03_Seaborn_RegLog

December 10, 2019

1 Seaborn - Iris Classification

[0]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id
=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redire
ct_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20http
s%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.c
om%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.reado
nly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
ůůůůůůůůůů
Mounted at /content/drive
```

Move to the correct folder

[0]:
```
cd /content/drive/My Drive/03-seaborn-classification
```

```
/content/drive/My Drive/03-seaborn-classification
```

[0]:
```
!pwd
```

```
/content/drive/My Drive/03-seaborn-classification
```

2 Iris Data from Pandas

[0]:
```
import numpy as np    # array
import pandas as pd   # dataframe
import seaborn as sns   #
import matplotlib.pyplot as plt
sns.set(color_codes=True)
%matplotlib inline
```

[0]:
```
from IPython.display import Image
Image(filename='iris.png')
```

[0]:

[0]:
```
df = pd.read_csv('iris.data') # ! relative path !!!
```

[0]:
```
df.head() #  see first 5 rows
```

[0]:

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa

Remove header

[0]:
```
df = pd.read_csv('iris.data', header=None)
df.head()
```

[0]:

	0	1	2	3	4
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Attribute Information from UCI Dataset
https://archive.ics.uci.edu/ml/datasets/iris

[0]:
```
col_name =   ['sepal length', 'sepal width', 'petal length', 'petal width',␣
↪'class']
```

[0]:
```
df.columns = col_name
```

[0]:
```
df.shape
```

[0]: (150, 5)

[0]:
```
df.head()
```

[0]:

	sepal length	sepal width	petal length	petal width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

[0]:

```
df.ix[:,0] # try to print all rows
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: FutureWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ix-indexer-
is-deprecated
  """Entry point for launching an IPython kernel.
```

[0]:

```
0      5.1
1      4.9
2      4.7
3      4.6
4      5.0
      ...
145    6.7
146    6.3
147    6.5
148    6.2
149    5.9
Name: sepal length, Length: 150, dtype: float64
```

3 Iris Data from Seaborn

[0]:

```
iris = sns.load_dataset('iris')
iris.head()
```

[0]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

[0]:

```
iris.tail()
```

[0]:

```
df.describe()
```

[0]:

	sepal length	sepal width	petal length	petal width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

[0]:
```
iris.describe()
```

[0]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

[0]:
```
print(iris.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
sepal_length    150 non-null float64
sepal_width     150 non-null float64
petal_length    150 non-null float64
petal_width     150 non-null float64
species         150 non-null object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
```

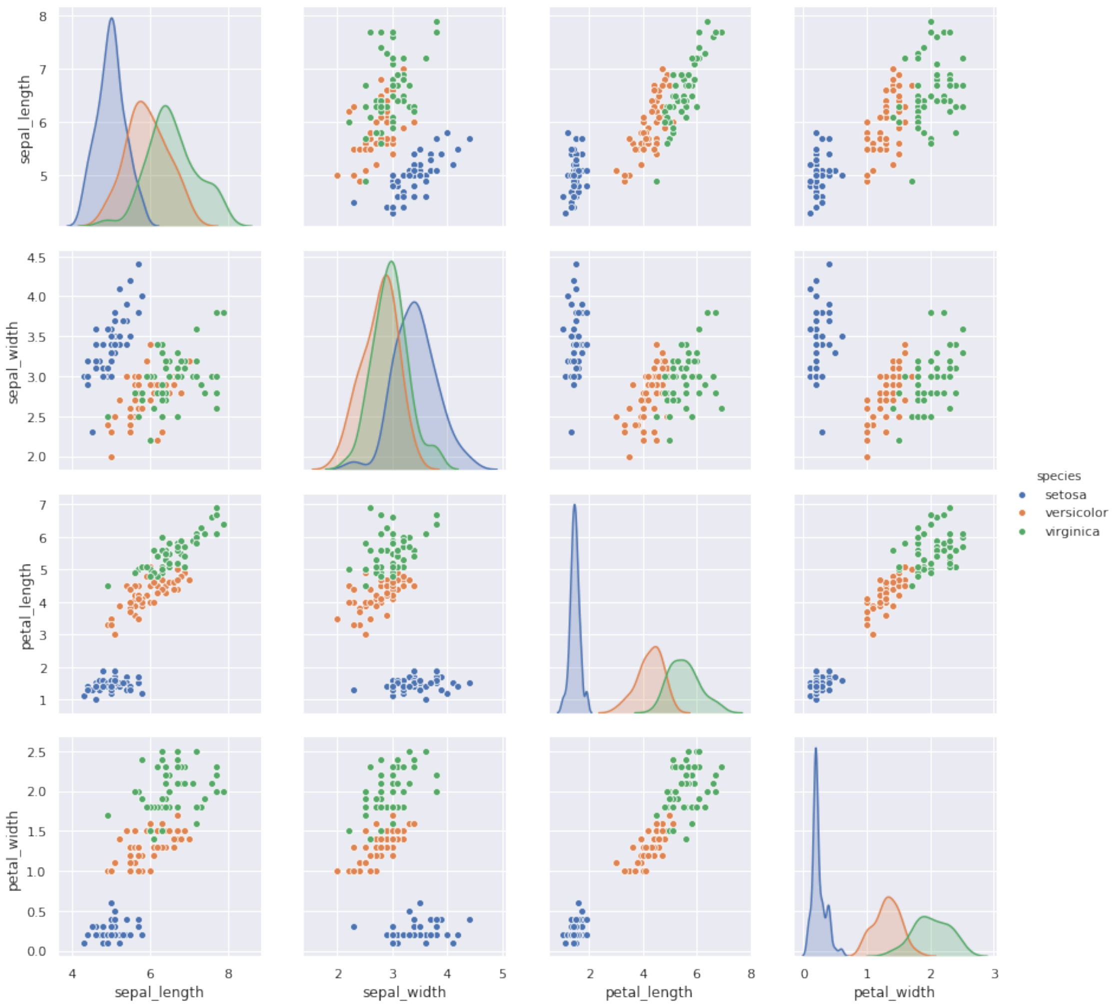
[0]:
```
print(iris.groupby('species').size())
```

