Before you turn this problem in, make sure everything runs as expected. First, **restart the kernel** (in the menu bar, select Kernel$\rightarrow$Restart) and then **run all cells** (in the menu bar, select Cell$\rightarrow$Run All).

Below, please fill in your name and collaborators, if any:

In [1]:
```
NAME = "Jordan Vercillo"
COLLABORATORS = "Jordan Vercillo"
```

Assignment 4 - Classification

In this assignment, you will practice using the kNN (k-Nearest Neighbors) algorithm to solve a classification problem. The kNN is a simple and robust classifier, which is used in different applications.

We will use the Iris dataset for this assignment. The dataset was first introduced by statistician R. Fisher and consists of 50 observations from each of three species Iris (*Iris setosa*, *Iris virginica* and *Iris versicolor*). For each sample, 4 features are given: the sepal length and width, and the petal length and width.

The goal is to train kNN algorithm to distinguish the species from one another.

1. The dataset can be downloaded from UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/machine-learning-databases/iris/.
2. Download `iris.data` file from the Data Folder. The Data Set description with the definitions of all the columns can be found on the dataset page - https://archive.ics.uci.edu/ml/datasets/Iris. Alternatively, you can import the data using sklearn.datasets. You will need to dowload both the sepal/petal data and the target variable information, then merge the two datasets.
3. *(1 points)* Load the data from the file (`iris.data`) into the DataFrame. Set the names of columns according to the column definitions given in Data Description.
4. *(2 points)* **Data inspection.**
   - Display the first 5 rows of the dataset and use any relevant functions that can help you to understand the data.
   - Prepare 2 scatter plots - `sepal_width` vs `sepal_length` and `petal_width` vs `petal_length`. Scatter plots should show each class in different color (`seaborn.lmplot` is recommended for plotting).
5. *(2 points)* **Prepare the data for classification**.

- Using the pandas operators prepare the feature variables `X` and the response `Y` for the fit. Note that `sklean` expects data as arrays, so convert extracted columns into arrays.

6. *(1 point)* **Split** the data into `train` and `test` using `sklearn` `train_test_split` function.

7. *(2 points)* **Run the fit** using `KNeighborsClassifier` from `sklearn.neighbors`.

   - First, instantiate the model,
   - Then, run the classifier on the training set.

8. *(3 points)* Use learning model to **predict the class from features**, run prediction on `X` from test part.

   - Show the **accuracy score** of the prediction by comparing predicted iris classes and the `Y` values from the test.
   - Comparing these two arrays (predicted classes and test `Y`), count the numbers of correct predictions and predictions that were wrong. (**HINTS:** `NumPy` arrays can be compared using `==` operator. You can also use `NumPy`'s operator `count_nonzero` to count number of non-False values).

9. *(4 points)* In this task, we want to see how accuracy score and the number of correct predictions change with the number of neighbors `k`. We will use the following **number of neighbors `k`: 1, 3, 5, 7, 10, 20, 30, 40, and 50**:

   - Generate 10 random train/test splits for each value of `k`
   - Fit the model for each split and generate predictions
   - Average the accuracy score for each `k`
   - Calculate the average number of correct predictions for each `k` as well
   - Plot the accuracy score for different values of `k`. What conclusion can you make based on the graph?

In [2]:
```
# Here are all imports that you will need

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
```

In [11]:
```
# Data download from sklearn
from sklearn.datasets import load_iris
data=load_iris().data
target=load_iris().target
df_data=pd.DataFrame(data,columns=['sepal_length','sepal_width','petal_length','pet
df_target=pd.DataFrame(target,columns=['class'])

# Remember to merge the DataFrames into one after they are created.
```

In [13]:
```
df = pd.concat([df_data,df_target],axis=1)

df
```

Out[13]:

	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   Class         150 non-null    int32  
dtypes: float64(4), int32(1)
memory usage: 5.4 KB
```

In [6]:
```
df.describe()
```

Out[6]:

	sepal_length	sepal_width	petal_length	petal_width	Class
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

In [17]:

