In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statistics
```

In [2]:
```
# data loading

df = pd.read_csv('titanic_train.csv')
df.head()

```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.250
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.283
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.925
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.100
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.050

In [3]:
```
# shape

df.shape
```

Out[3]:
```
(891, 12)
```

In [4]:
```
# size
df.size
```

Out[4]:
```
10692
```

In [5]:

```
# info

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 [6]:

```
# describe

df.describe()
```

Out[6]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Far
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.00000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.20420
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.69342
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.00000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.91040
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.45420
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.00000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.32920

In [7]:
```
# check for missing  / null values

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

Out[7]:
```
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 [8]:
```
(df.isna().sum()/df.shape[0])*100
```

Out[8]:
```
PassengerId     0.000000
Survived        0.000000
Pclass          0.000000
Name            0.000000
Sex             0.000000
Age            19.865320
SibSp           0.000000
Parch           0.000000
Ticket          0.000000
Fare            0.000000
Cabin          77.104377
Embarked        0.224467
dtype: float64
```

In [9]:
```
# Treatment of missing data
```

In [10]:
```
# 1) Delete missing rows/ columns
```

In [11]:
```
df.isna().mean()*100
```

Out[11]:
```
PassengerId     0.000000
Survived        0.000000
Pclass          0.000000
Name            0.000000
Sex             0.000000
Age            19.865320
SibSp           0.000000
Parch           0.000000
Ticket          0.000000
Fare            0.000000
Cabin          77.104377
Embarked        0.224467
dtype: float64
```

In [12]:
```
df_1 = df.dropna(how = 'any')
```

In [13]:
```
df_1.isnull().sum()
```

Out[13]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

In [14]:
```
df_1.shape
```

Out[14]:
```
(183, 12)
```

In [15]:
```
# deleting specific column

df_2 = df.drop(columns = ['Cabin'])
```

In [16]:
```
df_2.columns
```

Out[16]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age',
'SibSp',
       'Parch', 'Ticket', 'Fare', 'Embarked'],
      dtype='object')
```

In [17]:
```
df_2.isnull().sum()
```

Out[17]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Embarked         2
dtype: int64
```

In [18]:
```
# 2) impute missing value
```

In [19]:
```
# imputing using mean

df_3 = df['Age'].fillna(df['Age'].mean())

```

In [20]:
```
df_3.isnull().sum()
```

Out[20]: 0

In [21]:
```
# imputing using median

df_4 = df['Age'].fillna(df['Age'].median())
df_4.isnull().sum()
```

Out[21]: 0

In [22]:
```
# imputing using mode

df_5 = df['Embarked'].fillna(df['Embarked'].mode()[0])
df_5.isnull().sum()
```

Out[22]: 0

In [23]:
```
# imputing using another methods
```

In [24]:
```
df_6 = df['Embarked'].fillna(method = 'ffill')
df_6.isnull().sum()
```

Out[24]: 0

In [25]:
```
df_7 = df['Embarked'].fillna(method = 'bfill')
df_7.isnull().sum()
```

Out[25]: 0

In [26]:
```
# load dataset
```

In [27]:

```
from sklearn.datasets import load_boston
bos_house = load_boston()

column_name = bos_house.feature_names
df_boston = pd.DataFrame(bos_house.data)
df_boston.columns = column_name

df_boston.head()
```

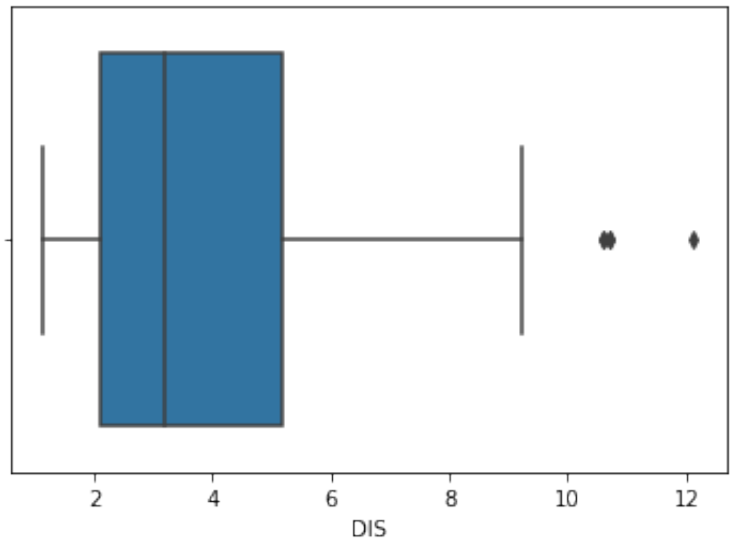
Out[27]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90

In [28]:

```
# boxplot

sns.boxplot(df_boston['DIS'])
plt.show()
```



In [29]:

```
# position of outlier

print(np.where(df_boston['DIS']>10))
```

```
(array([351, 352, 353, 354, 355]),)
```

In [30]:
```
# Z score
from scipy import stats
z = np.abs(stats.zscore(df_boston['DIS']))
print(z)
```

```
[1.40213603e-01 5.57159875e-01 5.57159875e-01 1.07773662e+00
 1.07773662e+00 1.07773662e+00 8.39243922e-01 1.02463789e+00
 1.08719646e+00 1.32963473e+00 1.21297914e+00 1.15593484e+00
 7.87143464e-01 4.33754047e-01 3.17003386e-01 3.34449434e-01
 3.34449434e-01 2.20028082e-01 6.92761271e-04 6.92761271e-04
 1.35827806e-03 1.03277421e-01 8.64493539e-02 1.42685523e-01
 2.87387889e-01 3.13533191e-01 4.21632134e-01 3.12962749e-01
 3.13580728e-01 2.11043605e-01 2.08191390e-01 1.80619980e-01
 9.26766896e-02 3.72817172e-03 1.67532861e-02 2.06663257e-01
 1.98296760e-01 6.61510917e-02 2.48415135e-02 7.63470081e-01
 7.63470081e-01 9.15493132e-01 9.15493132e-01 9.15493132e-01
 9.15493132e-01 6.20526581e-01 6.20526581e-01 9.00519004e-01
 9.86370670e-01 1.08985853e+00 1.43545190e+00 1.43545190e+00
 1.43545190e+00 1.43545190e+00 1.67551331e+00 2.33004908e+00
 2.56345533e+00 2.15330683e+00 1.91086857e+00 1.49121270e+00
 1.63068600e+00 1.43725830e+00 1.63073354e+00 1.98982739e+00
 2.58023586e+00 1.33885689e+00 1.33885689e+00 1.28490249e+00
 1.28490249e+00 1.28490249e+00 7.09373073e-01 7.09373073e-01
 7.09373073e-01 7.09373073e-01 2.16985719e-01 3.36350910e-01
```

In [31]:
```
threshold = 3
# position of outlier

print(np.where(z>3))
```

```
(array([351, 352, 353, 354, 355]),)
```

In [32]:
```
# IQR method

# calculate q1, q3
q1 = np.percentile(df_boston['DIS'],25,interpolation = 'midpoin
q3 = np.percentile(df_boston['DIS'],75,interpolation = 'midpoin

# calculate Interquartile range
IQR = q3-q1

# calculate upper fence
upper = df_boston['DIS']>=(q3+1.5*IQR)
print("datapoints above upper fence :",np.where(upper))

# calculate lower fence
lower = df_boston['DIS']<=(q1-1.5*IQR)
print("datapoints below lower fence :",np.where(lower))
```

```
datapoints above upper fence : (array([351, 352, 353, 354, 355]),)
datapoints below lower fence : (array([], dtype=int64),)
```

In [ ]:
```

```

In [33]:
```
# feature scaling by minmax scaler
```

In [34]:
```
from sklearn.preprocessing import MinMaxScaler

# initialize scaler
normalizer = MinMaxScaler()

# fit on data
normalizer.fit(df_boston)

# transform
boston_transformed_df = normalizer.transform(df_boston)

# save into diff data
boston_transformed_df = pd.DataFrame(boston_transformed_df)
boston_transformed_df.columns = df_boston.columns

```

In [35]:
```
# before scaling
df_boston.head()
```

Out[35]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90

In [36]:
```
# after scaling
boston_transformed_df.head()
```

Out[36]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	
0	0.000000	0.18	0.067815	0.0	0.314815	0.577505	0.641607	0.269203	0.000000	0.208
1	0.000236	0.00	0.242302	0.0	0.172840	0.547998	0.782698	0.348962	0.043478	0.104
2	0.000236	0.00	0.242302	0.0	0.172840	0.694386	0.599382	0.348962	0.043478	0.104
3	0.000293	0.00	0.063050	0.0	0.150206	0.658555	0.441813	0.448545	0.086957	0.066
4	0.000705	0.00	0.063050	0.0	0.150206	0.687105	0.528321	0.448545	0.086957	0.066

In [37]:
```
# feature scaling by standard scaler
```

In [38]:

```
from sklearn.preprocessing import StandardScaler

# initialize scaler
standardizer = StandardScaler()

# fit and transform on data
boston_std_df =  standardizer.fit_transform(df_boston)


# save into diff data
boston_std_df = pd.DataFrame(boston_std_df)
boston_std_df.columns = df_boston.columns

boston_std_df.head()
```

Out[38]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	
0	-0.419782	0.284830	-1.287909	-0.272599	-0.144217	0.413672	-0.120013	0.140214	-0.9
1	-0.417339	-0.487722	-0.593381	-0.272599	-0.740262	0.194274	0.367166	0.557160	-0.8
2	-0.417342	-0.487722	-0.593381	-0.272599	-0.740262	1.282714	-0.265812	0.557160	-0.8
3	-0.416750	-0.487722	-1.306878	-0.272599	-0.835284	1.016303	-0.809889	1.077737	-0.7
4	-0.412482	-0.487722	-1.306878	-0.272599	-0.835284	1.228577	-0.511180	1.077737	-0.7

In [39]:

```
# titanic data
df.head()
```

Out[39]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.250
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.283
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.925
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.100
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.050

In [40]:

```
# categorical to numerical data conversion
```

In [41]:

```
# 1) label encoding
```

In [42]:

```
from sklearn.preprocessing import LabelEncoder

# initialize

encoder = LabelEncoder()

df['Gender_encoded'] = encoder.fit_transform(df['Sex'])
```

In [43]:

```
df.head()
```

Out[43]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.250
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.283
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.925
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.100
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.050

In [44]:

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

Out[44]:

```
male      577
female    314
Name: Sex, dtype: int64
```

In [45]:

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

Out[45]:

```
1    577
0    314
Name: Gender_encoded, dtype: int64
```

In [46]:
```
# 2) By using user defined dictionary

data_dictionary = {'female': 10,'male': 7}

# use apply method
df['Gender_new'] = df.Sex.map(data_dictionary)

```

In [47]:
```
df.head()
```

Out[47]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.250
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.283
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.925
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.100
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.050