```
species
setosa        50
versicolor    50
virginica     50
dtype: int64
```

4 Visualisation

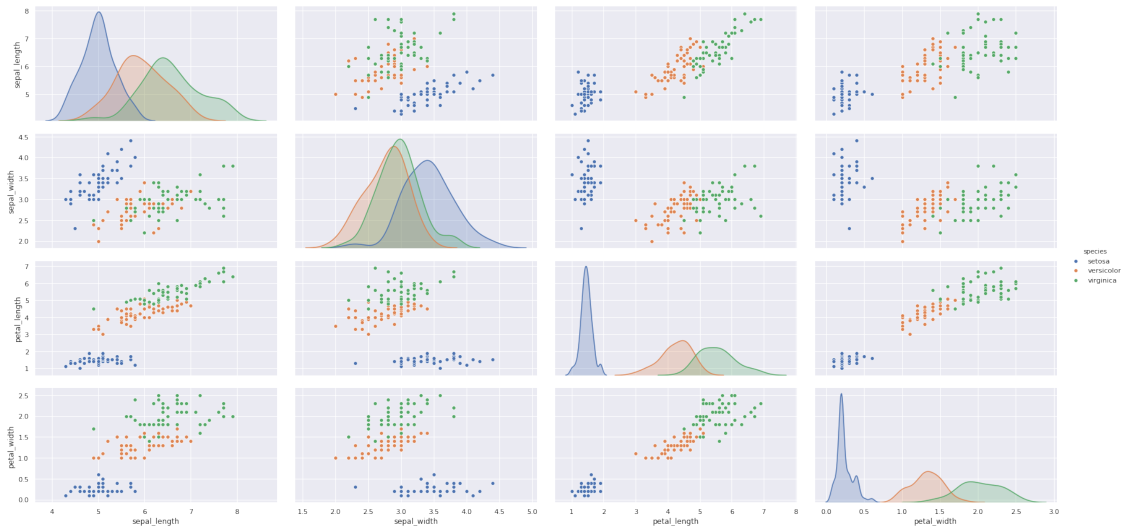
[0]:
```
sns.pairplot(iris, hue='species', size=3, aspect=1);
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/axisgrid.py:2065: UserWarning:
The `size` parameter has been renamed to `height`; pleaes update your code.
  warnings.warn(msg, UserWarning)
```

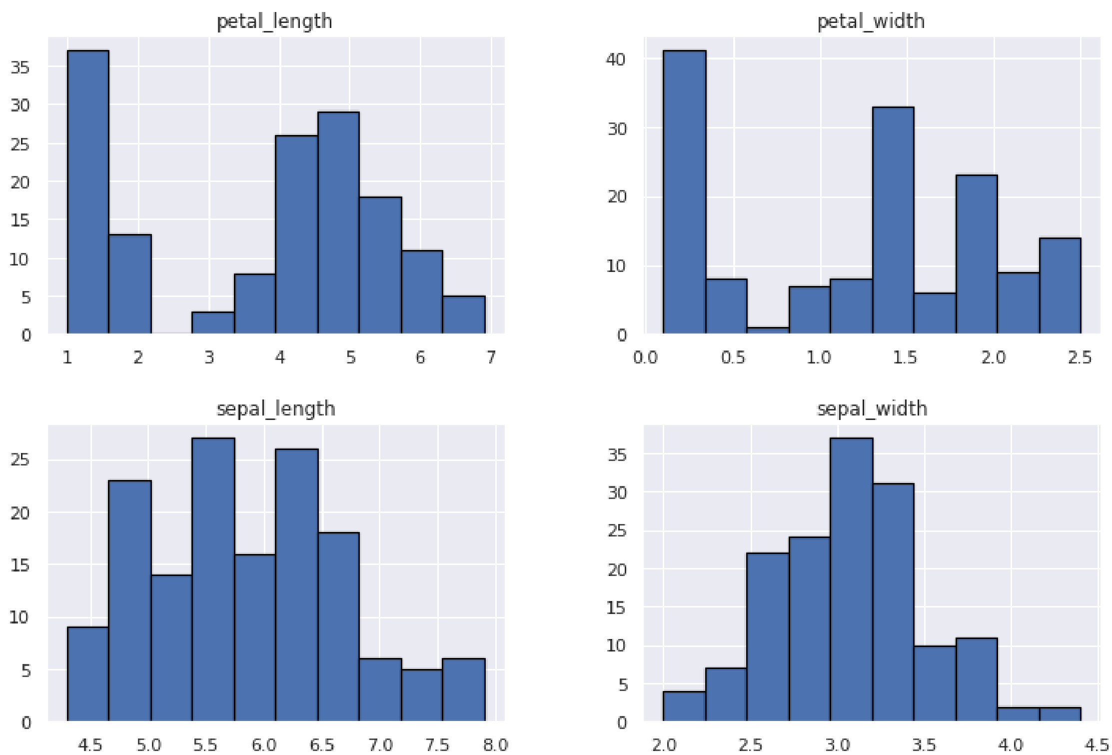


[0]:
```
sns.pairplot(iris, hue='species', size=3, aspect=2);
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/axisgrid.py:2065: UserWarning:
The `size` parameter has been renamed to `height`; pleaes update your code.
  warnings.warn(msg, UserWarning)
```

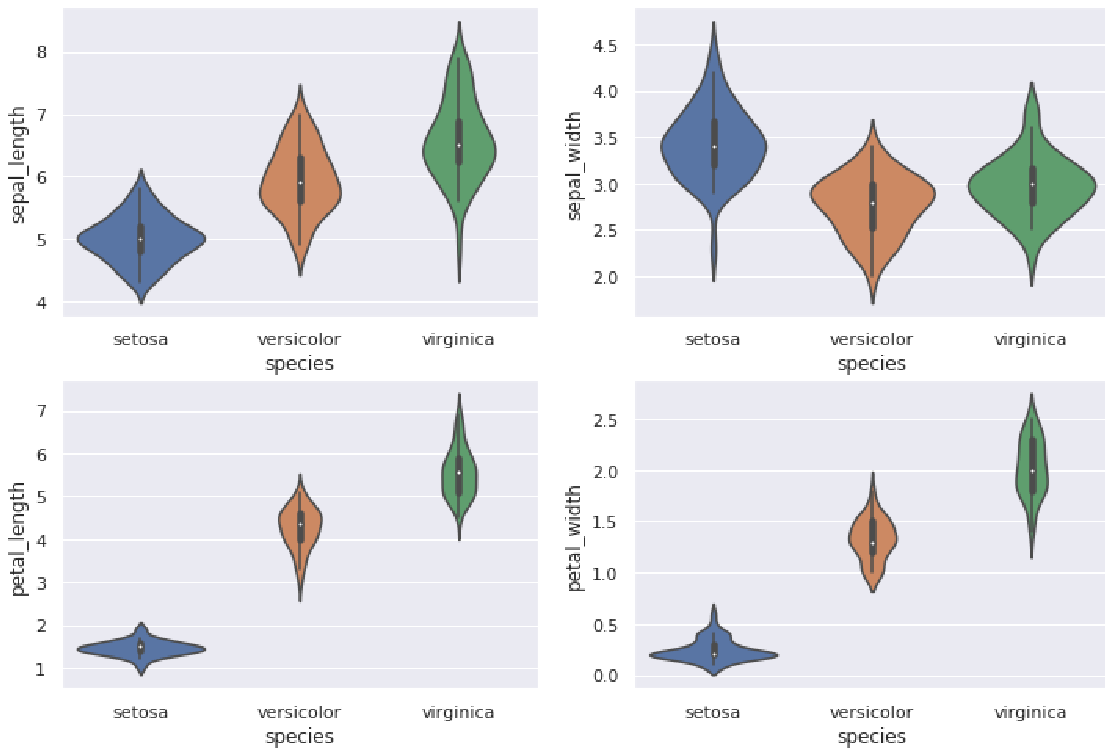


[0]:
```
iris.hist(edgecolor='black', linewidth=1.2, figsize=(12,8));
plt.show();
```

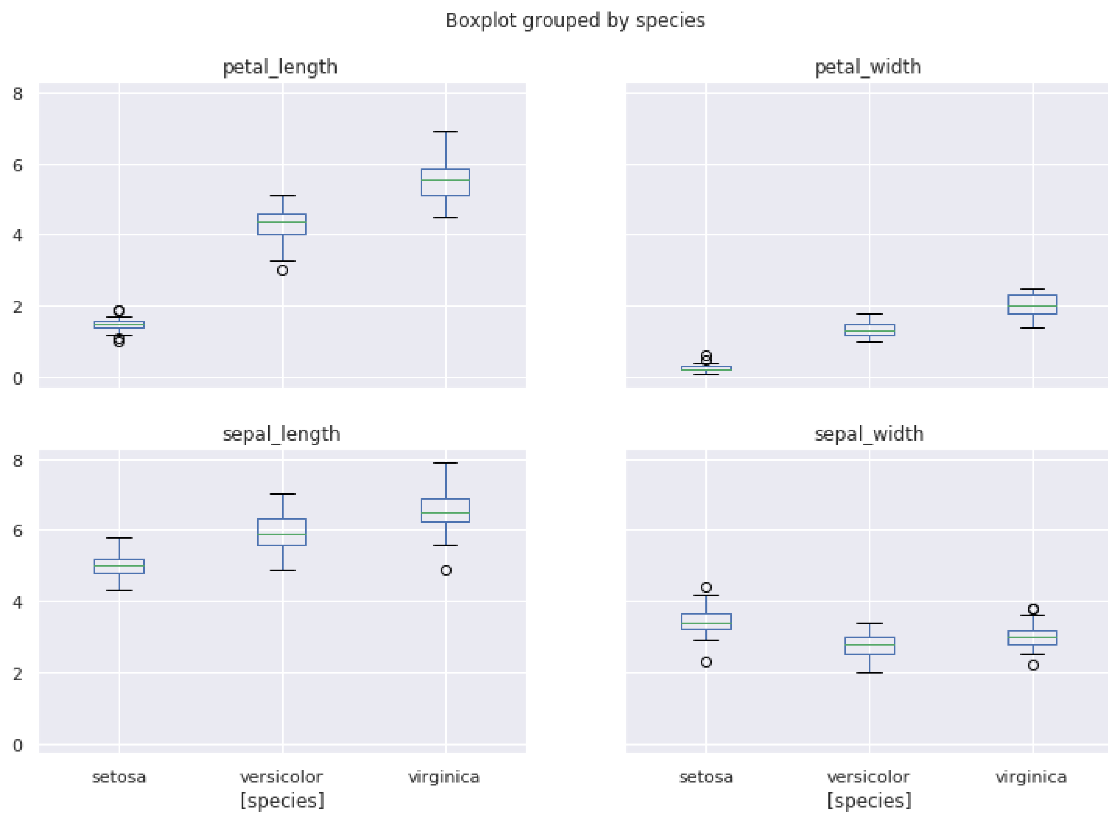


[0]:
```
plt.figure(figsize=(12,8));
plt.subplot(2,2,1)
sns.violinplot(x='species', y='sepal_length', data=iris)
```

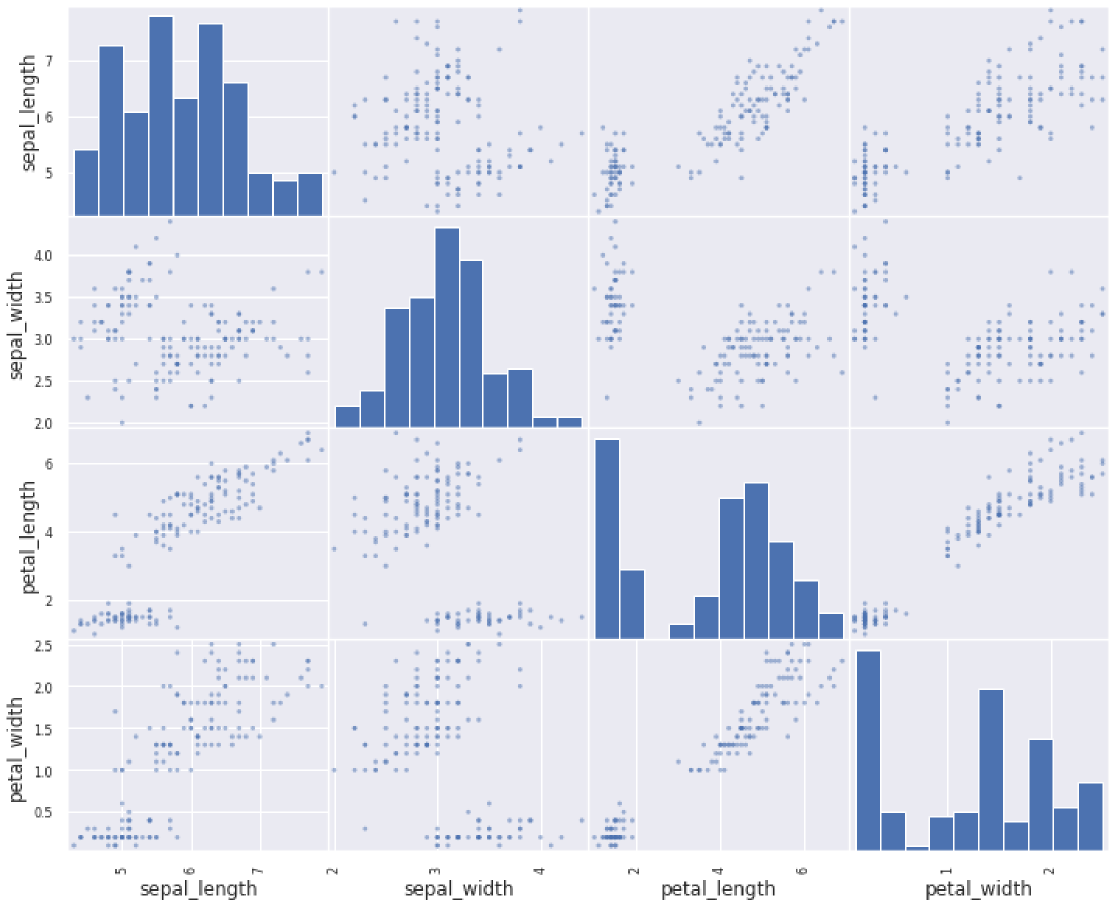
```
plt.subplot(2,2,2)
sns.violinplot(x='species', y='sepal_width', data=iris)
plt.subplot(2,2,3)
sns.violinplot(x='species', y='petal_length', data=iris)
plt.subplot(2,2,4)
sns.violinplot(x='species', y='petal_width', data=iris);
```



[0]:
```
iris.boxplot(by='species', figsize=(12,8));
```



[0]:
```
pd.plotting.scatter_matrix(iris, figsize=(12,10))
plt.show()
```



5 scikit-learn

url = http://scikit-learn.org/stable/

[0]:
```
%%html
<iframe src="https://scikit-learn.org/stable/" width="1200" height="1000"></
↪iframe>
```

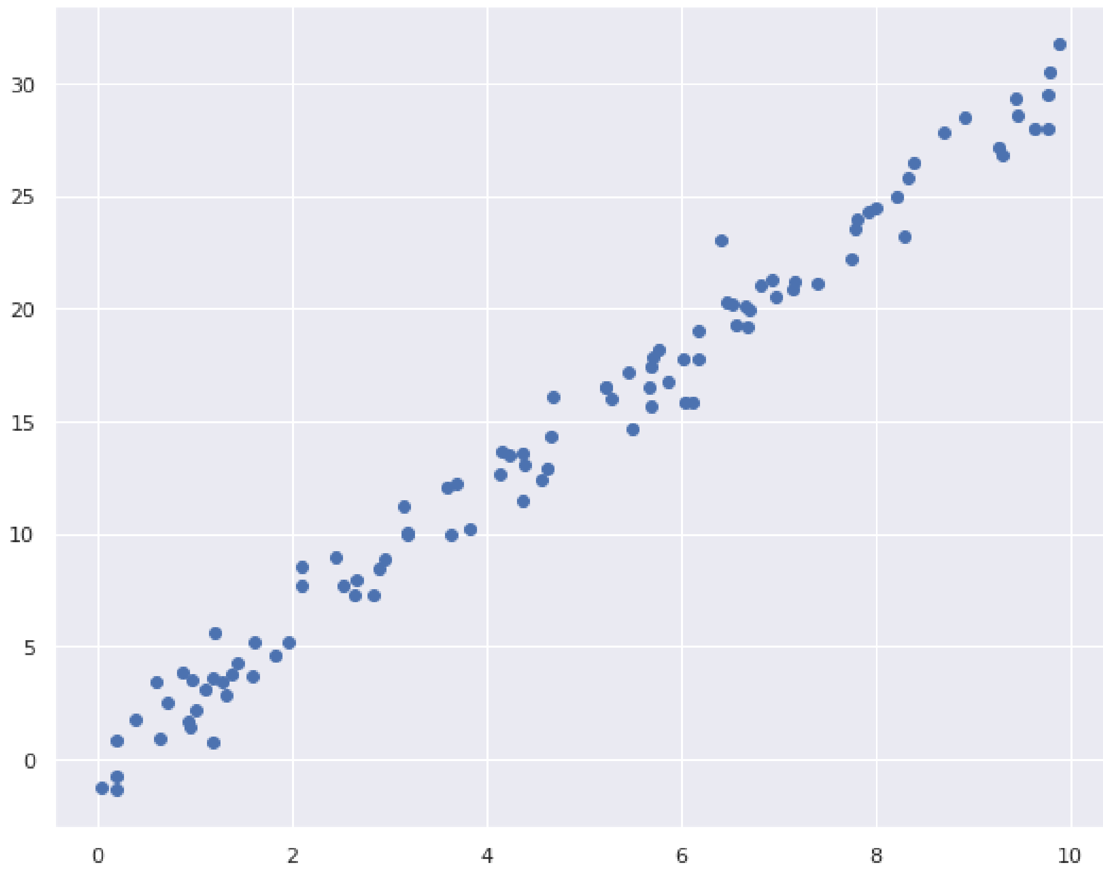
```
<IPython.core.display.HTML object>
```

6 Supervised Learning: Simple Linear Regression

[0]:
```
import numpy as np
generate_random = np.random.RandomState(0)
x = 10 * generate_random.rand(100)
```

[0]:
```
y = 3 * x + np.random.randn(100)
```

[0]:
```
plt.figure(figsize = (10, 8))
plt.scatter(x, y);
```



6.1 Step 1. Choose a class of model

[0]:
```
from sklearn.linear_model import LinearRegression
```

6.2 Step 2. Choose model hyperparameters

[0]:
```
model = LinearRegression(fit_intercept=True)
```

[0]:
```
model
```

[0]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

6.3 Step 3. Arrage data into features matrix and target array

[0]:
```
x
```

[0]:
```
array([5.48813504, 7.15189366, 6.02763376, 5.44883183, 4.23654799,
       6.45894113, 4.37587211, 8.91773001, 9.63662761, 3.83441519,
       7.91725038, 5.2889492 , 5.68044561, 9.25596638, 0.71036058,
       0.871293  , 0.20218397, 8.32619846, 7.78156751, 8.70012148,
       9.78618342, 7.99158564, 4.61479362, 7.80529176, 1.18274426,
       6.39921021, 1.43353287, 9.44668917, 5.21848322, 4.1466194 ,
       2.64555612, 7.74233689, 4.56150332, 5.68433949, 0.187898  ,
       6.17635497, 6.12095723, 6.16933997, 9.43748079, 6.81820299,
       3.59507901, 4.37031954, 6.97631196, 0.60225472, 6.66766715,
       6.7063787 , 2.10382561, 1.28926298, 3.15428351, 3.63710771,
       5.7019677 , 4.38601513, 9.88373838, 1.02044811, 2.08876756,
       1.61309518, 6.53108325, 2.53291603, 4.66310773, 2.44425592,
       1.58969584, 1.10375141, 6.56329589, 1.38182951, 1.96582362,
       3.68725171, 8.2099323 , 0.97101276, 8.37944907, 0.96098408,
       9.76459465, 4.68651202, 9.76761088, 6.0484552 , 7.39263579,
       0.39187792, 2.82806963, 1.20196561, 2.96140198, 1.18727719,
       3.17983179, 4.14262995, 0.64147496, 6.92472119, 5.66601454,
       2.65389491, 5.23248053, 0.93940511, 5.75946496, 9.29296198,
       3.18568952, 6.6741038 , 1.31797862, 7.16327204, 2.89406093,
       1.83191362, 5.86512935, 0.20107546, 8.28940029, 0.04695476])
```

[0]:
```
X = x.reshape(-1, 1)
X.shape
```

[0]:
```
(100, 1)
```

[0]:
```
X
```

[0]:
```
array([[5.48813504],
       [7.15189366],
       [6.02763376],
       [5.44883183],
       [4.23654799],
       [6.45894113],
       [4.37587211],
       [8.91773001],
       [9.63662761],
       [3.83441519],
       [7.91725038],
       [5.2889492 ],
       [5.68044561],
       [9.25596638],
       [0.71036058],
       [0.871293  ],
       [0.20218397],
       [8.32619846],
```

```
[7.78156751],
[8.70012148],
[9.78618342],
[7.99158564],
[4.61479362],
[7.80529176],
[1.18274426],
[6.39921021],
[1.43353287],
[9.44668917],
[5.21848322],
[4.1466194 ],
[2.64555612],
[7.74233689],
[4.56150332],
[5.68433949],
[0.187898  ],
[6.17635497],
[6.12095723],
[6.16933997],
[9.43748079],
[6.81820299],
[3.59507901],
[4.37031954],
[6.97631196],
[0.60225472],
[6.66766715],
[6.7063787 ],
[2.10382561],
[1.28926298],
[3.15428351],
[3.63710771],
[5.7019677 ],
[4.38601513],
[9.88373838],
[1.02044811],
[2.08876756],
[1.61309518],
[6.53108325],
[2.53291603],
[4.66310773],
[2.44425592],
[1.58969584],
[1.10375141],
[6.56329589],
[1.38182951],
[1.96582362],
```

```
       [3.68725171],
       [8.2099323 ],
       [0.97101276],
       [8.37944907],
       [0.96098408],
       [9.76459465],
       [4.68651202],
       [9.76761088],
       [6.0484552 ],
       [7.39263579],
       [0.39187792],
       [2.82806963],
       [1.20196561],
       [2.96140198],
       [1.18727719],
       [3.17983179],
       [4.14262995],
       [0.64147496],
       [6.92472119],
       [5.66601454],
       [2.65389491],
       [5.23248053],
       [0.93940511],
       [5.75946496],
       [9.29296198],
       [3.18568952],
       [6.6741038 ],
       [1.31797862],
       [7.16327204],
       [2.89406093],
       [1.83191362],
       [5.86512935],
       [0.20107546],
       [8.28940029],
       [0.04695476]])
```

6.4 Step 4. Fit model to data

[0]:
```
model.fit(X, y)
```

[0]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

[0]:
```
model.coef_
```

[0]:
```
array([3.05175136])
```

[0]:
```
model.intercept_
```

[0]:
```
-0.24158591968593868
```

6.5 Step 5. Apply trained model to new data

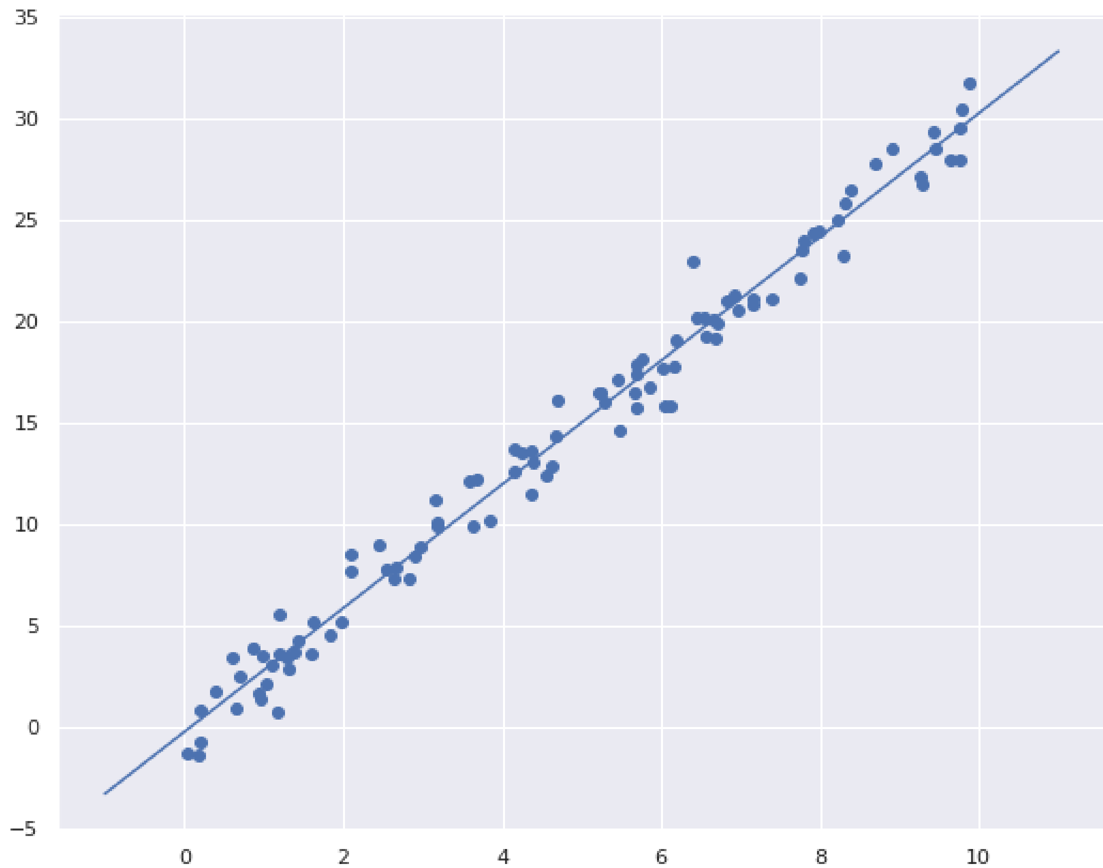
[0]:
```
x_fit = np.linspace(-1, 11)
```

[0]:
```
X_fit = x_fit.reshape(-1,1)
```

[0]:
```
y_fit = model.predict(X_fit)
```

6.6 Visualise

[0]:
```
plt.figure(figsize = (10, 8))
plt.scatter(x, y)
plt.plot(x_fit, y_fit);
```



7 Logistic regression

[0]:
```
%%html
<iframe src="https://static.javatpoint.com/tutorial/machine-learning/images/
↪logistic-regression-in-machine-learning.png" width="600" height="300"></
↪iframe>
```

```
<IPython.core.display.HTML object>
```

S(z)=1/1+ez

[0]:
```

```

[0]:
```
from sklearn.datasets import load_iris
iris = load_iris()
type(iris)
```

[0]:
```
sklearn.utils.Bunch
```

[0]:
```
# each row represents each sample
# each column represents the features
print(iris.data)
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.7 1.5 0.2]
 [4.8 3.4 1.6 0.2]
 [4.8 3.  1.4 0.1]
 [4.3 3.  1.1 0.1]
 [5.8 4.  1.2 0.2]
 [5.7 4.4 1.5 0.4]
 [5.4 3.9 1.3 0.4]
 [5.1 3.5 1.4 0.3]
 [5.7 3.8 1.7 0.3]
 [5.1 3.8 1.5 0.3]
 [5.4 3.4 1.7 0.2]
 [5.1 3.7 1.5 0.4]
 [4.6 3.6 1.  0.2]
 [5.1 3.3 1.7 0.5]
 [4.8 3.4 1.9 0.2]
```

```
[5.  3.  1.6 0.2]
[5.  3.4 1.6 0.4]
[5.2 3.5 1.5 0.2]
[5.2 3.4 1.4 0.2]
[4.7 3.2 1.6 0.2]
[4.8 3.1 1.6 0.2]
[5.4 3.4 1.5 0.4]
[5.2 4.1 1.5 0.1]
[5.5 4.2 1.4 0.2]
[4.9 3.1 1.5 0.2]
[5.  3.2 1.2 0.2]
[5.5 3.5 1.3 0.2]
[4.9 3.6 1.4 0.1]
[4.4 3.  1.3 0.2]
[5.1 3.4 1.5 0.2]
[5.  3.5 1.3 0.3]
[4.5 2.3 1.3 0.3]
[4.4 3.2 1.3 0.2]
[5.  3.5 1.6 0.6]
[5.1 3.8 1.9 0.4]
[4.8 3.  1.4 0.3]
[5.1 3.8 1.6 0.2]
[4.6 3.2 1.4 0.2]
[5.3 3.7 1.5 0.2]
[5.  3.3 1.4 0.2]
[7.  3.2 4.7 1.4]
[6.4 3.2 4.5 1.5]
[6.9 3.1 4.9 1.5]
[5.5 2.3 4.  1.3]
[6.5 2.8 4.6 1.5]
[5.7 2.8 4.5 1.3]
[6.3 3.3 4.7 1.6]
[4.9 2.4 3.3 1. ]
[6.6 2.9 4.6 1.3]
[5.2 2.7 3.9 1.4]
[5.  2.  3.5 1. ]
[5.9 3.  4.2 1.5]
[6.  2.2 4.  1. ]
[6.1 2.9 4.7 1.4]
[5.6 2.9 3.6 1.3]
[6.7 3.1 4.4 1.4]
[5.6 3.  4.5 1.5]
[5.8 2.7 4.1 1. ]
[6.2 2.2 4.5 1.5]
[5.6 2.5 3.9 1.1]
[5.9 3.2 4.8 1.8]
[6.1 2.8 4.  1.3]
[6.3 2.5 4.9 1.5]
```

```
[6.1 2.8 4.7 1.2]
[6.4 2.9 4.3 1.3]
[6.6 3.  4.4 1.4]
[6.8 2.8 4.8 1.4]
[6.7 3.  5.  1.7]
[6.  2.9 4.5 1.5]
[5.7 2.6 3.5 1. ]
[5.5 2.4 3.8 1.1]
[5.5 2.4 3.7 1. ]
[5.8 2.7 3.9 1.2]
[6.  2.7 5.1 1.6]
[5.4 3.  4.5 1.5]
[6.  3.4 4.5 1.6]
[6.7 3.1 4.7 1.5]
[6.3 2.3 4.4 1.3]
[5.6 3.  4.1 1.3]
[5.5 2.5 4.  1.3]
[5.5 2.6 4.4 1.2]
[6.1 3.  4.6 1.4]
[5.8 2.6 4.  1.2]
[5.  2.3 3.3 1. ]
[5.6 2.7 4.2 1.3]
[5.7 3.  4.2 1.2]
[5.7 2.9 4.2 1.3]
[6.2 2.9 4.3 1.3]
[5.1 2.5 3.  1.1]
[5.7 2.8 4.1 1.3]
[6.3 3.3 6.  2.5]
[5.8 2.7 5.1 1.9]
[7.1 3.  5.9 2.1]
[6.3 2.9 5.6 1.8]
[6.5 3.  5.8 2.2]
[7.6 3.  6.6 2.1]
[4.9 2.5 4.5 1.7]
[7.3 2.9 6.3 1.8]
[6.7 2.5 5.8 1.8]
[7.2 3.6 6.1 2.5]
[6.5 3.2 5.1 2. ]
[6.4 2.7 5.3 1.9]
[6.8 3.  5.5 2.1]
[5.7 2.5 5.  2. ]
[5.8 2.8 5.1 2.4]
[6.4 3.2 5.3 2.3]
[6.5 3.  5.5 1.8]
[7.7 3.8 6.7 2.2]
[7.7 2.6 6.9 2.3]
[6.  2.2 5.  1.5]
[6.9 3.2 5.7 2.3]
```

```
 [5.6 2.8 4.9 2. ]
 [7.7 2.8 6.7 2. ]
 [6.3 2.7 4.9 1.8]
 [6.7 3.3 5.7 2.1]
 [7.2 3.2 6.  1.8]
 [6.2 2.8 4.8 1.8]
 [6.1 3.  4.9 1.8]
 [6.4 2.8 5.6 2.1]
 [7.2 3.  5.8 1.6]
 [7.4 2.8 6.1 1.9]
 [7.9 3.8 6.4 2. ]
 [6.4 2.8 5.6 2.2]
 [6.3 2.8 5.1 1.5]
 [6.1 2.6 5.6 1.4]
 [7.7 3.  6.1 2.3]
 [6.3 3.4 5.6 2.4]
 [6.4 3.1 5.5 1.8]
 [6.  3.  4.8 1.8]
 [6.9 3.1 5.4 2.1]
 [6.7 3.1 5.6 2.4]
 [6.9 3.1 5.1 2.3]
 [5.8 2.7 5.1 1.9]
 [6.8 3.2 5.9 2.3]
 [6.7 3.3 5.7 2.5]
 [6.7 3.  5.2 2.3]
 [6.3 2.5 5.  1.9]
 [6.5 3.  5.2 2. ]
 [6.2 3.4 5.4 2.3]
 [5.9 3.  5.1 1.8]]
```

[0]:
```python
# print the names of the four features
print(iris.feature_names)
```

```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width
(cm)']
```

[0]:
```python
# 0, 1, and 2 represent different species
print(iris.target)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

[0]:
```
# print the encoding scheme for species: 0 = setosa, 1 = versicolor, 2 =␣
↪virginica
print(iris.target_names)
```

```
['setosa' 'versicolor' 'virginica']
```

[0]:
```
# check the types of the features and response
print(type(iris.data))
print(type(iris.target))
```

```
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

[0]:
```
# check the shape of the features (first dimension = number of observations,␣
↪second dimensions = number of features)
print(iris.data.shape)
```

```
(150, 4)
```

[0]:
```
# check the shape of the response (single dimension matching the number of␣
↪observations)
print(iris.target.shape)
```

```
(150,)
```

[0]:
```
# store feature matrix in "X"
X = iris.data
# store response vector in "y"
y = iris.target
```

7.1 from pandas data load from our storage

[0]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

Start preparing the training data set by storing all of the independent variables/columns/features into a variable called 'X', and store the independent variable/target into a variable called 'y'.

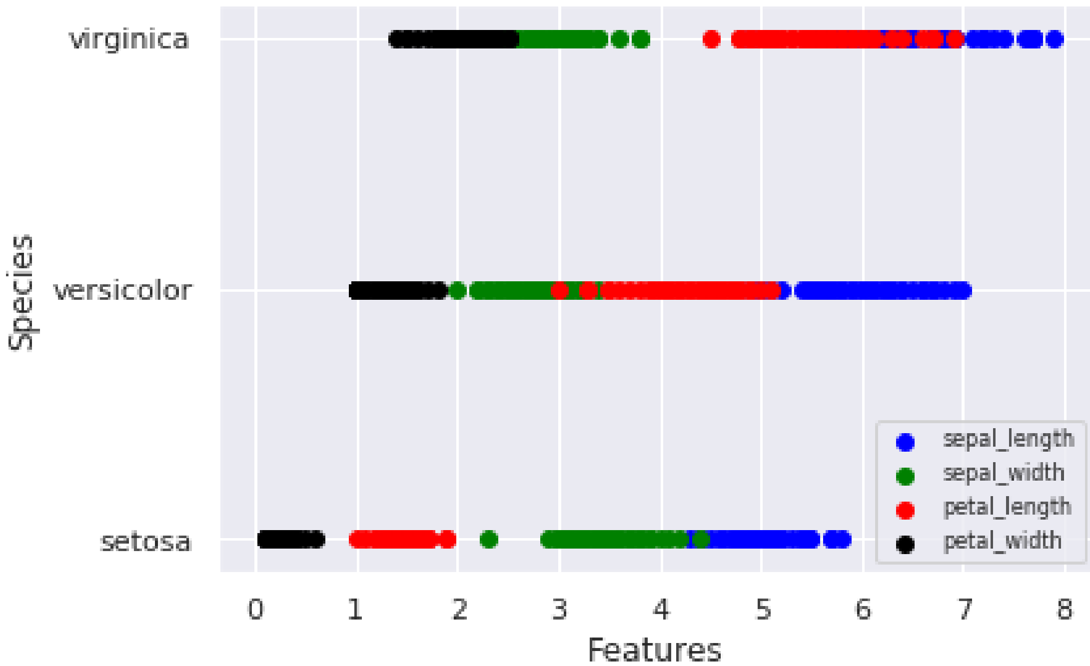
[0]:
```
iris = sns.load_dataset('iris')
iris.head()
```

[0]:
```
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
```

1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

[0]:
```
#Prepare the training set

# X = feature values, all the columns except the last column
X = iris.iloc[:, :-1]

# y = target values, last column of the data frame
y = iris.iloc[:, -1]
```

[0]:
```
# Plot the relation of each feature with each species

plt.xlabel('Features')
plt.ylabel('Species')

pltX = iris.loc[:, 'sepal_length']
pltY = iris.loc[:,'species']
plt.scatter(pltX, pltY, color='blue', label='sepal_length')

pltX = iris.loc[:, 'sepal_width']
pltY = iris.loc[:,'species']
plt.scatter(pltX, pltY, color='green', label='sepal_width')

pltX = iris.loc[:, 'petal_length']
pltY = iris.loc[:,'species']
plt.scatter(pltX, pltY, color='red', label='petal_length')

pltX = iris.loc[:, 'petal_width']
pltY = iris.loc[:,'species']
plt.scatter(pltX, pltY, color='black', label='petal_width')

plt.legend(loc=4, prop={'size':8})
plt.show()
```



7.2 Splitting dataset

Split the data into 80% training and 20 % testing by using the method train_test_split() from the sklearn.model_selection library, and store the data into x_train, x_test, y_train, and y_test.

[0]:
```
#Split the data into 80% training and 20% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=667)
print(y_test)
```

```
116     virginica
15         setosa
73     versicolor
80     versicolor
93     versicolor
51     versicolor
129     virginica
141     virginica
94     versicolor
123     virginica
12         setosa
106     virginica
69     versicolor
24         setosa
125     virginica
```

```
148     virginica
6          setosa
13         setosa
42         setosa
132     virginica
60     versicolor
68     versicolor
1          setosa
7          setosa
102     virginica
121     virginica
38         setosa
97     versicolor
98     versicolor
34         setosa
Name: species, dtype: object
```

[0]:
```

```

[0]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train) #Training the model
```

```
      ␣
↪---------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
↪last)

       <ipython-input-111-6cedee946f9a> in <module>()
         1 from sklearn.linear_model import LinearRegression
         2 model = LinearRegression()
   ----> 3 model.fit(X_train, y_train) #Training the model


       /usr/local/lib/python3.6/dist-packages/sklearn/linear_model/base.py in␣
↪fit(self, X, y, sample_weight)
       461         n_jobs_ = self.n_jobs
       462         X, y = check_X_y(X, y, accept_sparse=['csr', 'csc', 'coo'],
   --> 463                          y_numeric=True, multi_output=True)
       464
       465         if sample_weight is not None and np.
↪atleast_1d(sample_weight).ndim > 1:
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py in␣
↪check_X_y(X, y, accept_sparse, accept_large_sparse, dtype, order, copy,␣
↪force_all_finite, ensure_2d, allow_nd, multi_output, ensure_min_samples,␣
↪ensure_min_features, y_numeric, warn_on_dtype, estimator)
    725         _assert_all_finite(y)
    726     if y_numeric and y.dtype.kind == 'O':
--> 727         y = y.astype(np.float64)
    728
    729     check_consistent_length(X, y)


ValueError: could not convert string to float: 'setosa'
```

7.3 Why Error?

[0]:
```
#Train the model with Logistic Regression
model = LogisticRegression()
model.fit(X_train, y_train) #Training the model
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/logistic.py:432:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/logistic.py:469:
FutureWarning: Default multi_class will be changed to 'auto' in 0.22. Specify
the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

[0]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

[0]:
```
X_test.shape
```

[0]:
```
(30, 4)
```

[0]:
```
model.predict_proba(X_test)
```

[0]:
```
array([[1.23579374e-03, 3.67458143e-01, 6.31306063e-01],
       [9.51101519e-01, 4.88928812e-02, 5.59993021e-06],
       [1.17452600e-02, 6.90571943e-01, 2.97682797e-01],
       [4.59260796e-02, 7.66395975e-01, 1.87677946e-01],
       [1.09027017e-01, 7.54849712e-01, 1.36123271e-01],
       [4.53318792e-02, 6.92729993e-01, 2.61938128e-01],
       [1.07574006e-03, 5.12604766e-01, 4.86319493e-01],
       [3.33105672e-03, 2.64563090e-01, 7.32105853e-01],
```

```
       [2.52859825e-02, 6.29085962e-01, 3.45628055e-01],
       [3.11514201e-03, 4.03177244e-01, 5.93707614e-01],
       [7.87298557e-01, 2.12638646e-01, 6.27965091e-05],
       [2.94180473e-03, 3.01812203e-01, 6.95245993e-01],
       [4.55117155e-02, 7.74315514e-01, 1.80172771e-01],
       [8.20547288e-01, 1.79310123e-01, 1.42588305e-04],
       [7.31523900e-04, 4.13471419e-01, 5.85797057e-01],
       [1.65378963e-03, 1.67605924e-01, 8.30740287e-01],
       [8.78269981e-01, 1.21648662e-01, 8.13569024e-05],
       [8.26752945e-01, 1.73176282e-01, 7.07732696e-05],
       [8.51350218e-01, 1.48557961e-01, 9.18202940e-05],
       [4.52412755e-04, 2.88579704e-01, 7.10967884e-01],
       [3.78424315e-02, 7.32388890e-01, 2.29768678e-01],
       [4.68238900e-03, 5.68381169e-01, 4.26936442e-01],
       [7.99941002e-01, 1.99989669e-01, 6.93282051e-05],
       [8.49494690e-01, 1.50462786e-01, 4.25248372e-05],
       [4.64189251e-04, 3.38545039e-01, 6.60990772e-01],
       [1.99546947e-03, 2.53960290e-01, 7.44044241e-01],
       [8.22057233e-01, 1.77821379e-01, 1.21387225e-04],
       [4.25903639e-02, 7.68357741e-01, 1.89051895e-01],
       [2.80913836e-01, 6.49535122e-01, 6.95510420e-02],
       [8.07016690e-01, 1.92907886e-01, 7.54244952e-05]])
```

[0]:
```
# Array index
## 0 = setosa, 1 = versicolor, 2 = virginica
```

[0]:
```
#Test the model
y_predict  = model.predict(X_test)
print(y_predict )# printing predictions
```

```
['virginica' 'setosa' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'virginica' 'versicolor' 'virginica' 'setosa' 'virginica'
 'versicolor' 'setosa' 'virginica' 'virginica' 'setosa' 'setosa' 'setosa'
 'virginica' 'versicolor' 'versicolor' 'setosa' 'setosa' 'virginica'
 'virginica' 'setosa' 'versicolor' 'versicolor' 'setosa']
```

[0]:
```
type(y_predict)
```

[0]:
```
numpy.ndarray
```

[0]:
```
type(y_test)
```

[0]:
```
pandas.core.series.Series
```

[0]:
```
y_test.values
```

[0]:
```
array(['virginica', 'setosa', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'virginica', 'virginica', 'versicolor', 'virginica',
       'setosa', 'virginica', 'versicolor', 'setosa', 'virginica',
       'virginica', 'setosa', 'setosa', 'setosa', 'virginica',
       'versicolor', 'versicolor', 'setosa', 'setosa', 'virginica',
```

```
       'virginica', 'setosa', 'versicolor', 'versicolor', 'setosa'],
      dtype=object)
```

[0]:
```
comparison = pd.DataFrame(data = [y_predict,y_test])
comparison.T
```

[0]:

	0	1
0	virginica	virginica
1	setosa	setosa
2	versicolor	versicolor
3	versicolor	versicolor
4	versicolor	versicolor
5	versicolor	versicolor
6	versicolor	virginica
7	virginica	virginica
8	versicolor	versicolor
9	virginica	virginica
10	setosa	setosa
11	virginica	virginica
12	versicolor	versicolor
13	setosa	setosa
14	virginica	virginica
15	virginica	virginica
16	setosa	setosa
17	setosa	setosa
18	setosa	setosa
19	virginica	virginica
20	versicolor	versicolor
21	versicolor	versicolor
22	setosa	setosa
23	setosa	setosa
24	virginica	virginica
25	virginica	virginica
26	setosa	setosa
27	versicolor	versicolor
28	versicolor	versicolor
29	setosa	setosa

[0]:
```

```

8 Exercise

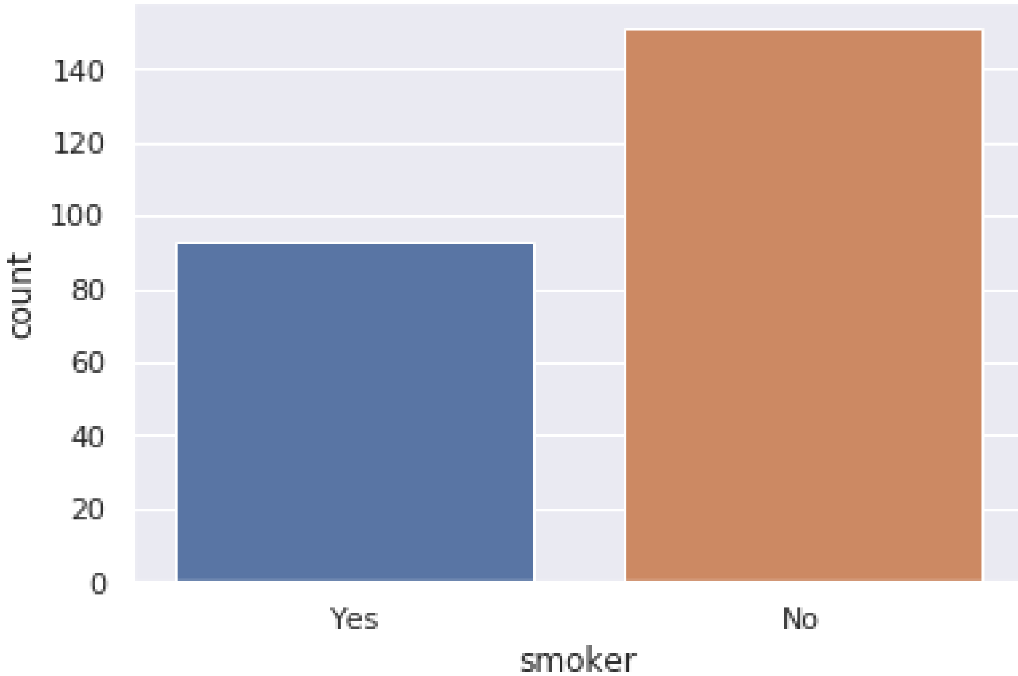
[0]:
```
# import all libraries needed
```

[0]:

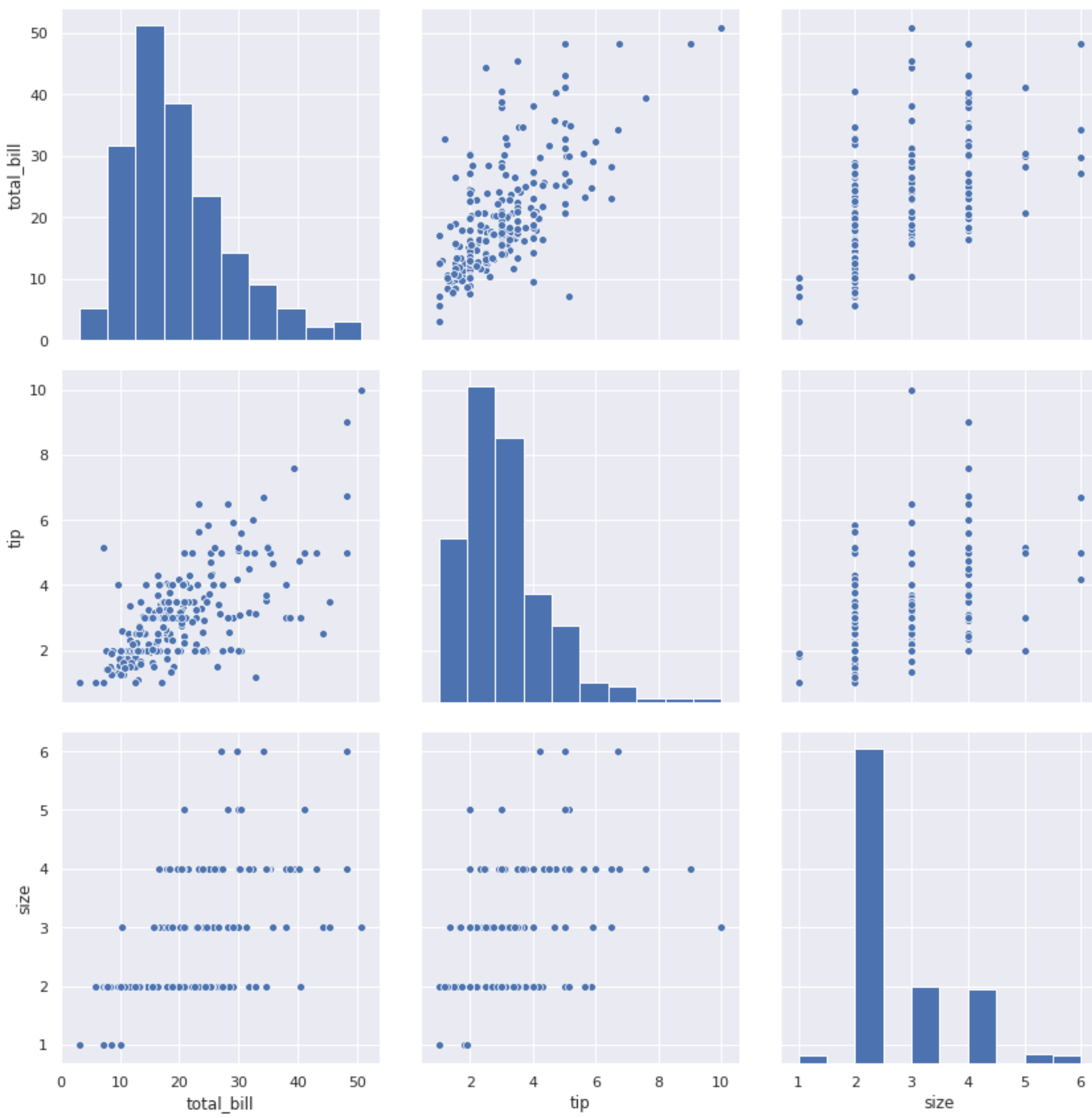
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3