```
df.head()
```

Out[17]:

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

In [16]:

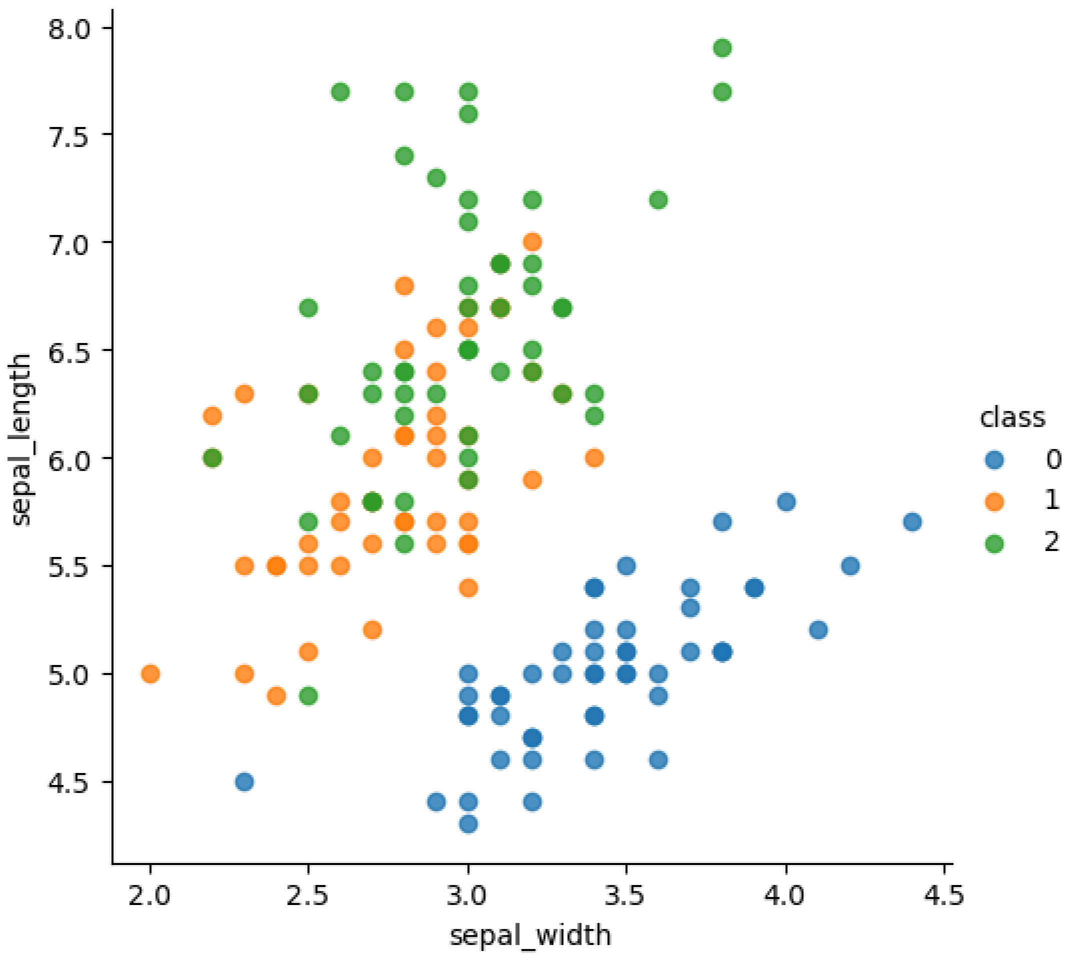
```
import seaborn as sns
```

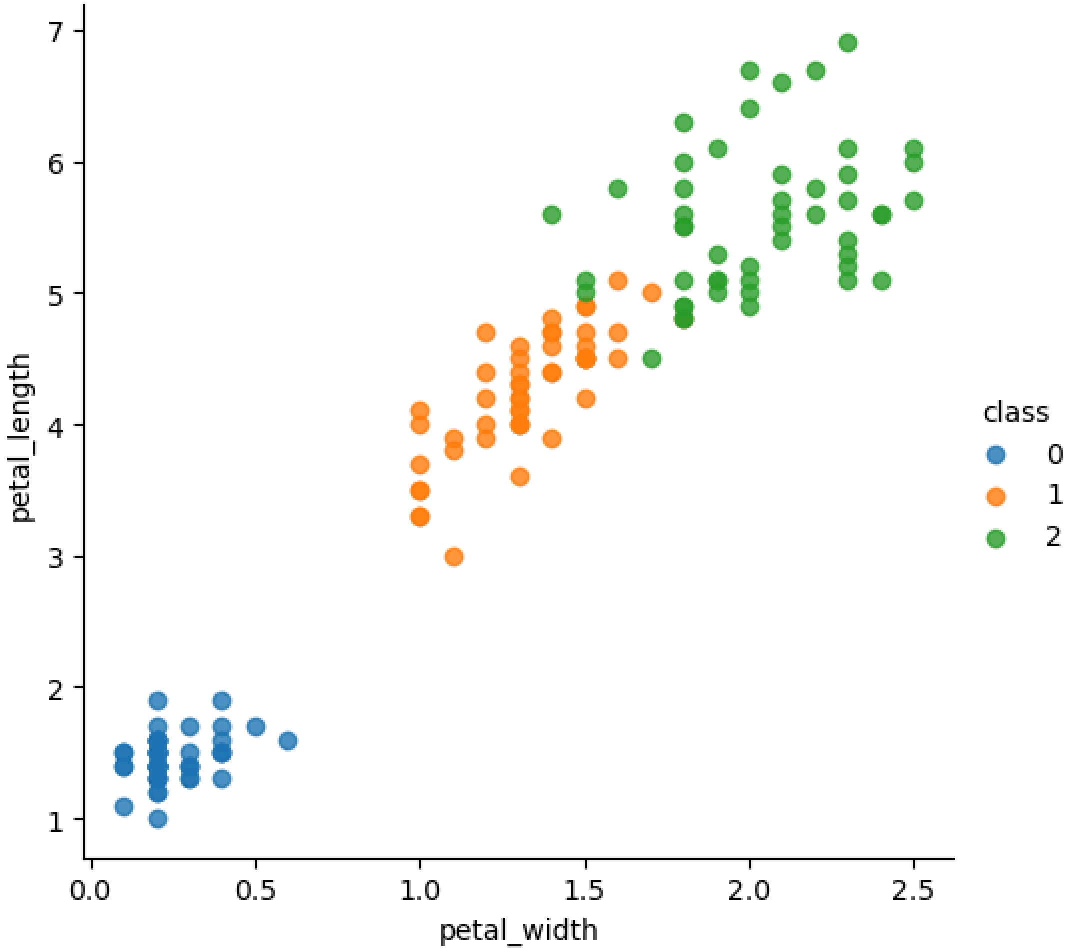
In [14]:

```
sns.lmplot(x='sepal_width', y='sepal_length', data=df, hue='class', fit_reg=False)
sns.lmplot(x='petal_width', y='petal_length', data=df, hue='class', fit_reg=False)
```

Out[14]:

```
<seaborn.axisgrid.FacetGrid at 0x1b240486090>
```


8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
sepal_length
2.0
2.5
3.0
3.5
4.0
4.5
sepal_width
class
0
1
2



In [33]:
```
X = df[['sepal_length','sepal_width','petal_length','petal_width']]
y = df['class']
```

In [34]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

In [35]:
```
%%time
from sklearn.neighbors import KNeighborsClassifier
kn_clf = KNeighborsClassifier(n_neighbors=5) #create an instance of the model
kn_clf.fit(X_train, y_train)
```

```
CPU times: total: 0 ns
Wall time: 499 µs
```

Out[35]:
```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [37]:
```
y_pred = kn_clf.predict(X_test) #make predictions for the test set
print(y_pred.shape)
y_pred #these are the predictions made by the model
```

```
(30,)
```

Out[37]:
```
array([1, 2, 1, 2, 2, 2, 0, 1, 2, 0, 1, 0, 2, 2, 0, 2, 2, 1, 2, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 1, 0])
```

In [38]:
```
#compare y_pred to our actual y_test
(y_pred == y_test).sum()
```

Out[38]:
```
28
```

In [39]:
```
# pro way:
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_pred))
```

```
0.9333333333333333
```

In [40]:
```
y_test.value_counts()
```

Out[40]:
```
class
0    12
2    10
1     8
Name: count, dtype: int64
```

In [ ]:

In [ ]:

In [ ]: