LAB 2 - INTRODUCTION TO DATA VISUALIZATION AND EDA

Submitted by

Name : Kavitha.S
Register Number : 21122033
Class : 2MscDS

LAB OVERVIEW:

OBJECTIVE:

Part A)Create a dataset with 12 features and 3 classes using make_classification.
Name the features F1, F2, ... upto F12.
F1: Can hold two values (True/False)
F2: Can hold two values (Type 1 or Type 2)
F3: Can hold three values (A or B or C)
F4: Can hold three values (HIGH / MEDIUM / LOW)
Features F5 to F12 will have numeric values generated during the creation process.

Part B) Use either of the above datasets to show the usage of distplot, jointplot, pairplot, rugplot, catplot, barplot, countplot, violinplot, striplot, swarmplot and facetgrid plots.

Part C) Load the Iris Dataset, and explore - scatterplot, scatter_3d, heatmap, boxplot, kdeplot etc.
In both the cases, write your observations on the plot outputs and how it is relevant.

PROBLEM DEFINTION:

Create a dataset with various features using make_classification.
Import Iris toy dataset.
Exploring the datasets using various visualization technique.
Analyzing the dataset and finding hidden patterns.

APPROACH:

Import the necessary libraries. Creating a dataset using make_classification. Using various kinds of graphs to visualize the dataset. Finding many patters and descriptions about the dataset from summary and graphs.

Importing Libraries

In [1]:
```
from sklearn import datasets
import numpy as np
import pandas as pd
from sklearn.datasets import make_classification
```

```
import seaborn as sns
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
```

PART 1

In [2]:
```
def make_classification_df(n_samples: int = 100,
                           n_num_features: int = 12,
                           n_informative : int = 3,
                           class_sep: float = 1.0,
                           n_classes: int = 3,
                           feature_name: str = 'F{}',
                           target_name: str = 'target',
                           random_state: int = 0):

    np.random.seed(random_state)
    X, y = make_classification(n_samples=n_samples, n_features=n_num_features,class_sep
                               random_state=random_state, n_classes=n_classes)

    return X,y
```

In [3]:
```
X,y = make_classification_df()
```

In [4]:
```
df = pd.DataFrame(X, columns=['F{}'.format(i) for i in range(1,13)])
df["Target"]=y
```

In [5]:
```
df['Class'] = df["Target"]
```

In [6]:
```
df['F1'] = pd.Series(np.random.choice(['True', 'False'],size=100 ))
df['F2'] = pd.Series(np.random.choice(['Type 1','Type 2'],size=100))
df['F3'] = pd.Series(np.random.choice(['A','B','C'],size=100))
df['F4'] = pd.Series(np.random.choice(['High','Low','Medium'],size=100))
```

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

Out[7]:

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
0	True	Type 2	B	High	-0.911351	-2.704330	1.242128	0.845067	-1.468556	-0.584190	-0.530002
1	False	Type 1	C	Low	-1.407409	-2.099632	0.386761	-0.930140	-0.005134	-1.494074	0.468739
2	False	Type 1	C	Low	3.113586	3.048215	-1.104686	1.727567	-0.723721	0.050057	0.034505
3	True	Type 2	C	High	0.893853	-0.731941	-2.191124	-1.276128	0.092591	0.634618	0.833959
4	False	Type 1	B	High	-0.846092	-0.858115	-1.336613	1.419603	-0.341672	-0.909416	0.590905

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
...	...	...	...	...	...	...	...	...	...	...	...
95	True	Type 1	B	Medium	1.055126	-0.891477	1.371886	-0.218017	-0.296105	0.459657	-0.298291
96	True	Type 1	B	Medium	2.685427	-0.698793	0.181983	-0.516968	-1.338409	-0.660834	0.305399
97	False	Type 2	A	High	2.176706	-1.086665	-1.383349	-1.873147	-0.287391	-1.006253	-0.212164
98	False	Type 1	C	Low	-1.909697	0.004045	-1.359697	0.400937	-2.138017	0.706842	0.792608
99	True	Type 1	C	Low	-2.084106	-3.428054	-0.391051	1.050142	-0.388420	-0.032069	-0.047734

100 rows × 14 columns

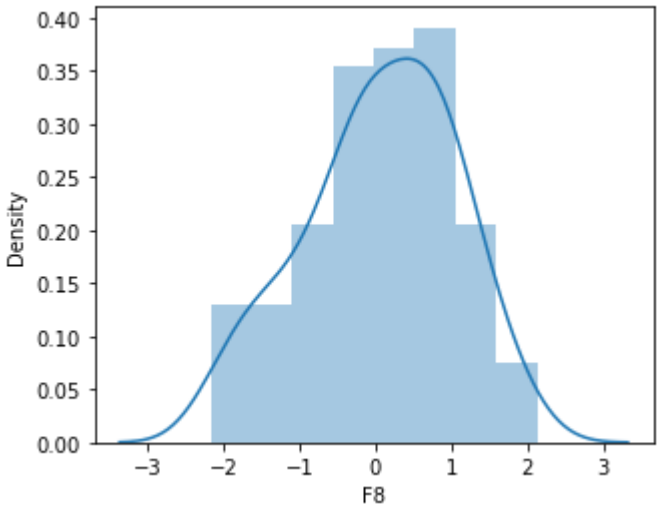
Distplot:

Seaborn distplot lets you show a histogram with a line on it. This can be shown in all kinds of variations. We use seaborn in combination with matplotlib, the Python plotting module. A distplot plots a univariate distribution of observations.

In [8]:
```
fig=plt.figure(figsize=(5,4))
sns.distplot(df['F8'])
```

```
C:\Users\SRIDHAR\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibilit
y) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[8]: <AxesSubplot:xlabel='F8', ylabel='Density'>



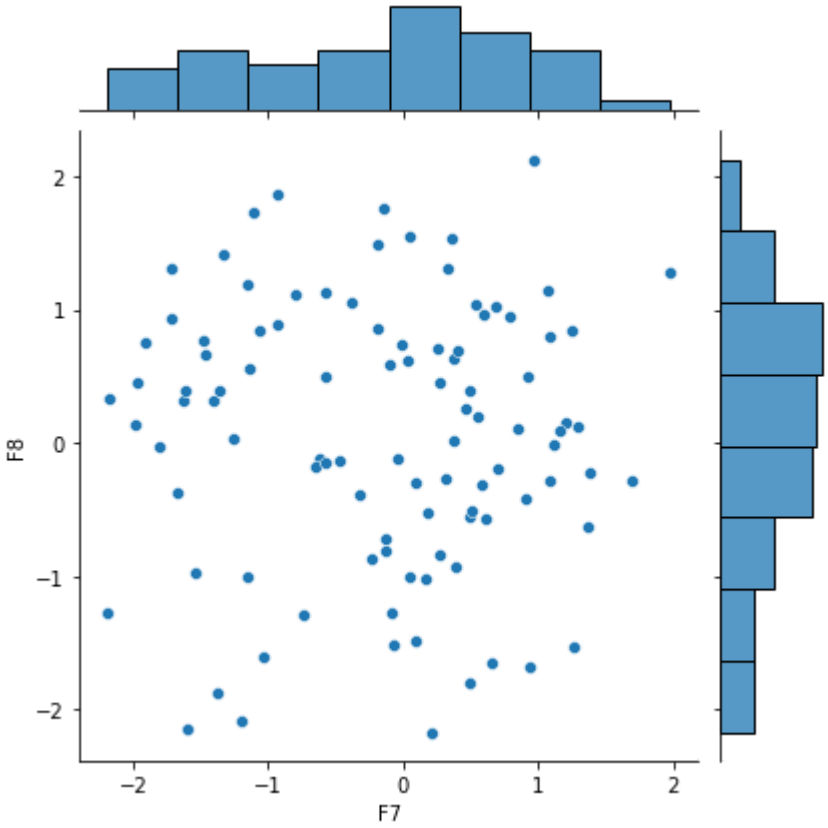
Jointplot:

Seaborn's jointplot displays a relationship between 2 variables (bivariate) as well as 1D profiles (univariate) in the margins.

In [9]:
```
sns.jointplot(df['F7'],df['F8'])
```

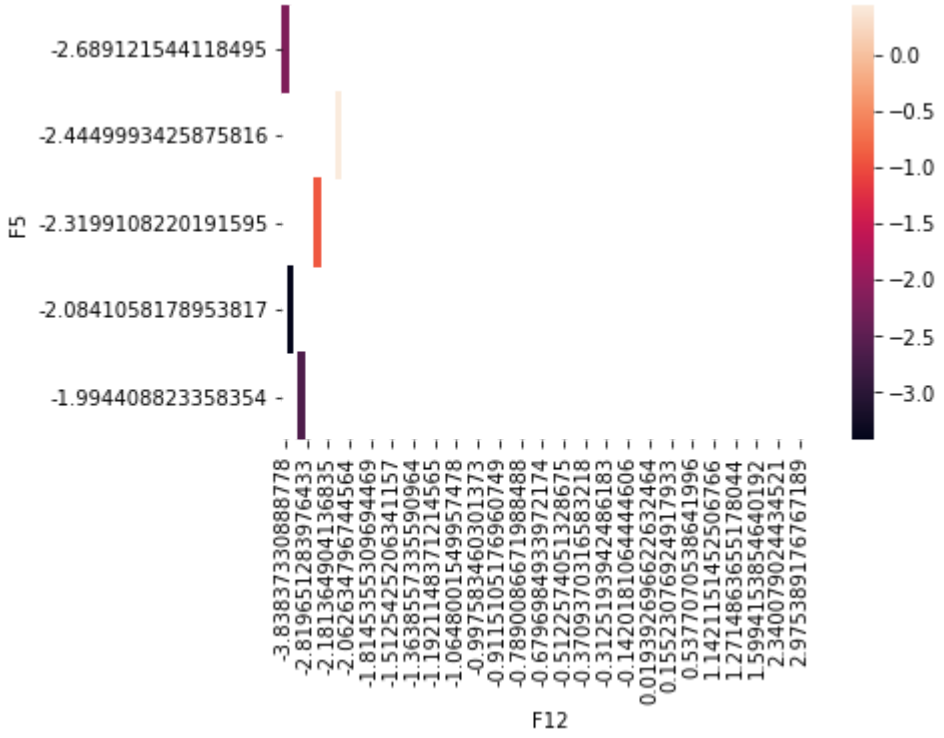
```
C:\Users\SRIDHAR\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: P
ass the following variables as keyword args: x, y. From version 0.12, the only valid pos
itional argument will be `data`, and passing other arguments without an explicit keyword
will result in an error or misinterpretation.
  warnings.warn(
```

Out[9]: <seaborn.axisgrid.JointGrid at 0x20c237ac670>



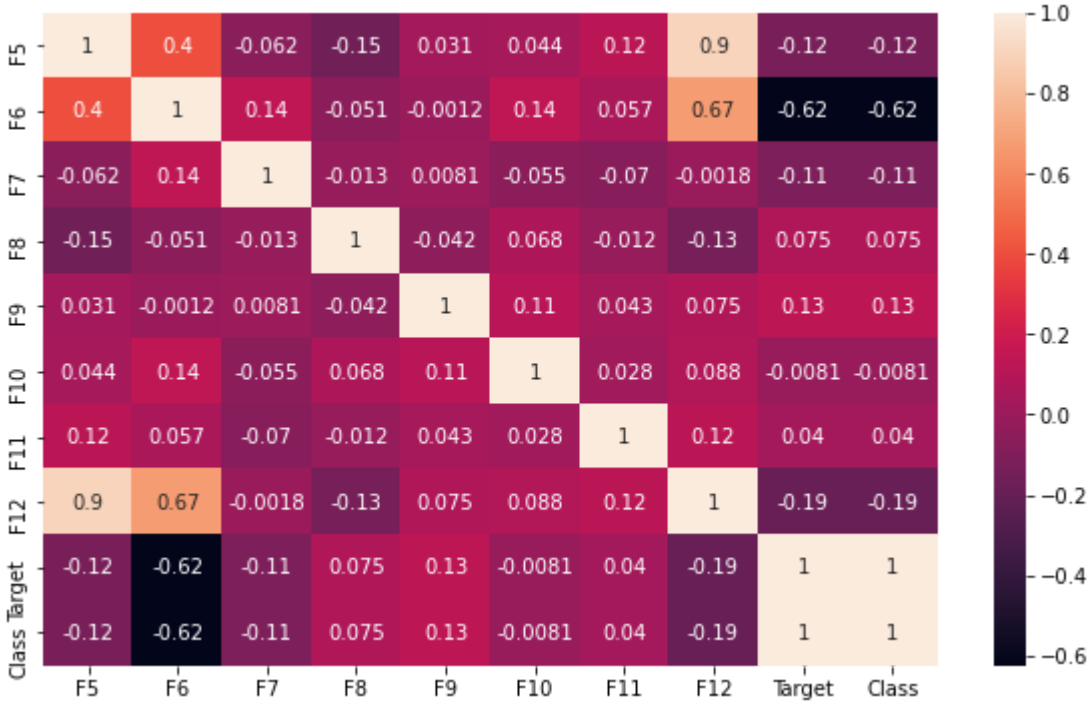
In [10]:
```