```
3        23.68  3.31    Male     No  Sun  Dinner     2
4        24.59  3.61  Female     No  Sun  Dinner     4
```

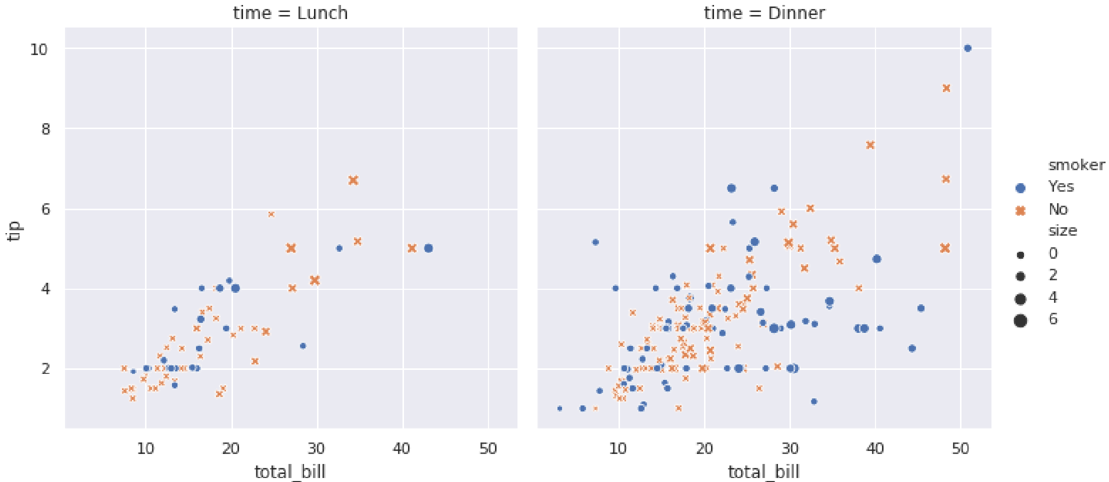
[0]:
```
# make this countplot
```



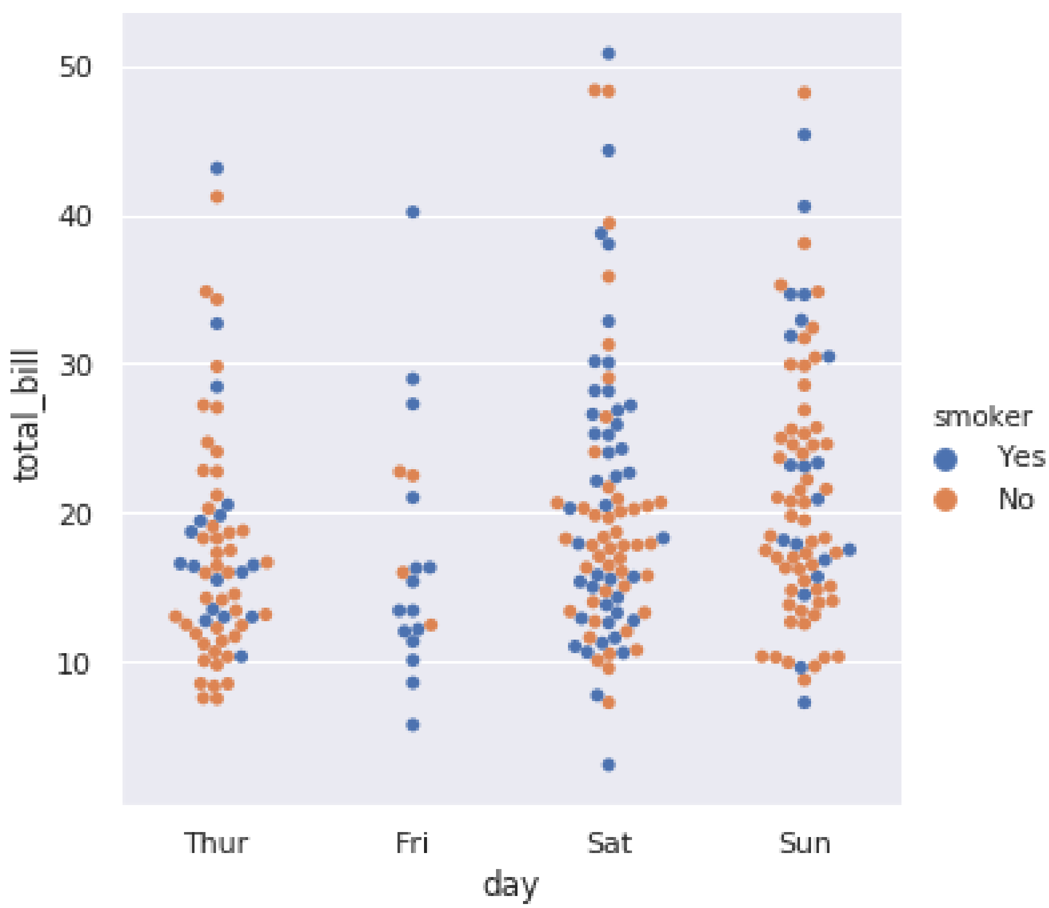
[0]:
```
# make a pairplot
```



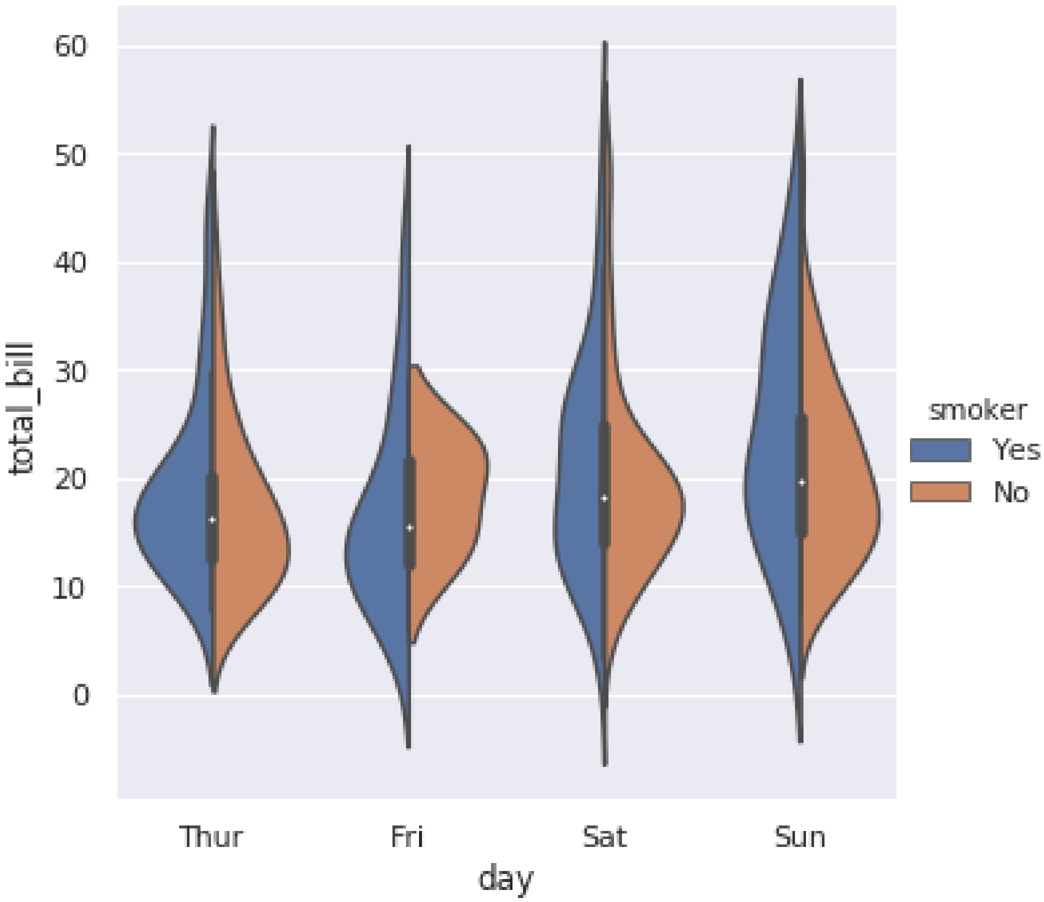
[0]:
```
# make a relplot
```



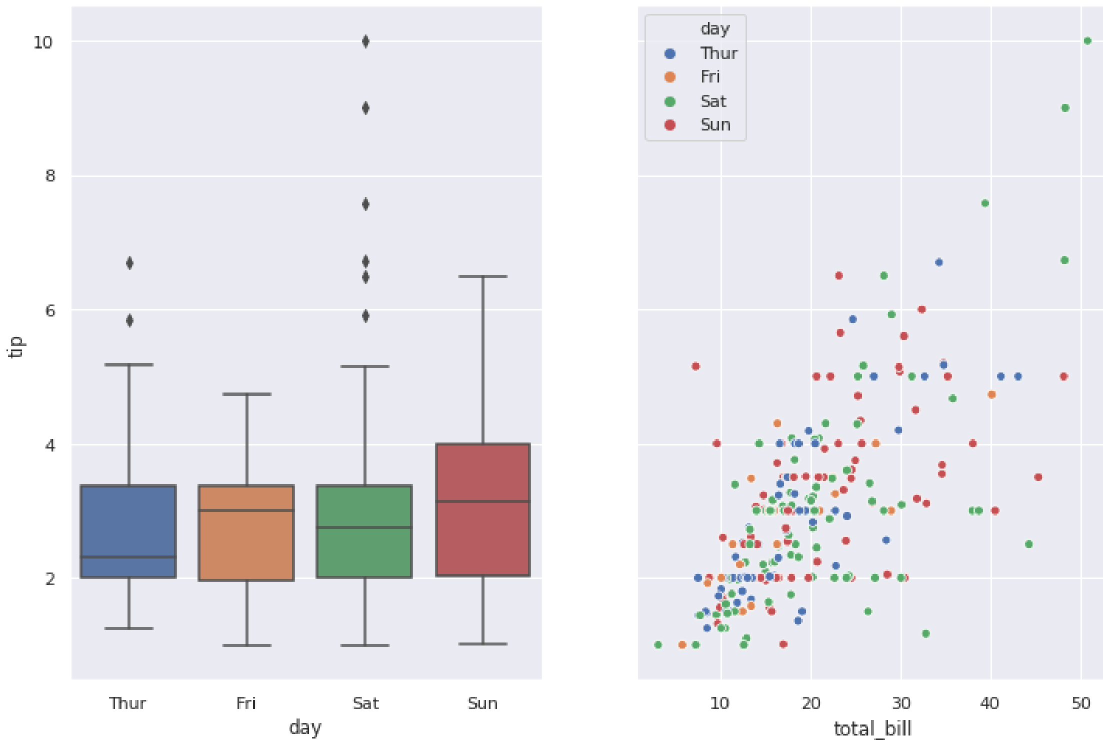
[0]:
```
# make swarm plot
```



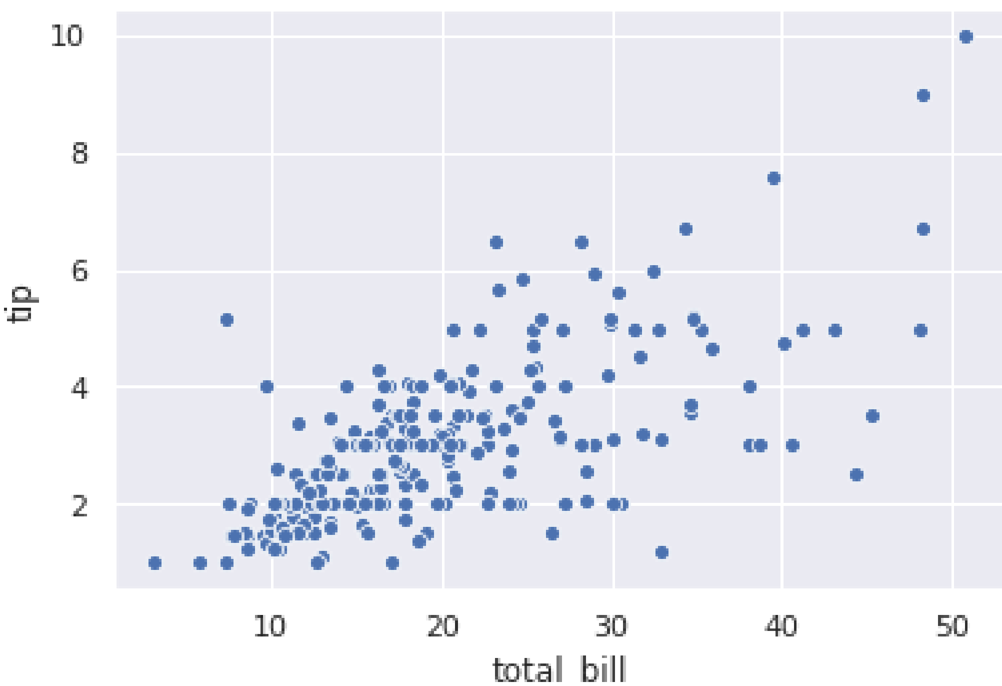
