پروژه داده کاوی

در این پروژه از دیتاست penguins استفاده کردم که دیتاستی است که بر اساس ابعاد پنگوئن‌ها نوع آن‌ها را پیشبینی می‌کند.

دیتاست penguins شامل ویژگی‌ها یا ستون‌های زیر است:

rowid : شماره سطر

species (در اینجا لیبل کلاس‌ها) : نوع پنگوئن شامل Adelie, Gentoo, Chinstrap

island : جزیره‌ای که پنگوئن در آنجا بوده.

bill_length_mm : طول نوک پنگوئن

bill_depth_mm : ارتفاع نوک پنگوئن

flipper_length_mm : طول بال پنگوئن

body_mass_g : توده بدنی

sex : جنسیت

year : سال اندازه گیری

در این پروژه از زبان برنامه نویسی پایتون محیط colab استفاده شده که به علت ابری بودن بسیاری از مشکلات مثل عدم نصب کتابخانه‌ها را ندارد.

```
import pandas as pd
import seaborn
import matplotlib.pyplot as plt
```

در بلاک اول کتابخانه‌ها را import کردم. شامل pandas که یک کتابخانه برای خواندن از فایل‌های دیتاست و پردازش روی آن داده هاست. seaborn و matplotlib هم کتابخانه‌هایی است که برای مصورسازی و رسم نمودار استفاده می‌شوند.


```
df = pd.read_csv('penguins.csv')
print(df.shape)
print(df.head())
```

در بلاک بعد فایل دیتاست را میخوانیم. چون فرمت فایل csv. است از دستور read_csv استفاده می‌کنیم و سپس ابعاد آن را با df.shape و پنج سطر اول دیتاست را با df.head() چاپ می‌کنیم.

```
(344, 9)
   rowid species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
0      1  Adelie  Torgersen            39.1           18.7              181.0
1      2  Adelie  Torgersen            39.5           17.4              186.0
2      3  Adelie  Torgersen            40.3           18.0              195.0
3      4  Adelie  Torgersen             NaN            NaN                NaN
4      5  Adelie  Torgersen            36.7           19.3              193.0

   body_mass_g     sex  year
0       3750.0    male  2007
1       3800.0  female  2007
2       3250.0  female  2007
3          NaN     NaN  2007
4       3450.0  female  2007
```

```
df.describe()
```

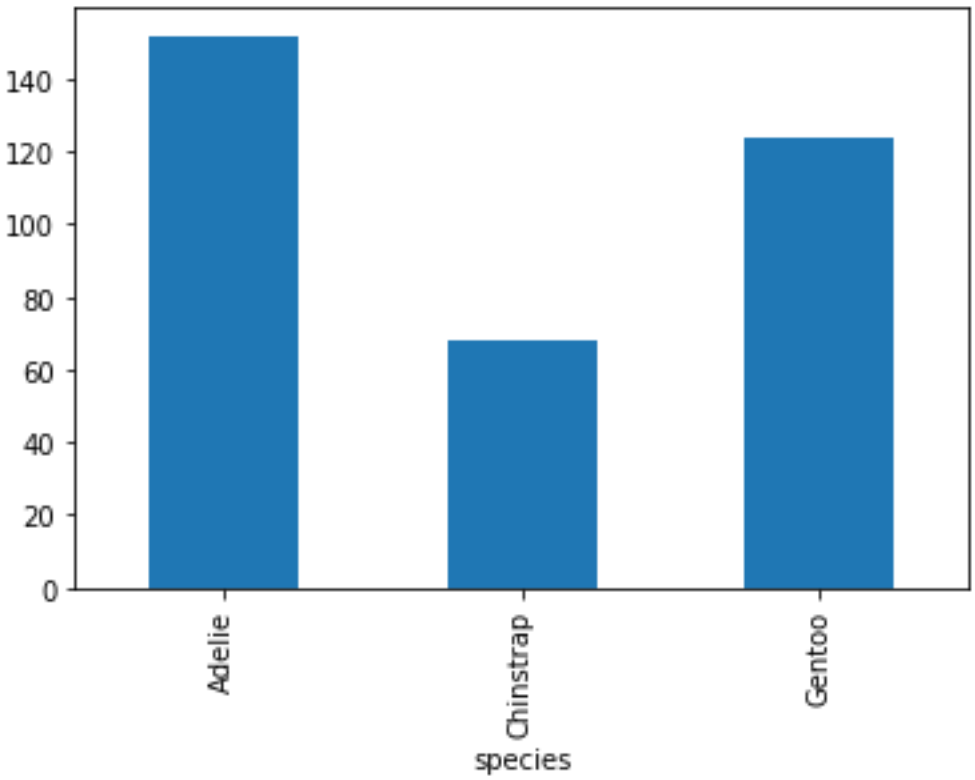
در بلاک بعد توصیفات آماری ستون‌های با مقدار عددی را با دستور describe نمایش می‌دهیم. این داده‌ها برای پر کردن داده‌های null یا رسم نمودار مفید است. مثلا اگر پراکندگی و تعداد داده‌های پرت زیاد باشد معمولا از میانه برای پر کردن داده‌های خالی استفاده می‌شود ولی اگر پراکندگی کمتر باشد می‌توان از میانگین استفاده کرد. همانطور که مشاهده می‌شود در این دیتاست داده‌های پرت زیادی وجود ندارد.

	rowid	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	year
count	344.000000	342.000000	342.000000	342.000000	342.000000	344.000000
mean	172.500000	43.921930	17.151170	200.915205	4201.754386	2008.029070
std	99.448479	5.459584	1.974793	14.061714	801.954536	0.818356
min	1.000000	32.100000	13.100000	172.000000	2700.000000	2007.000000
25%	86.750000	39.225000	15.600000	190.000000	3550.000000	2007.000000
50%	172.500000	44.450000	17.300000	197.000000	4050.000000	2008.000000
75%	258.250000	48.500000	18.700000	213.000000	4750.000000	2009.000000
max	344.000000	59.600000	21.500000	231.000000	6300.000000	2009.000000

در قسمت بعد نمودارهایی رسم می‌کنیم که می‌توانند در بخش‌های بعدی مفید باشند.

```
df2=df.groupby([df.columns[1]]).size()
df2.plot(kind='bar')
```

ابتدا bar plot داده‌ها را روی لیبل (نوع پنگوئن‌ها) رسم می‌کنیم.

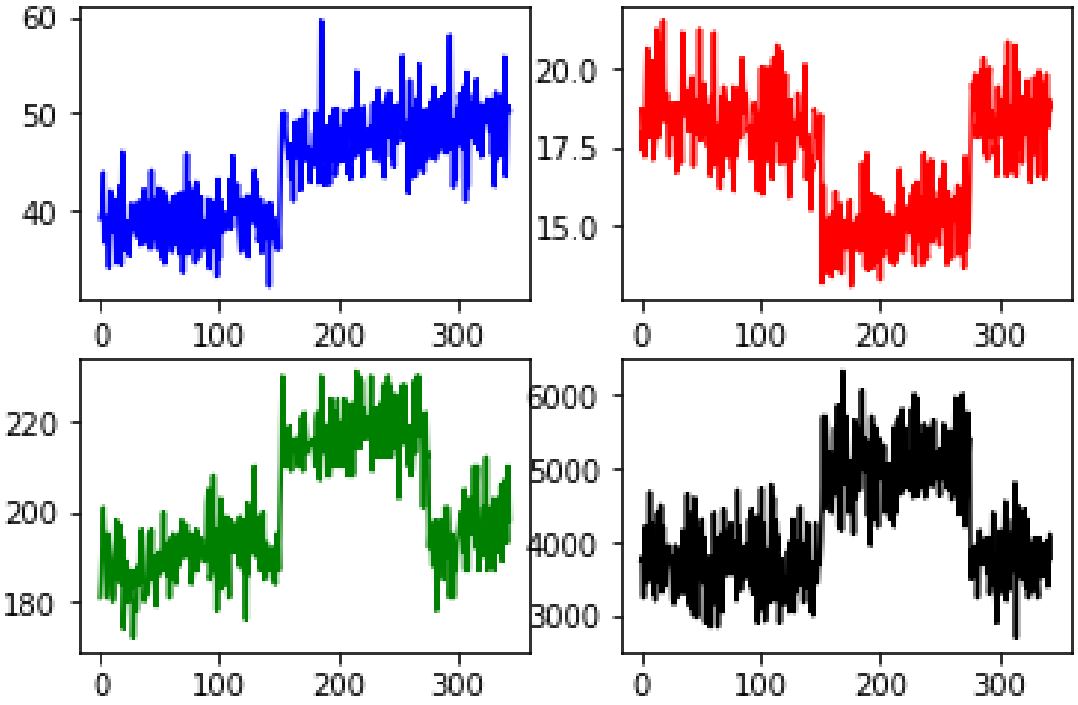


همان‌طور که مشاهده می‌شود داده‌های با برچسب Adelie به مقدار قابل توجهی بیشتر از داده‌های با برچسب Chinstrap است. این متوازن نبودن تعداد داده‌ها می‌تواند در خروجی مدل مشکل‌ساز شود، به خصوص چون تعداد کل داده‌هایمان کم است (۳۴۴ تا).

```
print(df.columns[3:7])

fig, axes = plt.subplots(nrows=2, ncols=2)

df[df.columns[3]].plot(ax=axes[0,0], color='blue')
df[df.columns[4]].plot(ax=axes[0,1], color='red')
df[df.columns[5]].plot(ax=axes[1,0], color='green')
df[df.columns[6]].plot(ax=axes[1,1], color='black')

plt.show()
```

در بلاک بعد داده‌ها را نسبت به چهار ستون با مقادیر عددی bill_length_mm، bill_depth_mm، flipper_length_mm و body_mass_g رسم می‌کنیم. داده‌ها در دیتاست بر اساس لیبل مرتب‌سازی شده‌اند. با استفاده از دستور subplots می‌توان

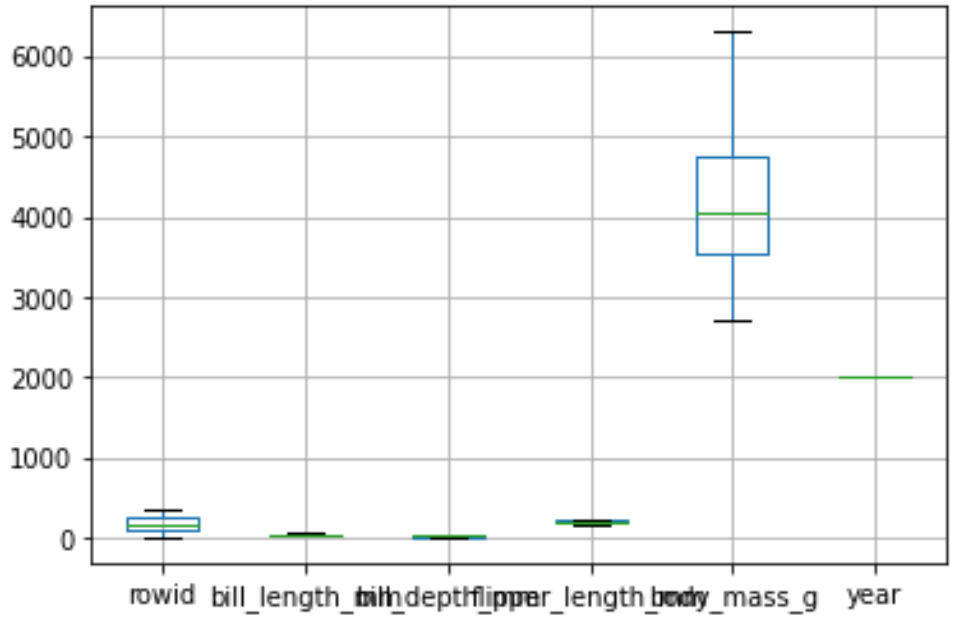
چند نمودار را در یک پنجره نمایش داد. نمودار بالا سمت چپ (آبی) ستون چهارم، نمودار بالا سمت راست (قرمز) ستون پنجم و ... است.



در نمودار اول لیبل اول از لیبل دوم و سوم جدا شده است. یعنی مقادیر این ویژگی برای کلاس دوم و سوم نزدیک به هم و از کلاس اول دور هستند. در نمودارهای دیگر مقادیر ویژگی‌ها برای کلاس اول و سوم نزدیک به هم و برای کلاس دوم این مقدار متفاوت است. همانطور که در این نمودارها هم پیداست تعداد داده‌هایمان برای کلاس سوم کمتر از کلاس-های اول و دوم است. درکل متفاوت بودن این مقادیر به ما نشان می‌دهد که عمل دسته‌بندی (classification) را روی این داده‌ها به آسانی می‌توان انجام داد.

```
df.boxplot()
plt.show(block=True)
```

در بلاک بعد نمودار boxplot داده‌ها را رسم کردیم. از این نمودار برای تشخیص داده‌های پرت استفاده می‌شود.



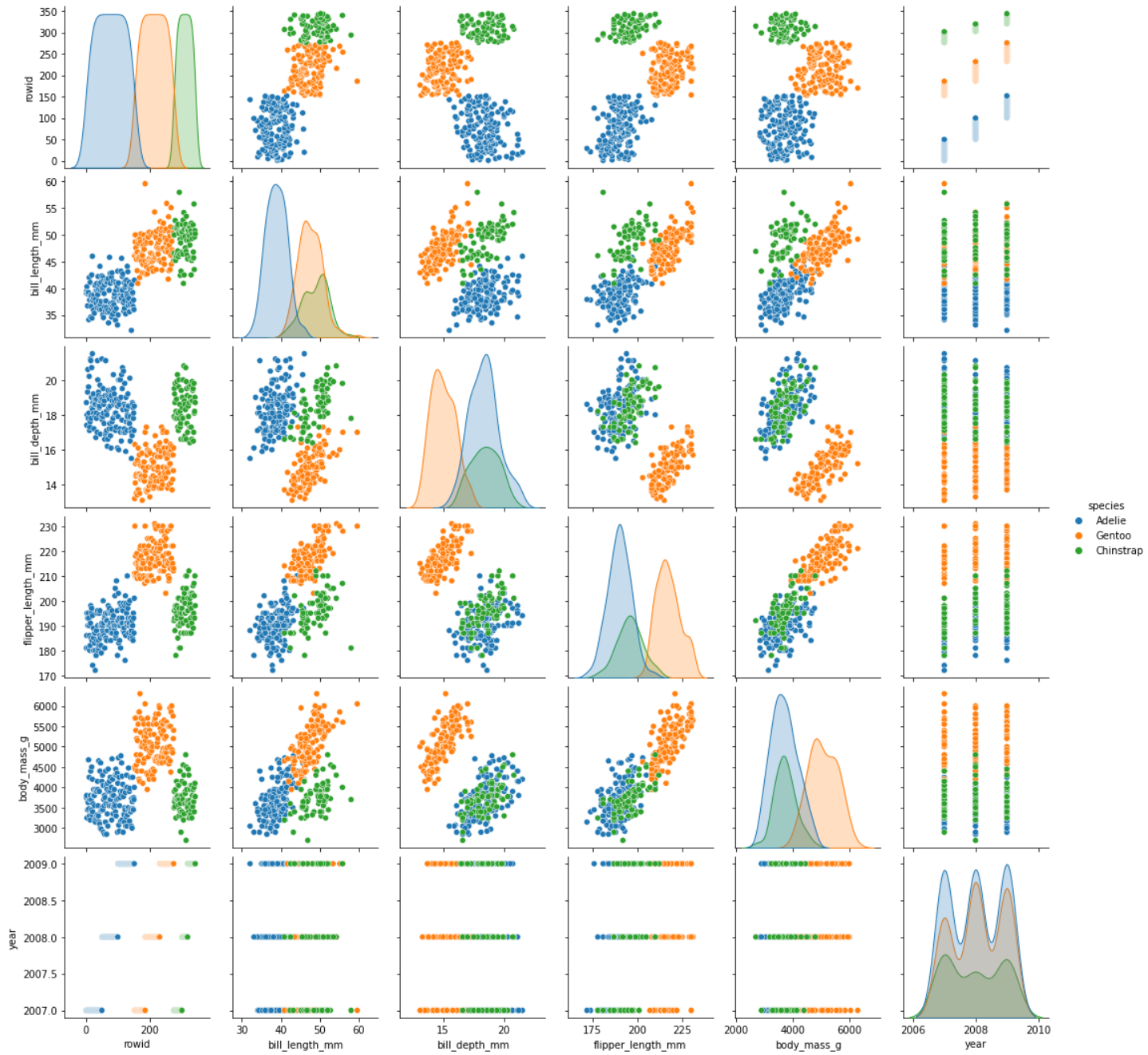
فرمول رسم نمودار box plot :

$$Maximum = \begin{cases} Max(x) & , Max(x) \leq Q3 + 1.5 \times IQR \\ Q3 + 1.5 \times IQR & , Max(x) > Q3 + 1.5 \times IQR \end{cases}$$

$$Minimum = \begin{cases} Min(x) & , Min(x) > Q1 - 1.5 \times IQR \\ Q1 - 1.5 \times IQR & , Min(x) \leq Q1 - 1.5 \times IQR \end{cases}$$

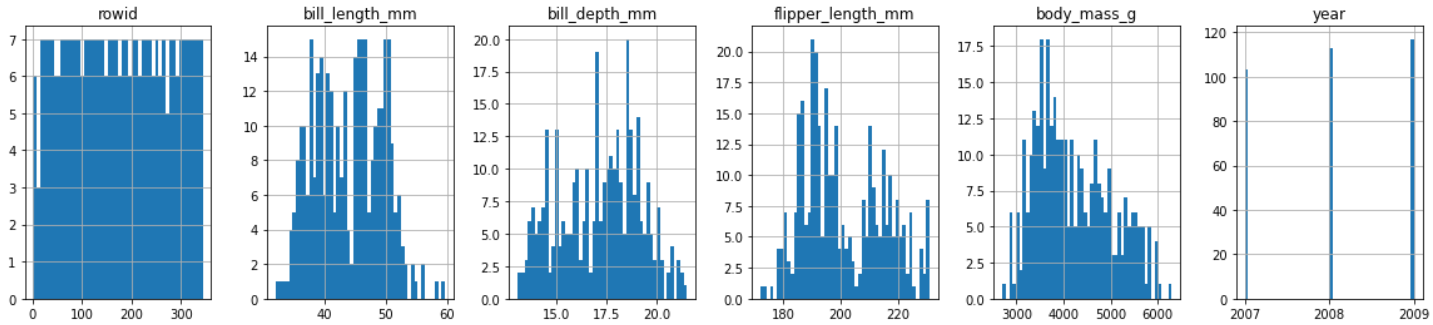
```
# Pair plot
seaborn.pairplot(df, df.columns[1])

plt.show(block=True)
```

به وسیله کتابخانه seaborn، تابع pairplot() را برای ترسیم scatter plot matrix فراخوانی می‌کنیم. df.columns[1] به این معناست که رنگ‌بندی بر اساس داده‌های ستون دوم که species است انجام شود. به وسیله این نمودارها می‌توان ارتباط جفت ویژگی‌ها با لیبل را پیدا کرد.




```
df.hist(bins=50, layout=(1,6), figsize=(20, 4))
plt.show()
```

نمودار بعدی هیستوگرام است. نمودار هیستوگرام، مجموعه‌ای از ستون‌ها است که ارتفاع هر ستون بیانگر میزان فراوانی دسته آن ستون است.



از هیستوگرام‌ها می‌توان فهمید که مثلا تعداد اندازه‌هایی که در سال ۲۰۰۹ گرفته شده مقدار بیشتر از سال پیش و سال ۲۰۰۸ بیشتر از سال قبل از خودش است. یا این که تجمع مقادیر body_mass_g بیشتر بین ۳۰۰۰ تا ۴۰۰۰ است.

Data cleaning

در بخش پاکسازی داده ابتدا داده‌ها null (داده‌هایی که مقداری برایشان وجود ندارد) را پر می‌کنیم. در این دیتاست برخلاف iris چند داده وجود دارد که حداقل یکی از ویژگی‌هایشان خالیست.

```
df.isnull().sum()
```

ابتدا تعداد داده‌های null به ازای هر ستون را چاپ می‌کنیم.

```
rowid                0
species              0
island               0
bill_length_mm       2
bill_depth_mm        2
flipper_length_mm    2
body_mass_g          2
sex                 11
year                 0
```

در سه ستون اول هیچ مقدار خالی وجود ندارد اما ۵ ستون بعد مقدار خالی دارند که در ادامه باید با روش‌های مناسب آن‌ها را پر کرد.

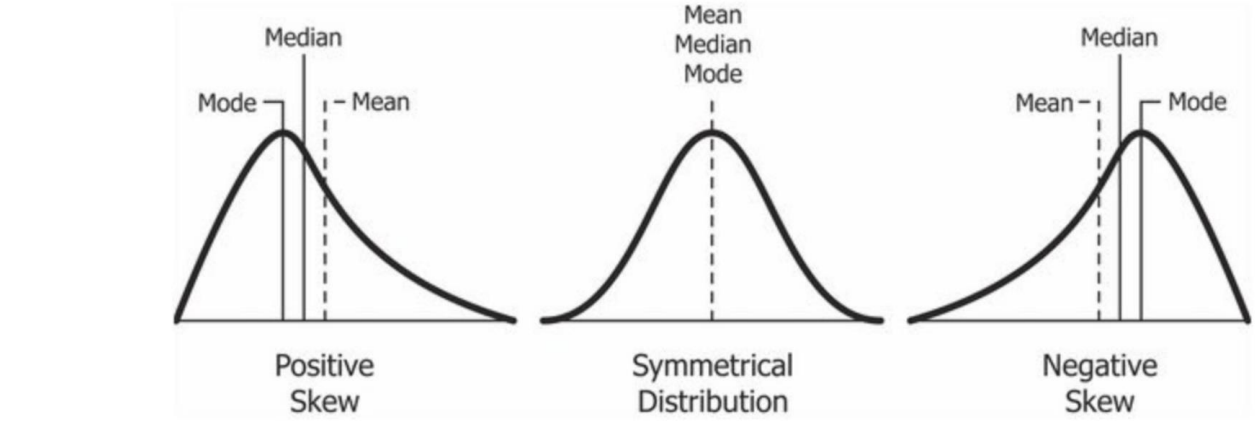
برای مقادیر عددی از روش‌هایی مثل میانگین‌گیری، میانه، مد و ... استفاده کرد. به این صورت که مثلا میانگین اعداد در آن ستون را به دست آوریم و جایگزین مقادیر خالی کنیم.

```
print(df[df.columns[3]].describe())
print('median : ' , df[df.columns[3]].median())
print('mode : ' , df[df.columns[3]].mode())
```

در اینجا ما به عنوان مثال برای ستون چهارم (bill_length_mm) مقادیر تعداد، میانگین، انحراف معیار، مینیمم، ماکسیمم، چارک‌ها، ماکسیمم، میانه و مد را محاسبه کردیم.

```
count    342.000000
mean      43.921930
std        5.459584
min       32.100000
25%       39.225000
50%       44.450000
75%       48.500000
max       59.600000
Name: bill_length_mm, dtype: float64
median :  44.45
mode :  0    41.1
```

طبق تصویر زیر اگر داده‌ها نسبتا متقارن باشند باید میانگین و میانه و مد نزدیک به هم باشند در غیر این صورت چولگی به چپ یا راست داریم که در اینطور مواقع بهتر است از میانه استفاده کنیم.



چون در اینجا این اعداد نسبتا نزدیک به هم هستند و داده پرت زیادی هم نداریم از میانگین استفاده می‌کنیم.

```
print(type(df[df.columns[5]].mean()))
df[df.columns[3]].fillna(value=round(df[df.columns[3]].mean(), 1), inplace=True)
df[df.columns[4]].fillna(value=round(df[df.columns[4]].mean(), 1), inplace=True)
df[df.columns[5]].fillna(value=round(df[df.columns[5]].mean()), inplace=True)
df[df.columns[6]].fillna(value=round(df[df.columns[6]].mean()), inplace=True)
df.head()
```

نوع داده بعد از میانگین‌گیری float است در نتیجه برای اینکه دقت اندازه‌گیری را حفظ کنیم، به همان مقدار که در دیتاست اعداد آن ستون اعشار دارند، به وسیله تابع round() اعشار می‌گذاریم. سپس پنج سطر اول را رسم می‌کنیم.

```
<class 'numpy.float64'>
```

	rowid	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
0	1	Adelie	Torgersen	39.1	18.7	181.0	3750.0	male	2007
1	2	Adelie	Torgersen	39.5	17.4	186.0	3800.0	female	2007
2	3	Adelie	Torgersen	40.3	18.0	195.0	3250.0	female	2007
3	4	Adelie	Torgersen	43.9	17.2	201.0	4202.0	NaN	2007
4	5	Adelie	Torgersen	36.7	19.3	193.0	3450.0	female	2007

همانطور که می‌بینیم مثلا برای داده‌ی چهارم که مقادیر ستون‌های چهارم تا هفتم خالی بود پر شده است.

در بخش بعد داده sex را که عددی نیست پر می‌کنیم. در اینجا از مد استفاده می‌کنیم.

```
print(df[df.columns[7]].describe())
df[df.columns[7]].fillna(df[df.columns[7]].mode()[0], inplace=True)
df.head()
```

ابتدا توصیفی از این ستون را نمایش می‌دهیم. داده‌ها دو دسته (male, female) هستند شامل ۳۴۴ تا که تعداد male ها بیشتر (۱۷۹ تا) است. سپس عملیات پر کردن داده‌های خالی را انجام می‌دهیم و پنج سطر اول را چاپ می‌کنیم. مشاهده می‌شود که مثلا داده‌ی چهارم که مقدار sex آن خالی بود پر شده است.

```
count      344
unique       2
top       male
freq       179
Name: sex, dtype: object
0    male
dtype: object
```

	rowid	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
0	1	Adelie	Torgersen	39.1	18.7	181.0	3750.0	male	2007
1	2	Adelie	Torgersen	39.5	17.4	186.0	3800.0	female	2007
2	3	Adelie	Torgersen	40.3	18.0	195.0	3250.0	female	2007
3	4	Adelie	Torgersen	43.9	17.2	201.0	4202.0	male	2007
4	5	Adelie	Torgersen	36.7	19.3	193.0	3450.0	female	2007

split datas to train and test

در این بخش داده‌ها را به دو دسته آموزش(train) و تست(test) تقسیم می‌کنیم.دسته آموزش داده‌های هستند که مدل ما بر اساس آنها آموزش می‌بینید و دسته تست داده‌هایی هستند که مدل ما در فاز آموزش آنها را ندیده و برای سنجش میزان دقت مدل استفاده می‌شود.

کتابخانه scikit-learn کتابخانه‌ای است که برای محاسبات ماشین لرنینگی استفاده می‌شود. این کتابخانه الگوریتم‌های مختلفی مثل decision tree، svm، knn و ... را در خود دارد.


```python
from sklearn.model_selection import train_test_split
x = df.iloc[:, 3:7]
y = df.iloc[:, 1] #labels
# Split data into the training sets and the testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.15, random_state=45)
print(x_train.shape)
print(x_test.shape)
```

ابتدا ماژول جداسازی داده‌های آموزشی و تست را از این کتابخانه import می‌کنیم.

سپس ستون‌هایی که قرار است عملیات دسته‌بندی بر اساسشان انجام شود را جدا می‌کنیم و در x میریزیم. داده‌های لیبل هم که قرار است خروجی مدلمان باشند را هم در y میریزیم.

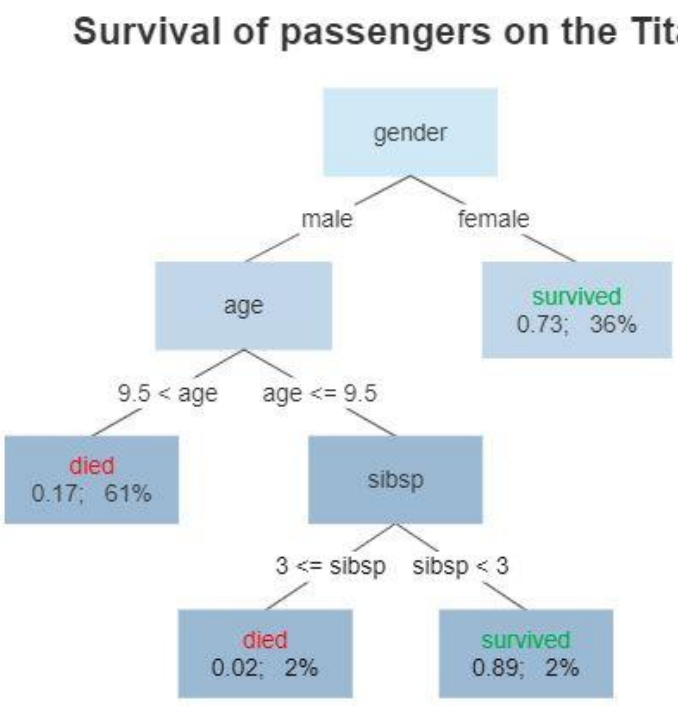
تابع train_test_split به جز x و y دو آرگومان دارد. یکی test_size که نسبت داده‌های آموزش و تست را مشخص می‌کنید. 0.15 یعنی پانزده درصد از داده‌ها به بخش تست و هشتاد و پنج درصد دیگر برای بخش آموزش استفاده شوند. random_state شیوه جداسازی را به صورت تصادفی تعیین می‌کند. تعیین این معیار باعث می‌شود که در دفعات مختلف اجرای الگوریتم جداسازی تصادفی باعث نشود به جواب‌های متفاوتی برسیم. سپس تعداد داده‌های جدا شده برای آموزش و داده‌های جدا شده برای تست را چاپ می‌کنیم.

```
(292, 4)
(52, 4)
```

در بخش بعد به آموزش مدل می‌پردازیم، ابتدا مدل درخت تصمیم (decision tree) و سپس مدل ماشین بردار پشتیبان (svm) را استفاده می‌کنیم.

train model (decision tree)

درخت تصمیم الگوریتمی است که به وسیله آموزش روی مجموعه داده‌ها، یک درخت به عنوان خروجی می‌دهد که در هر گره میانی یک شرط بررسی می‌شود و در گره‌های پایانی لیبل مربوط به آن داده، داده می‌شود. در شکل روبرو مثالی از یک درخت تصمیم را برای بازماندگان کشتی تایتانیک می‌بینیم.



```
from sklearn.tree import DecisionTreeClassifier
model1 = DecisionTreeClassifier()
model1.fit(x_train, y_train)
y_pred = model1.predict(x_test)
```

ماژول درخت تصمیم را از کتابخانه scikit، import می‌کنیم. عملیات یادگیری را به وسیله الگوریتم درخت تصمیم انجام می‌دهیم و داده‌های تست را برای بررسی و سنجش عملکرد مدل به الگوریتم می‌دهیم.

```
from sklearn.metrics import *
# summery of model
print(classification_report(y_test, y_pred))
# accuracy
accuracy = accuracy_score(y_pred,y_test)*100
print(accuracy)
```

ابتدا توابع اندازه‌گیری را از کتابخانه scikit، import می‌کنیم. سپس گزارشی از عملکرد مدل و دقت مدل برای داده‌های تست، نمایش می‌دهیم.

```
              precision    recall  f1-score   support

      Adelie       1.00      0.85      0.92        20
   Chinstrap       0.75      1.00      0.86         9
      Gentoo       0.96      0.96      0.96        23

    accuracy                           0.92        52
   macro avg       0.90      0.94      0.91        52
weighted avg       0.94      0.92      0.92        52

92.3076923076923
```

در اینجا معیارهایی برای سنجش عملکرد مدل می‌بینیم.

Confusion Matrix:

Actual class\Predicted class	C_1	$\neg C_1$
C_1	**True Positives (TP)**	**False Negatives (FN)**
$\neg C_1$	**False Positives (FP)**	**True Negatives (TN)**

Precision : (درستی) چند درصد از داده‌هایی که به عنوان داده درست دسته‌بندی شده‌اند واقعا درست هستند ؟

$$precision = \frac{TP}{TP + FP}$$

Recall : (کامل بودن) : چند درست از داده‌های با برچسب درست به درستی برچسبشان درست تشخیص داده شده است ؟

$$recall = \frac{TP}{TP + FN}$$

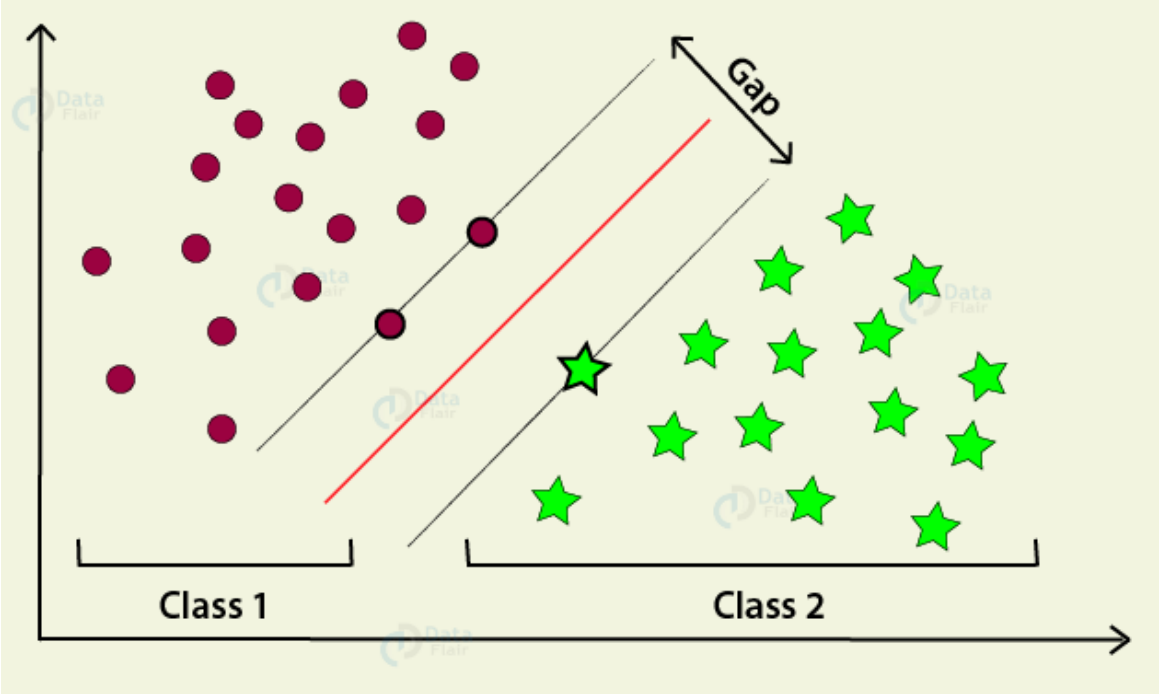
F-measure : میانگین هارمونیک **precision** و **recall**

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

train model (support vector machine)

روش دیگری که برای دسته‌بندی استفاده می‌کنیم روش ماشین بردار پشتیبان (svm) است.

در این روش داده‌ها را به وسیله یک خط مناسب از هم جدا می‌کنیم (تقسیم خطی).



همان عملیاتی که برای درخت تصمیم انجام دادیم را برای svm هم انجام می‌دهیم و میزان عملکرد مدل را نمایش می‌دهیم.

مشاهده می‌شود که برای این داده‌ها با این پیش‌پردازش های انجام شده دقت درخت تصمیم از svm بیشتر است.

```
from sklearn.svm import SVC
model2 = SVC()
model2.fit(x_train, y_train)
y_pred=model2.predict(x_test)
```

```
# summery of model
print(classification_report(y_test, y_pred))
# accuracy
accuracy = accuracy_score(y_pred,y_test)*100
print(accuracy)
```

```
              precision    recall  f1-score   support

      Adelie       0.62      0.80      0.70        20
   Chinstrap       0.00      0.00      0.00         9
      Gentoo       0.81      0.91      0.86        23

    accuracy                           0.71        52
   macro avg       0.47      0.57      0.52        52
weighted avg       0.59      0.71      0.65        52

71.15384615384616
```