In [ ]:
```

```

In [48]:
```
# one hot encoding
```

In [49]:
```
df['Embarked'].value_counts()
```

Out[49]:
```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

In [50]:
```
df_new = pd.get_dummies(df['Embarked'])
```

In [51]:
```
df_new
```

Out[51]:

	C	Q	S
0	0	0	1
1	1	0	0
2	0	0	1
3	0	0	1
4	0	0	1
...	...	...	...
886	0	0	1
887	0	0	1
888	0	0	1
889	1	0	0
890	0	1	0

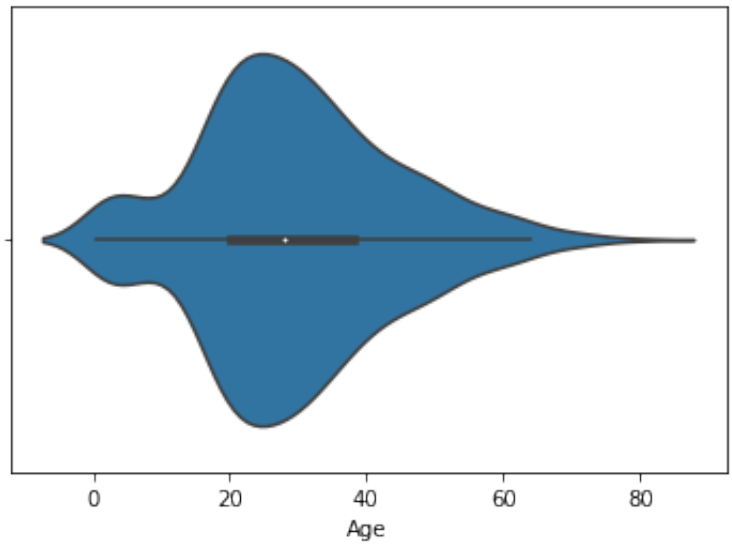
891 rows × 3 columns

In [52]:
```
df['Cabin'].value_counts()
```

Out[52]:
```
C23 C25 C27    4
B96 B98        4
G6             4
D              3
F2             3
              ..
D21            1
A6             1
C90            1
B71            1
A19            1
Name: Cabin, Length: 147, dtype: int64
```

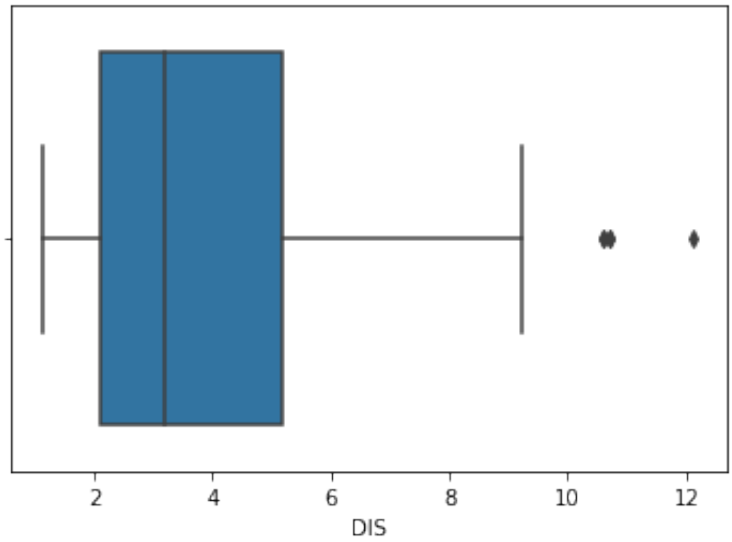
In [53]:
```
# violin plot
sns.violinplot(x=df["Age"])
```

Out[53]: <matplotlib.axes._subplots.AxesSubplot at 0x7f91b423b130>



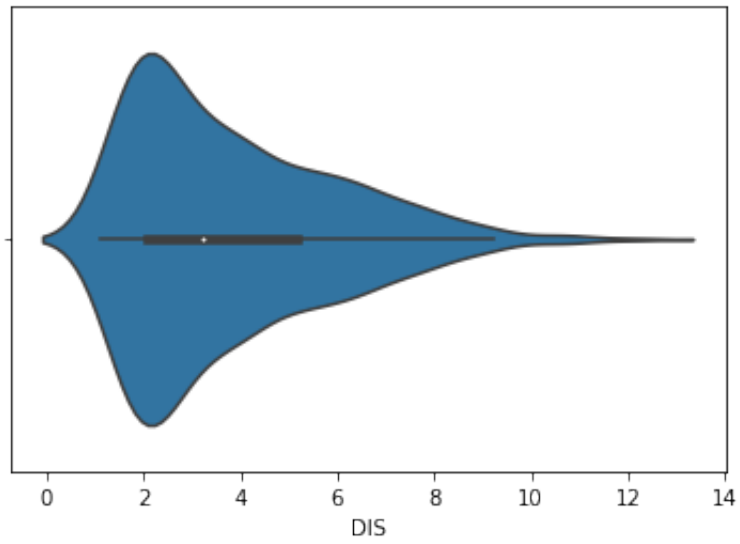
In [54]:
```

sns.boxplot(df_boston['DIS'])
plt.show()
```



In [55]:

```
1
2 sns.violinplot(df_boston['DIS'])
3 plt.show()
```



In [56]:
```
1 sns.get_dataset_names()
```

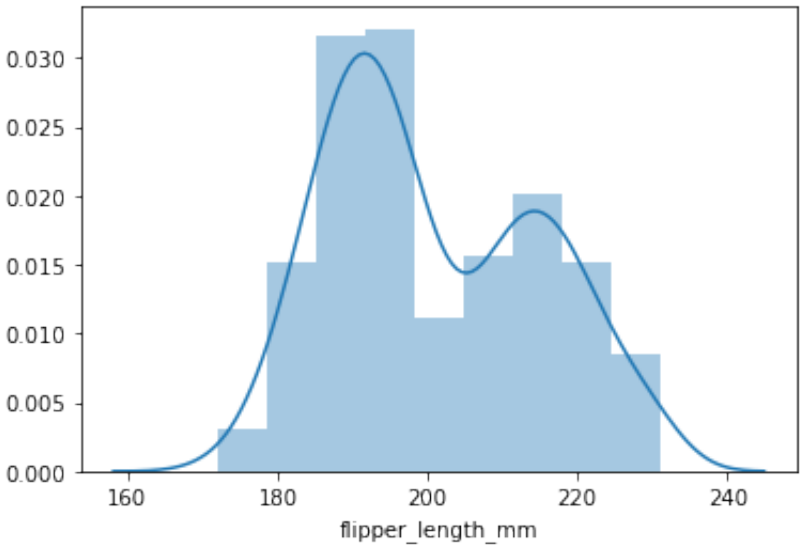
```
/Users/kunalshriwas/opt/anaconda3/lib/python3.8/site-packages/seab
orn/utils.py:384: GuessedAtParserWarning: No parser was explicitly
specified, so I'm using the best available HTML parser for this sy
stem ("lxml"). This usually isn't a problem, but if you run this c
ode on another system, or in a different virtual environment, it m
ay use a different parser and behave differently.

The code that caused this warning is on line 384 of the file /User
s/kunalshriwas/opt/anaconda3/lib/python3.8/site-packages/seaborn/u
tils.py. To get rid of this warning, pass the additional argument
'features="lxml"' to the BeautifulSoup constructor.

  gh_list = BeautifulSoup(http)
```

Out[56]:
```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'dowjones',
 'exercise',
 'flights',
 'fmri',
 'geyser',
 'glue',
 'healthexp',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'seaice',
 'taxis',
 'tips',
 'titanic']
```

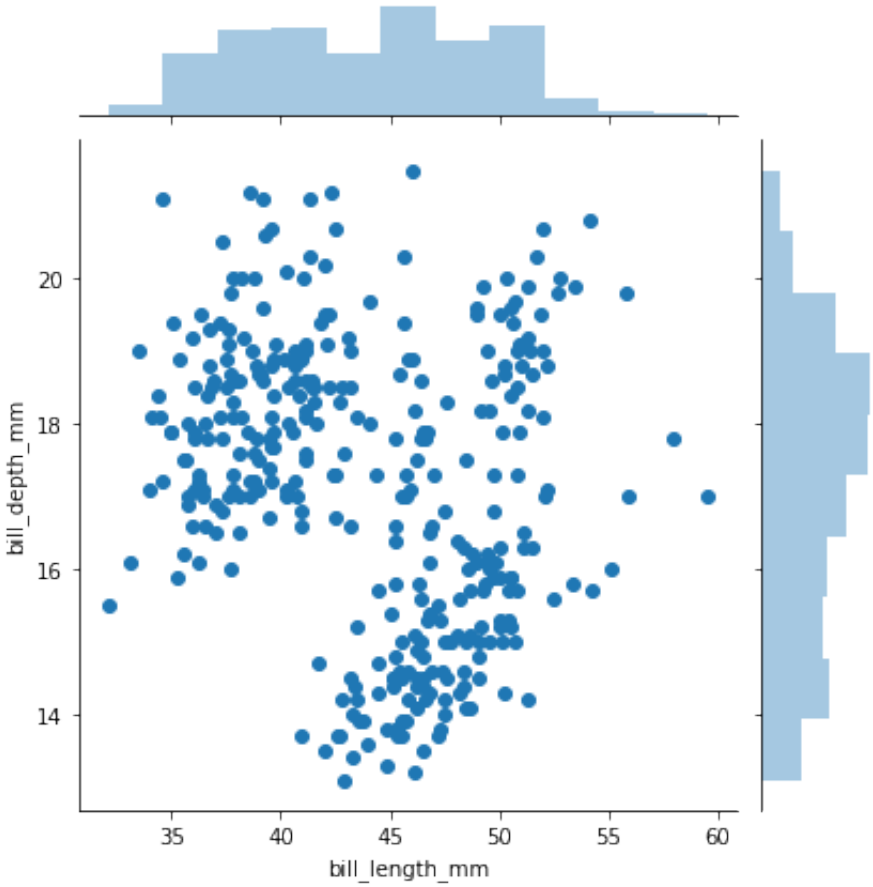
In [58]:
```
penguins = sns.load_dataset("penguins")
sns.distplot(penguins["flipper_length_mm"])
plt.show()
```



In [59]:
```
# joint plot

sns.jointplot(data = penguins,x = 'bill_length_mm',y = 'bill_de
```

Out[59]: <seaborn.axisgrid.JointGrid at 0x7f91b4ca51c0>



In [ ]:
```

```

In [ ]:

```
1
```