fig=plt.figure(figsize=(6,4))
data = df.pivot('F5','F12','F6')
sns.heatmap(data.head(5))
```

Out[10]: <AxesSubplot:xlabel='F12', ylabel='F5'>



In [11]:
```
fig=plt.figure(figsize=(10,6))
sns.heatmap(df.corr(),annot=True)
```

Out[11]: <AxesSubplot:>



kdeplot

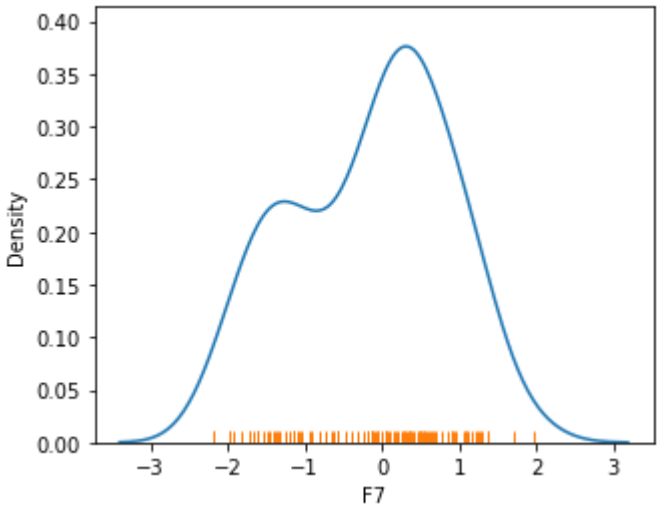
Show a univariate or bivariate distribution with a kernel density estimate.

rugplot

Draw small vertical lines to show each observation in a distribution.

In [12]:
```
ig=plt.figure(figsize=(5,4))
sns.kdeplot(data=df, x="F7")
sns.rugplot(data=df, x="F7")
```

Out[12]: <AxesSubplot:xlabel='F7', ylabel='Density'>



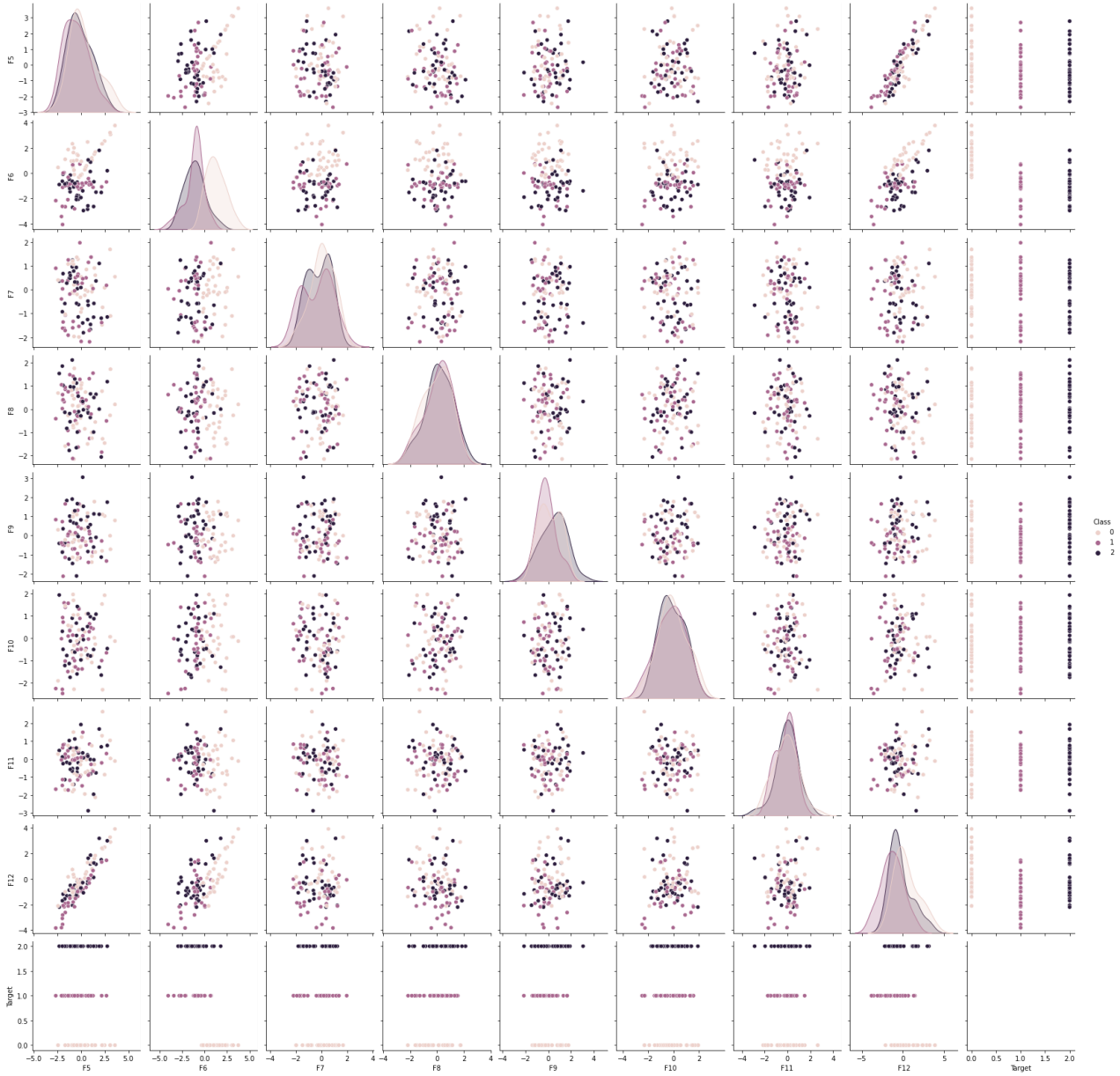
Pairplot:

To plot multiple pairwise bivariate distributions in a dataset, you can use the pairplot() function. This shows the relationship for (n, 2) combination of variable in a DataFrame as a matrix of plots and the diagonal plots are the univariate plots.

In [13]:
```
sns.pairplot(df, hue ='Class')
```

```
C:\Users\SRIDHAR\anaconda3\lib\site-packages\seaborn\distributions.py:306: UserWarning:
Dataset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\SRIDHAR\anaconda3\lib\site-packages\seaborn\distributions.py:306: UserWarning:
Dataset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\SRIDHAR\anaconda3\lib\site-packages\seaborn\distributions.py:306: UserWarning:
Dataset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
```

Out[13]: <seaborn.axisgrid.PairGrid at 0x20c23ca73a0>



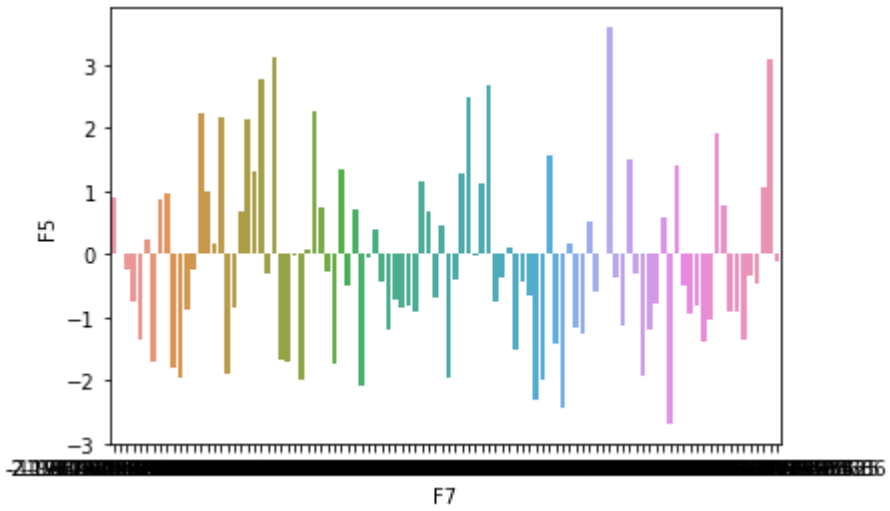
Barplot:

A bar plot or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent. The bar plots can be plotted horizontally or vertically. A bar chart describes the comparisons between the discrete categories. One of the axis of the plot represents the specific categories being compared, while the other axis represents the measured values corresponding to those categories.

In [14]:
```
sns.barplot('F7','F5',data=df)
```

```
C:\Users\SRIDHAR\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: P
ass the following variables as keyword args: x, y. From version 0.12, the only valid pos
itional argument will be `data`, and passing other arguments without an explicit keyword
will result in an error or misinterpretation.
  warnings.warn(
```

Out[14]:
```
<AxesSubplot:xlabel='F7', ylabel='F5'>
```



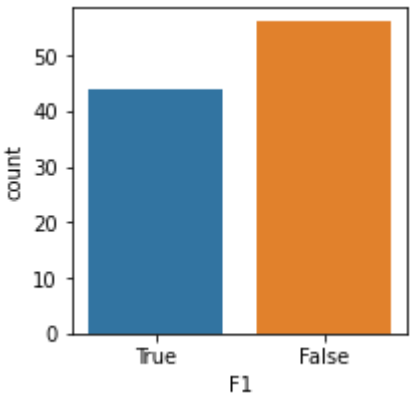
Countplot:

countplot() method is used to Show the counts of observations in each categorical bin using bars.It uses the concept of a bar chart for the visual depiction.

In [15]:
```
fig=plt.figure(figsize=(3,3))
sns.countplot(df.F1)
```

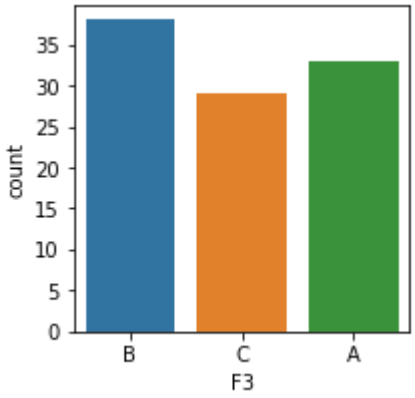
```
C:\Users\SRIDHAR\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: P
ass the following variable as a keyword arg: x. From version 0.12, the only valid positi
onal argument will be `data`, and passing other arguments without an explicit keyword wi
ll result in an error or misinterpretation.
  warnings.warn(
```

Out[15]: <AxesSubplot:xlabel='F1', ylabel='count'>



In [16]:
```
fig=plt.figure(figsize=(3,3))
sns.countplot(x="F3", data=df)
```

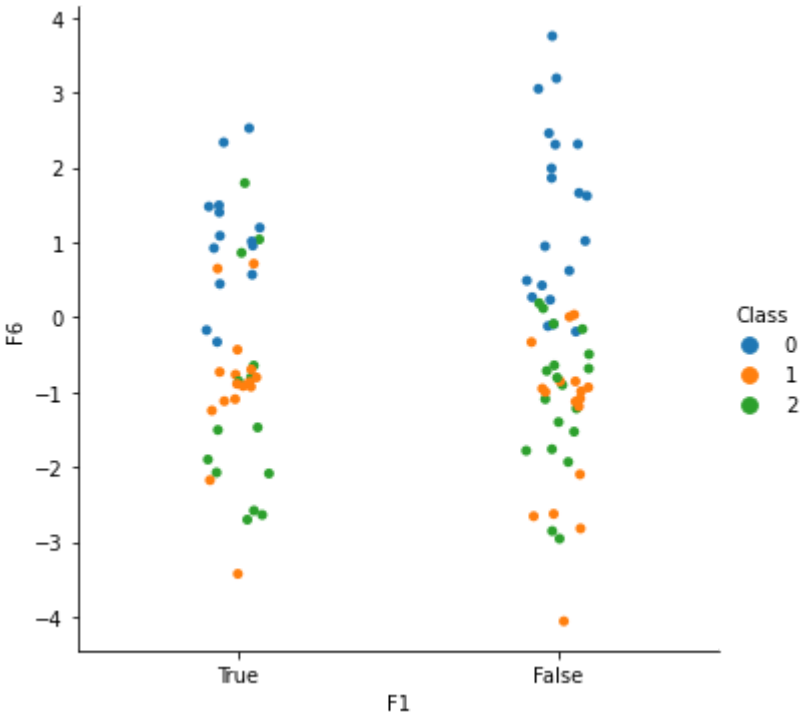
Out[16]: <AxesSubplot:xlabel='F3', ylabel='count'>

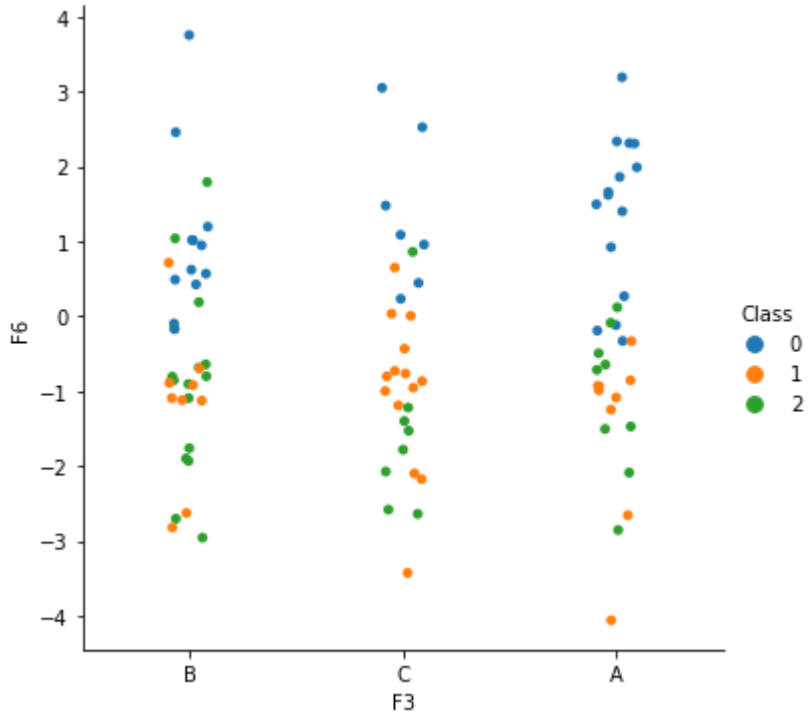


Catplot:

Figure-level interface for drawing categorical plots onto a FacetGrid. This function provides access to several axes-level functions that show the relationship between a numerical and one or more categorical variables using one of several visual representations.

In [17]:
```
sns.catplot(x="F1",y="F6", hue="Class", data=df)
sns.catplot(x="F3",y="F6", hue="Class", data=df)
plt.show()
```



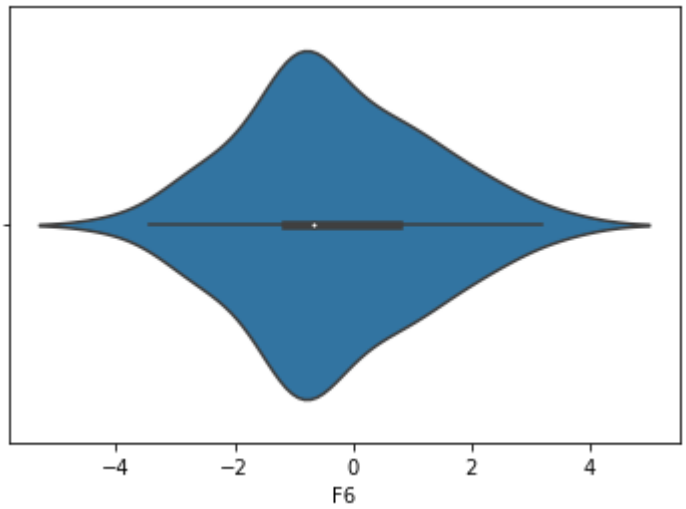


Violinplot:

Violin Plot is a method to visualize the distribution of numerical data of different variables. It is similar to Box Plot but with a rotated plot on each side, giving more information about the density estimate on the y-axis. The density is mirrored and flipped over and the resulting shape is filled in, creating an image resembling a violin. The advantage of a violin plot is that it can show nuances in the distribution that aren't perceptible in a boxplot. On the other hand, the boxplot more clearly shows the outliers in the data.

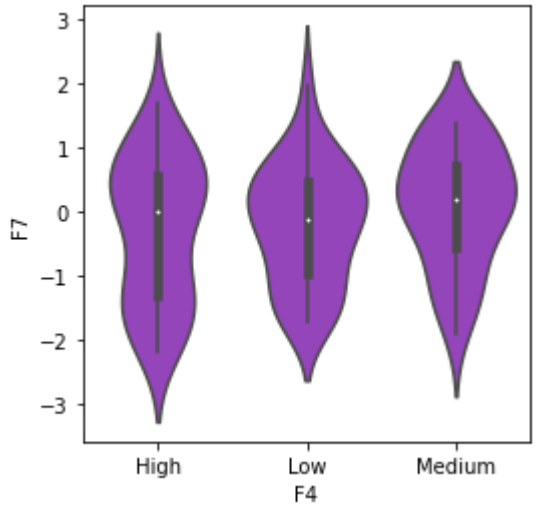
In [18]:
```
sns.violinplot(x=df["F6"])
```

Out[18]: <AxesSubplot:xlabel='F6'>



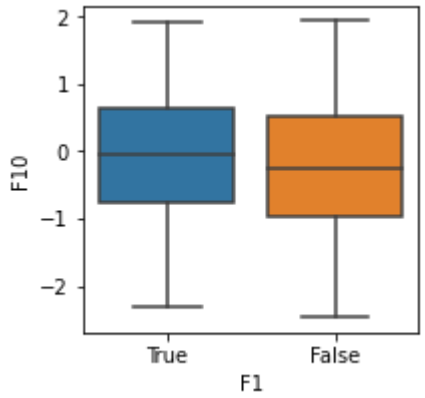
In [19]:
```
fig=plt.figure(figsize=(4,4))
sns.violinplot(x="F4",y="F7",  data=df,color="darkorchid")
```

Out[19]: <AxesSubplot:xlabel='F4', ylabel='F7'>



In [20]:
```
fig=plt.figure(figsize=(3,3))
sns.boxplot(x="F1", y="F10", data=df)
```

Out[20]: <AxesSubplot:xlabel='F1', ylabel='F10'>

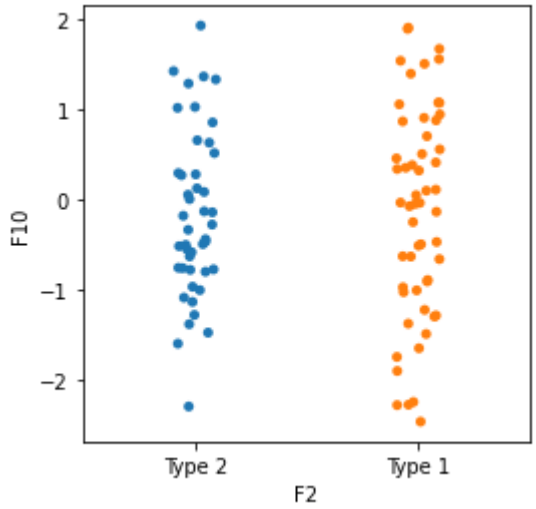


Striplot:

A strip plot is drawn on its own. It is a good complement to a boxplot or violinplot in cases where all observations are shown along with some representation of the underlying distribution. It is used to draw a scatter plot based on the category.

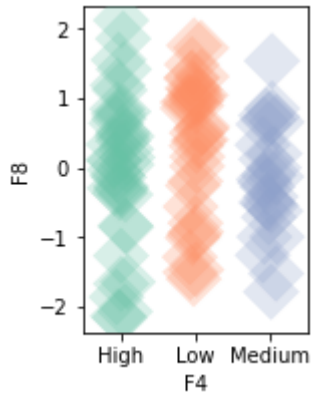
In [21]:
```
fig=plt.figure(figsize=(4,4))
sns.stripplot(x="F2", y="F10", data=df)
```

Out[21]: <AxesSubplot:xlabel='F2', ylabel='F10'>



In [22]:
```
fig=plt.figure(figsize=(2,3))
sns.stripplot(x="F4",y="F8", data=df,palette="Set2", size=20, marker="D", edgecolor="gr
```

Out[22]: <AxesSubplot:xlabel='F4', ylabel='F8'>

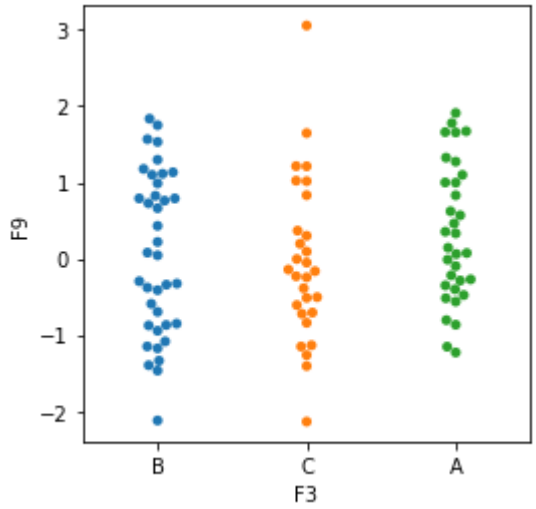


Swarmplot :

Seaborn swarmplot is probably similar to stripplot, only the points are adjusted so it won't get overlap to each other as it helps to represent the better representation of the distribution of values. A swarm plot can be drawn on its own, but it is also a good complement to a box, preferable because the associated names will be used to annotate the axes. This type of plot sometimes known as "beeswarm".

In [23]:
```
fig=plt.figure(figsize=(4,4))
sns.swarmplot(x="F3", y="F9", data=df)
```

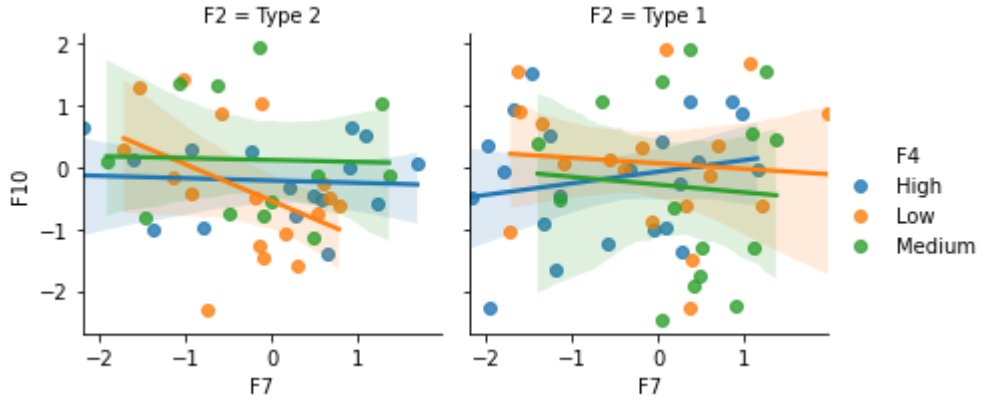
Out[23]: <AxesSubplot:xlabel='F3', ylabel='F9'>



Facetgrid Plot:

FacetGrid class helps in visualizing distribution of one variable as well as the relationship between multiple variables separately within subsets of your dataset using multiple panels.FacetGrid object takes a dataframe as input and the names of the variables that will form the row, column, or hue dimensions of the grid. The variables should be categorical and the data at each level of the variable will be used for a facet along that axis.

In [24]:
```
graph = sns.FacetGrid(df, col ='F2', hue ='F4')
graph.map(sns.regplot, "F7", "F10").add_legend()
plt.show()
```



PART C

Load iris dataset and explore

In [25]:
```
from sklearn.datasets import load_iris
```

In [26]:
```
iris = load_iris(as_frame=True)
iris['data']
```

Out[26]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

In [27]:
```
iris.feature_names
```

Out[27]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [28]:
```
iris.target
```

Out[28]:
```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: target, Length: 150, dtype: int32
```

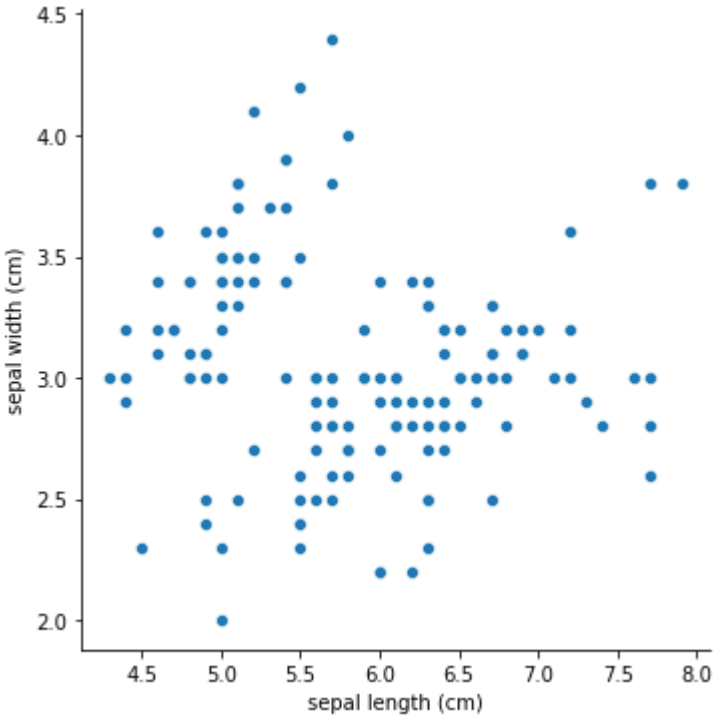
Relplot:

What is Relplot in seaborn? relplot() This function provides us the access to some other different axes-level functions which shows the relationships between two variables with semantic mappings of subsets.
Syntax : seaborn.relplot(x=None, y=None, data=None, **kwargs)

In [29]:
```
sns.relplot(x=iris['data']['sepal length (cm)'],y =iris['data']['sepal width (cm)'] , d
```

```
<seaborn.axisgrid.FacetGrid at 0x20c27958340>
```

Out[29]:

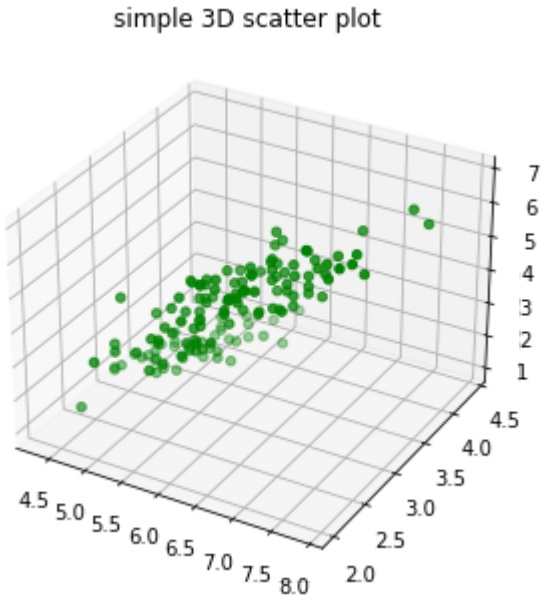


ScatterPlot:

Scatter plots are used to observe relationship between variables and uses dots to represent the relationship between them. The scatter() method in the matplotlib library is used to draw a scatter plot. Scatter plots are widely used to represent relation among variables and how change in one affects the other.

In [30]:

```
# Creating dataset
z = iris['data']['petal length (cm)']
x = iris['data']['sepal length (cm)']
y = iris['data']['sepal width (cm)']
# Creating figure
fig = plt.figure(figsize = (10, 5))
ax = plt.axes(projection ="3d")
# Creating plot
ax.scatter3D(x, y, z, color = "green")
plt.title("simple 3D scatter plot")
plt.show()
```

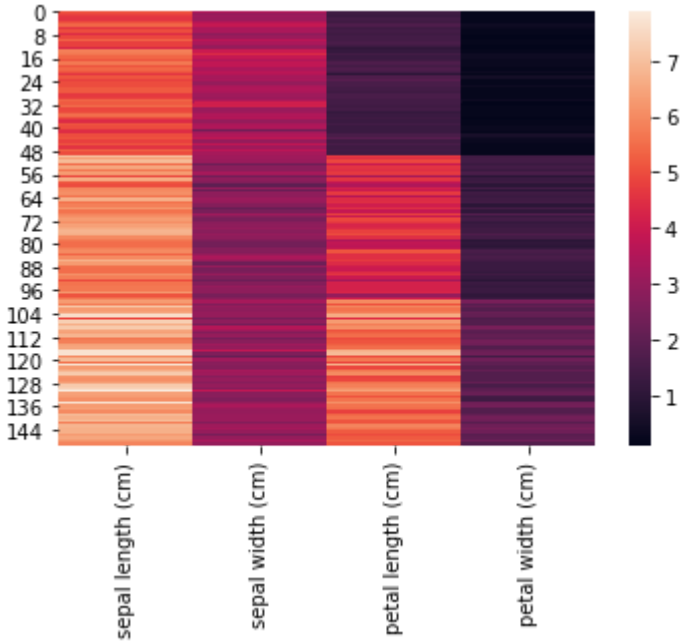


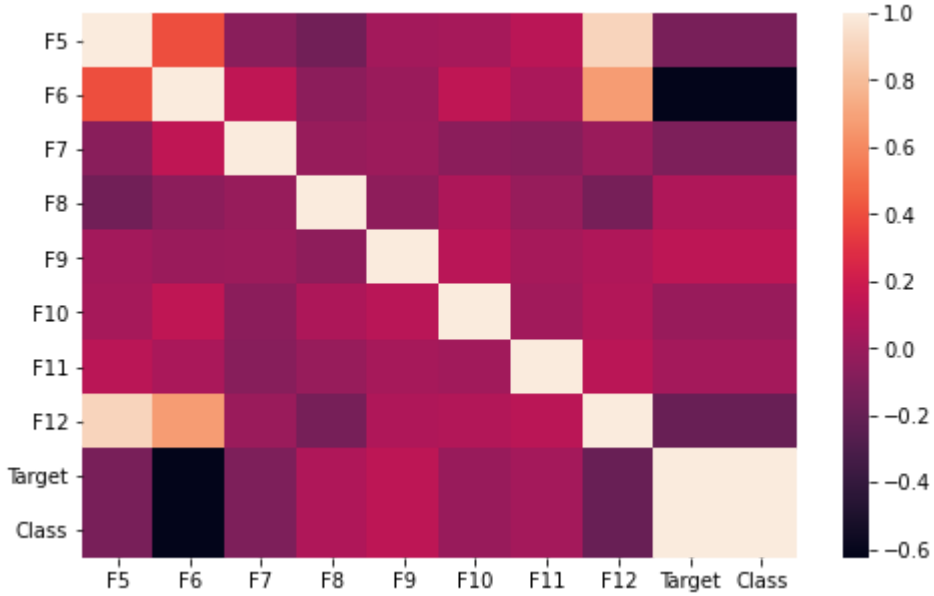
Heatmap:

A heatmap contains values representing various shades of the same colour for each value to be plotted. Usually the darker shades of the chart represent higher values than the lighter shade. For a very different value a completely different colour can also be used.

In [31]:
```
hm = sns.heatmap(data = iris['data'],annot_kws={"size": 10})
plt.figure(figsize = (8,5))
sns.heatmap(df.corr())
# displaying the plotted heatmap
```

Out[31]: <AxesSubplot:>

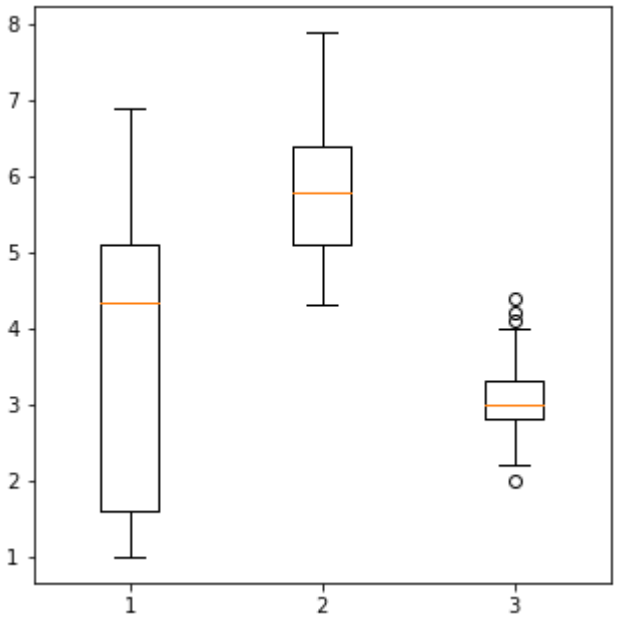




Boxplot:

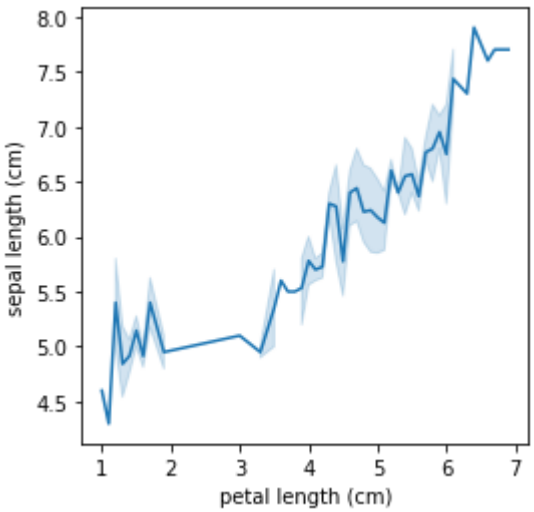
A Box Plot is also known as Whisker plot is created to display the summary of the set of data values having properties like minimum, first quartile, median, third quartile and maximum. In the box plot, a box is created from the first quartile to the third quartile, a vertical line is also there which goes through the box at the median. Here x-axis denotes the data to be plotted while the y-axis shows the frequency distribution.

In [32]:
```
data = [iris['data']['petal length (cm)'],iris['data']['sepal length (cm)'],iris['data'
fig = plt.figure(figsize =(4, 4))
# Creating axes instance
ax = fig.add_axes([0, 0, 1, 1])
# Creating plot
bp = ax.boxplot(data)
plt.show()
```



In [33]:
```
fig=plt.figure(figsize=(4,4))
sns.lineplot(data=iris['data'], x=iris['data']['petal length (cm)'], y=iris['data']['se
```

Out[33]:
```
<AxesSubplot:xlabel='petal length (cm)', ylabel='sepal length (cm)'>
```

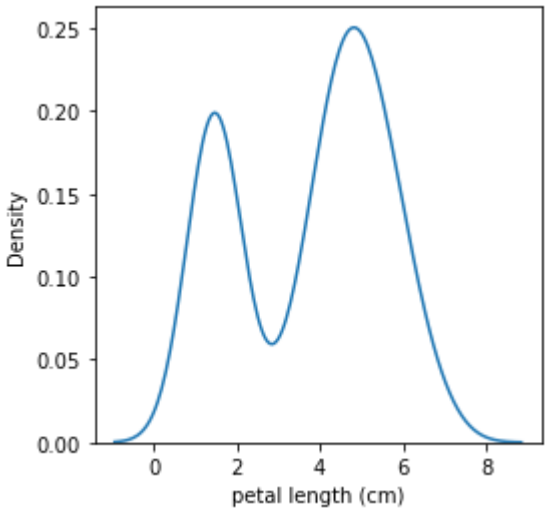


Kdeplot:

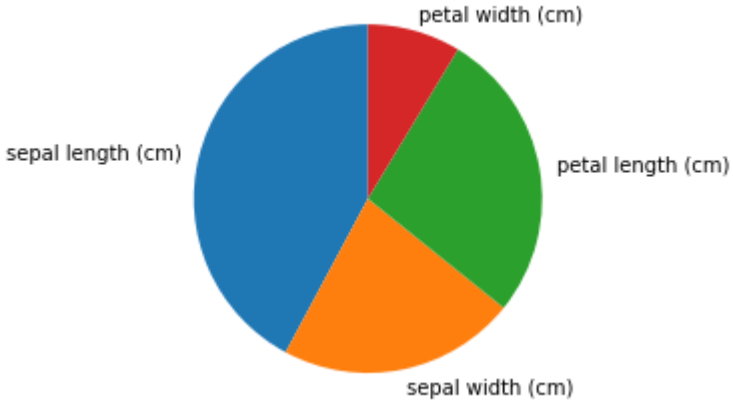
KDE Plot described as Kernel Density Estimate is used for visualizing the Probability Density of a continuous variable. It depicts the probability density at different values in a continuous variable. We can also plot a single graph for multiple samples which helps in more efficient data visualization.

In [34]:
```
fig=plt.figure(figsize=(4,4))
sns.kdeplot(data=iris['data'], x=iris['data']['petal length (cm)'])
```

Out[34]:
```
<AxesSubplot:xlabel='petal length (cm)', ylabel='Density'>
```

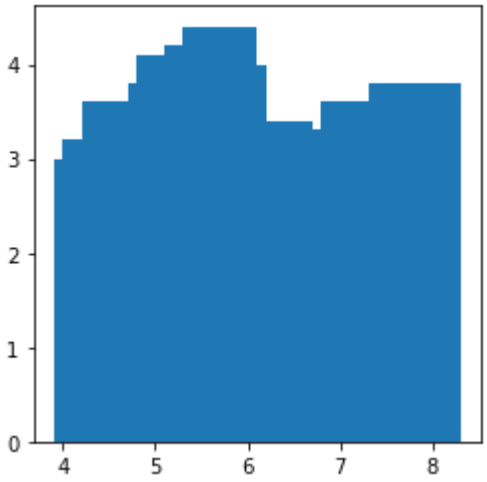


In [35]:
```
fig=plt.figure(figsize=(4,4))
y = np.array([iris['data']['sepal length (cm)'].mean(),iris['data']['sepal width (cm)']
mylabels = iris['data'].columns
plt.pie(y, labels = mylabels, startangle = 90)
plt.show()
```



In [36]:
```
fig = plt.figure(figsize =(4, 4))
plt.bar(iris['data']['sepal length (cm)'],iris['data']['sepal width (cm)'])
```

Out[36]: <BarContainer object of 150 artists>



CONCLUSION:

As instructed in the objectives, we have imported the necessary libraries, Created and explored the datasets. Explored more about the various visualisation techniques and EDA.