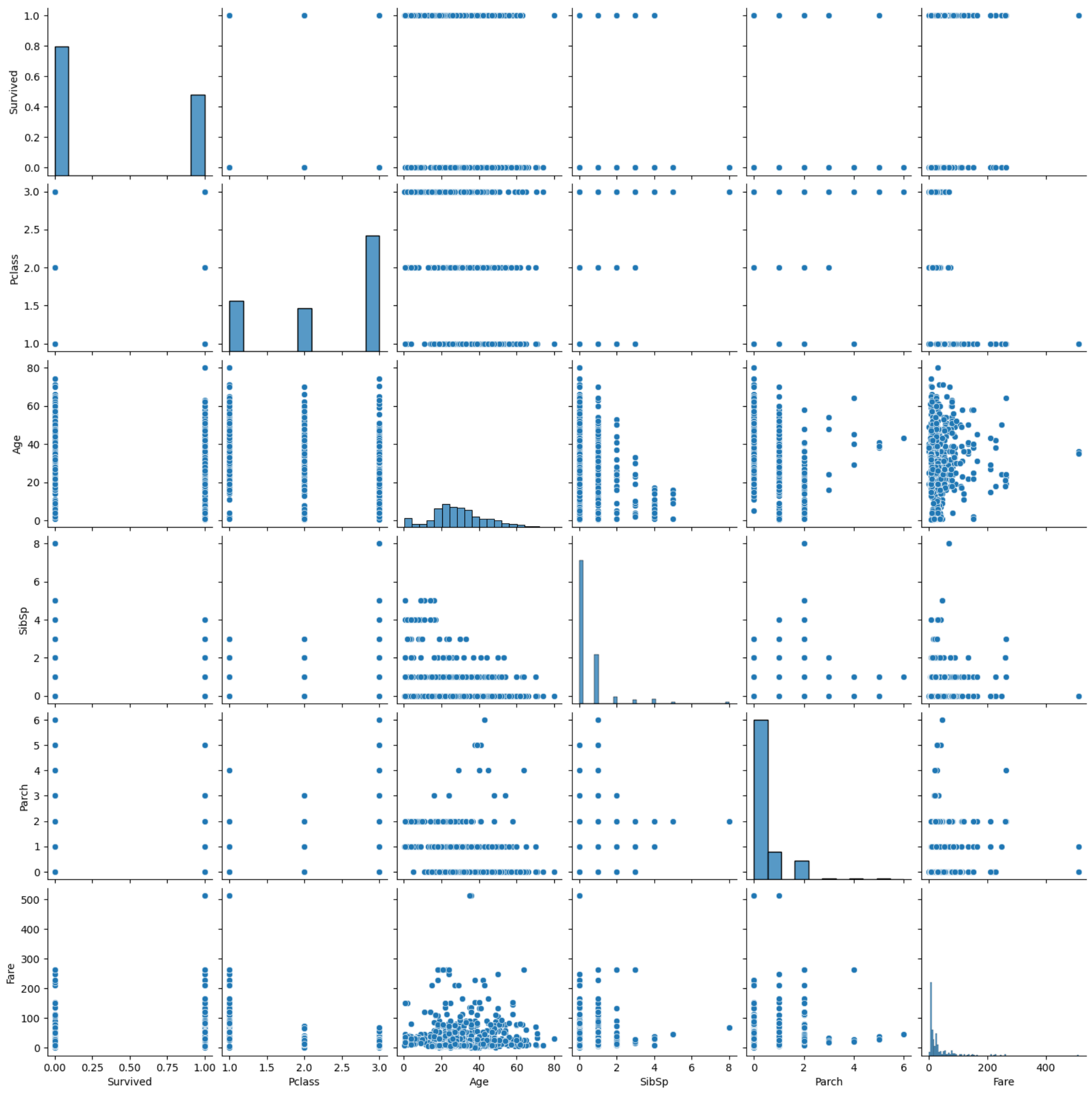
```
[31]: numeric_df=df.select_dtypes(exclude=[object])
      corr=numeric_df.corr()
      sns.heatmap(corr,annot=True)
```

```
[31]: <Axes: >
```



[32]:
```
sns.pairplot(df)
```

[32]:
```
<seaborn.axisgrid.PairGrid at 0x79bb45643c40>
```


Survived
Pclass
Age
SibSp
Parch
Fare