[0]:
```
# make violin plot
```



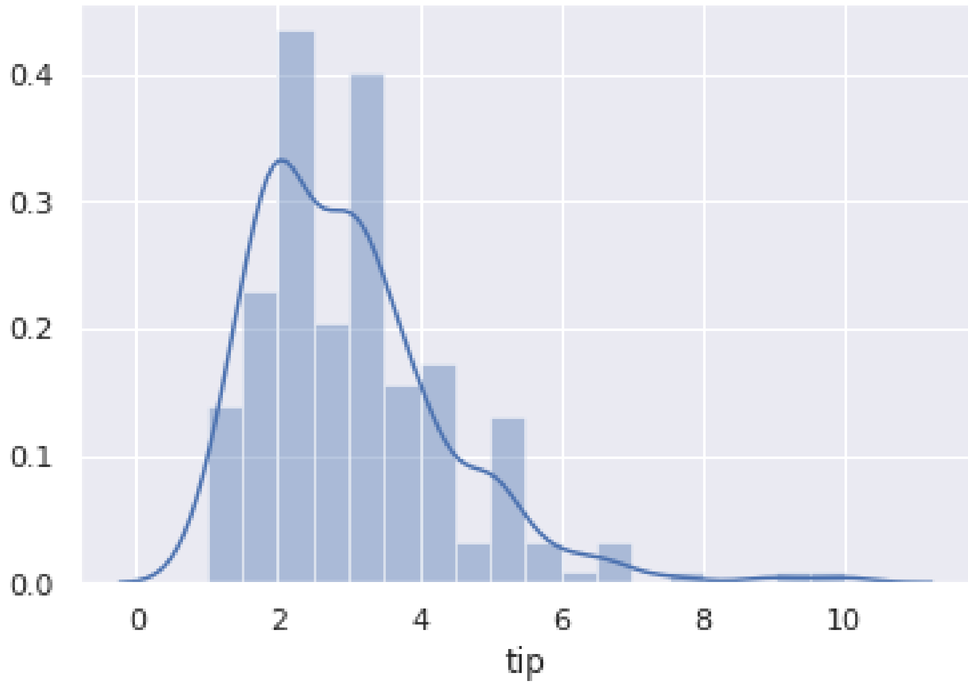
[0]:
```
#make a scatterplot and boxplot
```



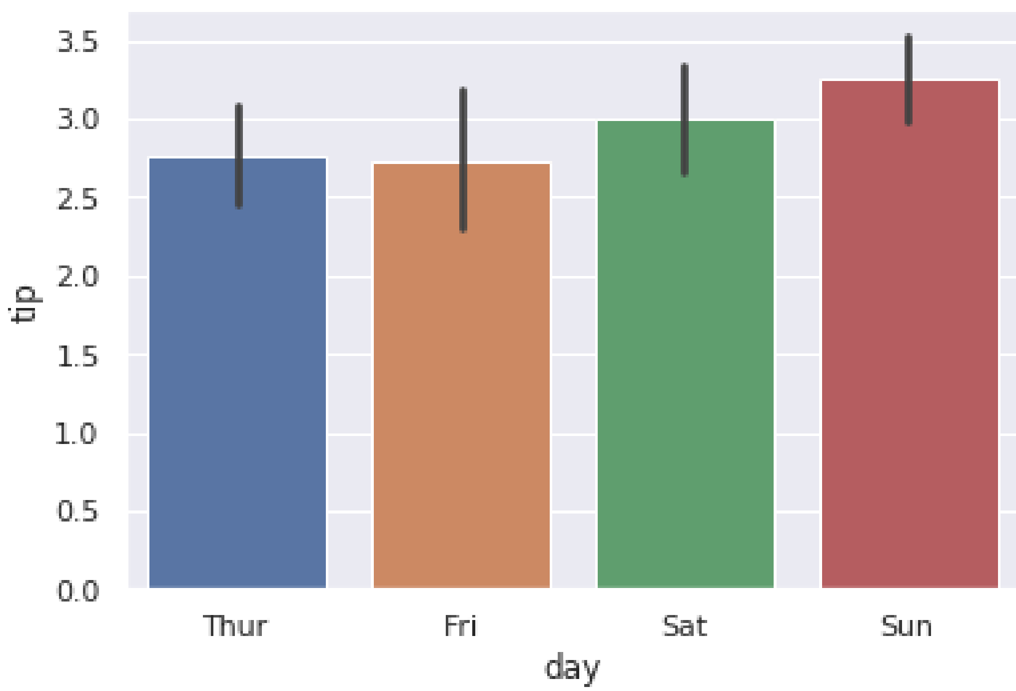
[0]:
```
# make scatterplot not divided by day
```



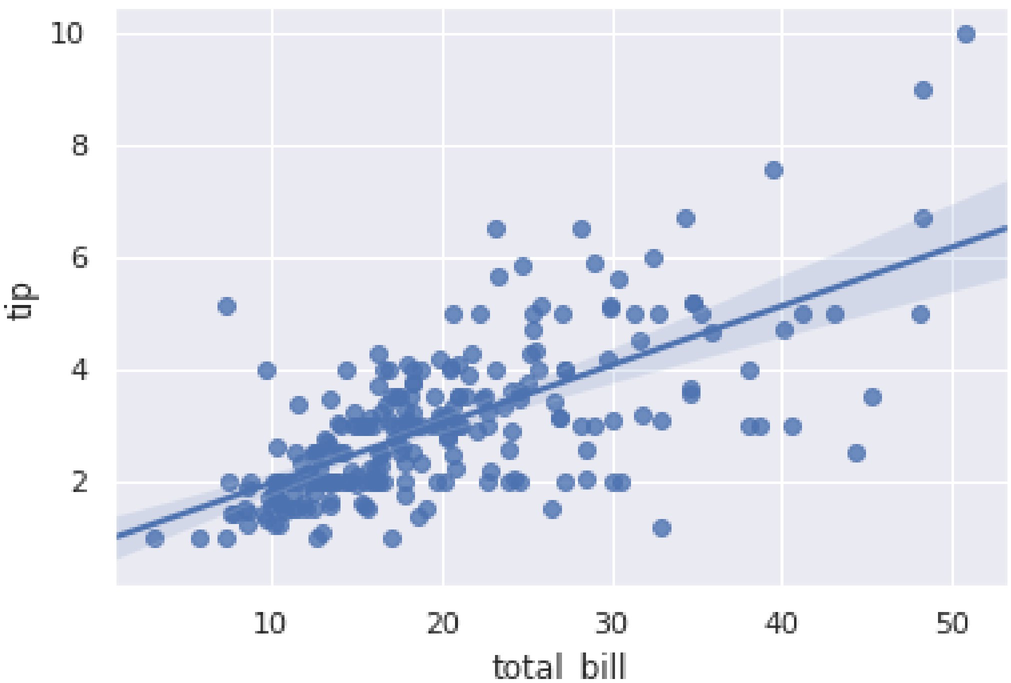
[0]:
```
# make distplot
```



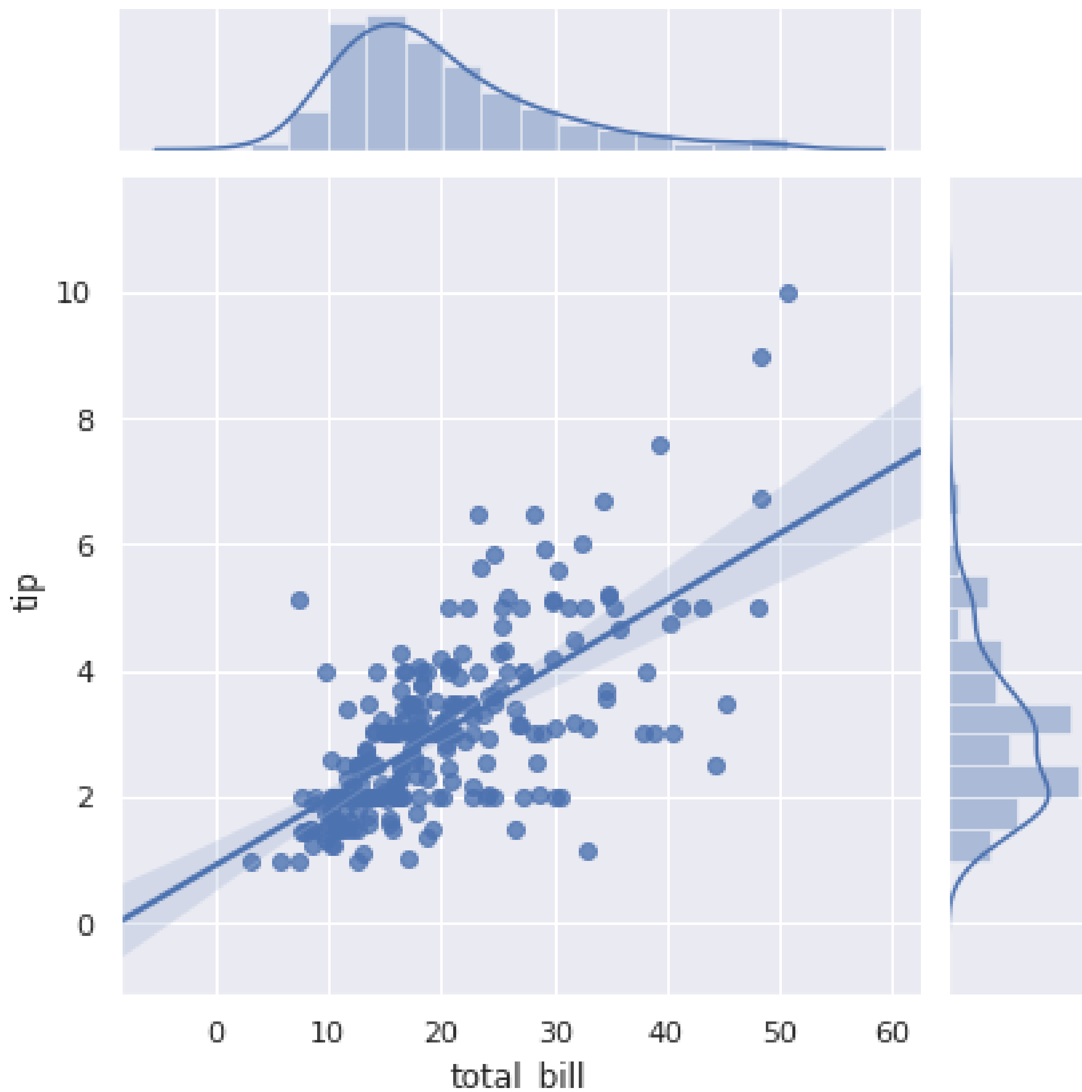
[0]:
```
# make a barplot
```



[0]:
```
# make a regplot
```



[0]:
```
# make jointplot
```



[0]: