About our Dataset

The iris dataset is a simple and beginner-friendly dataset that contains information about the flower petal and sepal sizes. The dataset has 3 classes with 50 instances in each class, therefore, it contains 150 rows with only 4 columns.

Our Target

Implement a machine learning classification or regression model on the dataset. Classification is the task of separating items into its corresponding class.

Importing all the important libray

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

Now reading our .csv file using Pandas Library

In [8]:
```
df = pd.read_csv('iris.data' , header = None)
df
```

Out[8]:

	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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

Giving the datdset its coloumns names

In [12]:
```
names = ['sepal length','sepal width','petal length','petal width' , 'class']
```

In [95]:
```
df['class'].unique()
```

Out[95]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [14]:
```
df.columns = names
df
```

Out[14]:

	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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

Checking for Null Values or missing Values in our Dataset

In [29]:
```
df.isna().sum()
```

Out[29]:
```
sepal length    0
sepal width     0
petal length    0
petal width     0
class           0
dtype: int64
```

In [30]:
```
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    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [31]:
```
df.describe()
```

Out[31]:

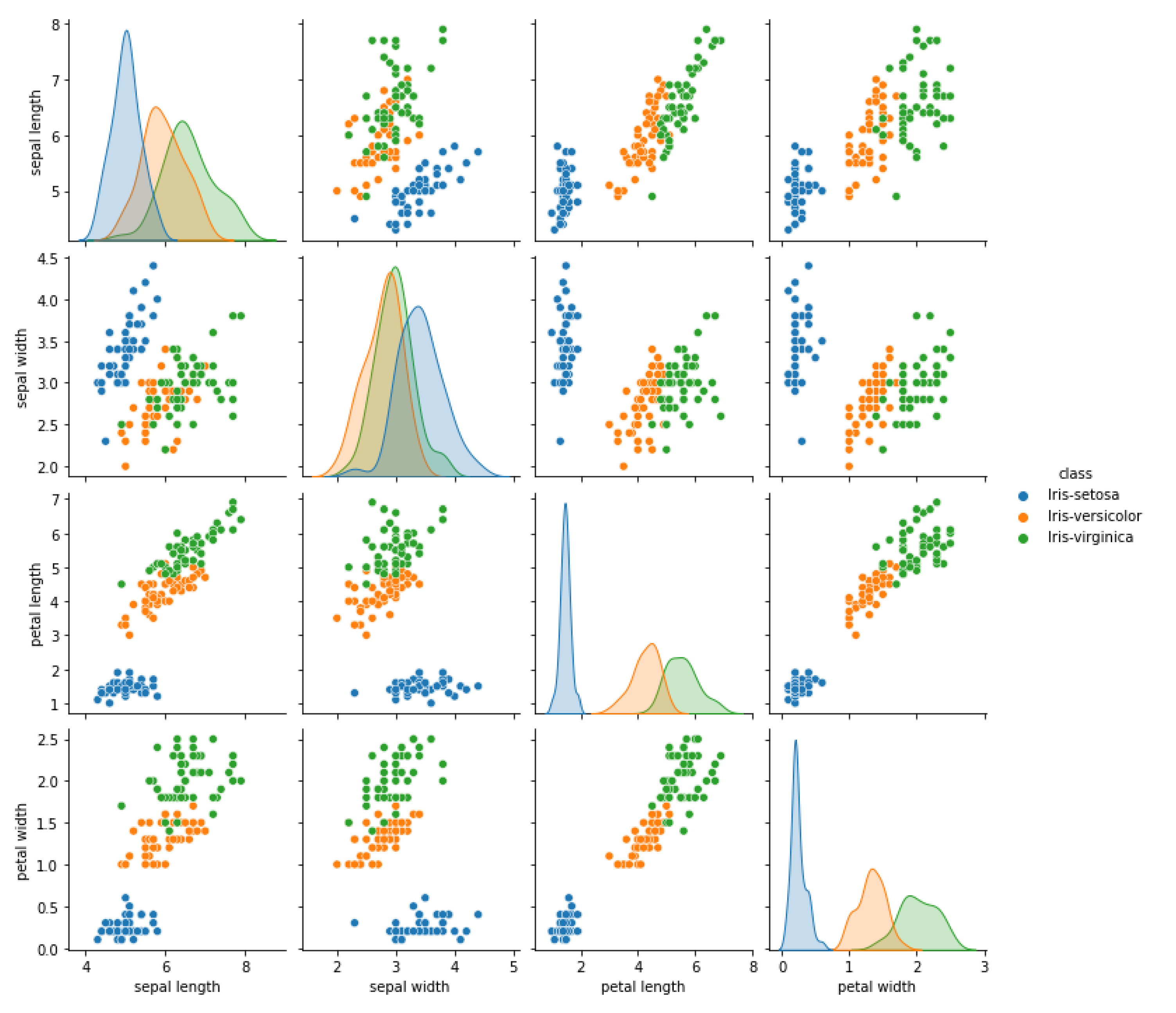
	sepal length	sepal width	petal length	petal width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000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

Plotting all the features against all other features.

We do this to get the main attributes that should be used for classifications

In [16]:
```
sns.pairplot(data = df ,hue ='class')
```

Out[16]:
```
<seaborn.axisgrid.PairGrid at 0x123d87fd7e0>
```



Getting the 'Correaltion' between all features

In [18]:
```
df.corr()
```

Out[18]:

	sepal length	sepal width	petal length	petal width
sepal length	1.000000	-0.109369	0.871754	0.817954
sepal width	-0.109369	1.000000	-0.420516	-0.356544
petal length	0.871754	-0.420516	1.000000	0.962757
petal width	0.817954	-0.356544	0.962757	1.000000

Ploting the Correlation using Heatmap

Selecting our X and Y data

In [38]:
```
y = df.iloc[:,-1]
```

Out[38]:
```
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
            ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: class, Length: 150, dtype: object
```

Transforming the Class or ['Class'] coloumn into numeric data.

We are using LabelEncoding for this

In [78]:
```
from sklearn.preprocessing import LabelEncoder

LE = LabelEncoder()
y = LE.fit_transform(y)
y
```

Out[78]:
```
array([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], dtype=int64)
```

Splitting our dataset into training and testing data

In [41]:
```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(X,y,random_state = 5)
```

In [44]:
```
x_train[:5]
```

Out[44]:
```
array([[5. , 3.5, 1.3, 0.3],
       [6.4, 3.2, 5.3, 2.3],
       [5.8, 2.7, 5.1, 1.9],
       [5.6, 2.5, 3.9, 1.1],
       [5.1, 3.5, 1.4, 0.3]])
```

Fitting our data in the model

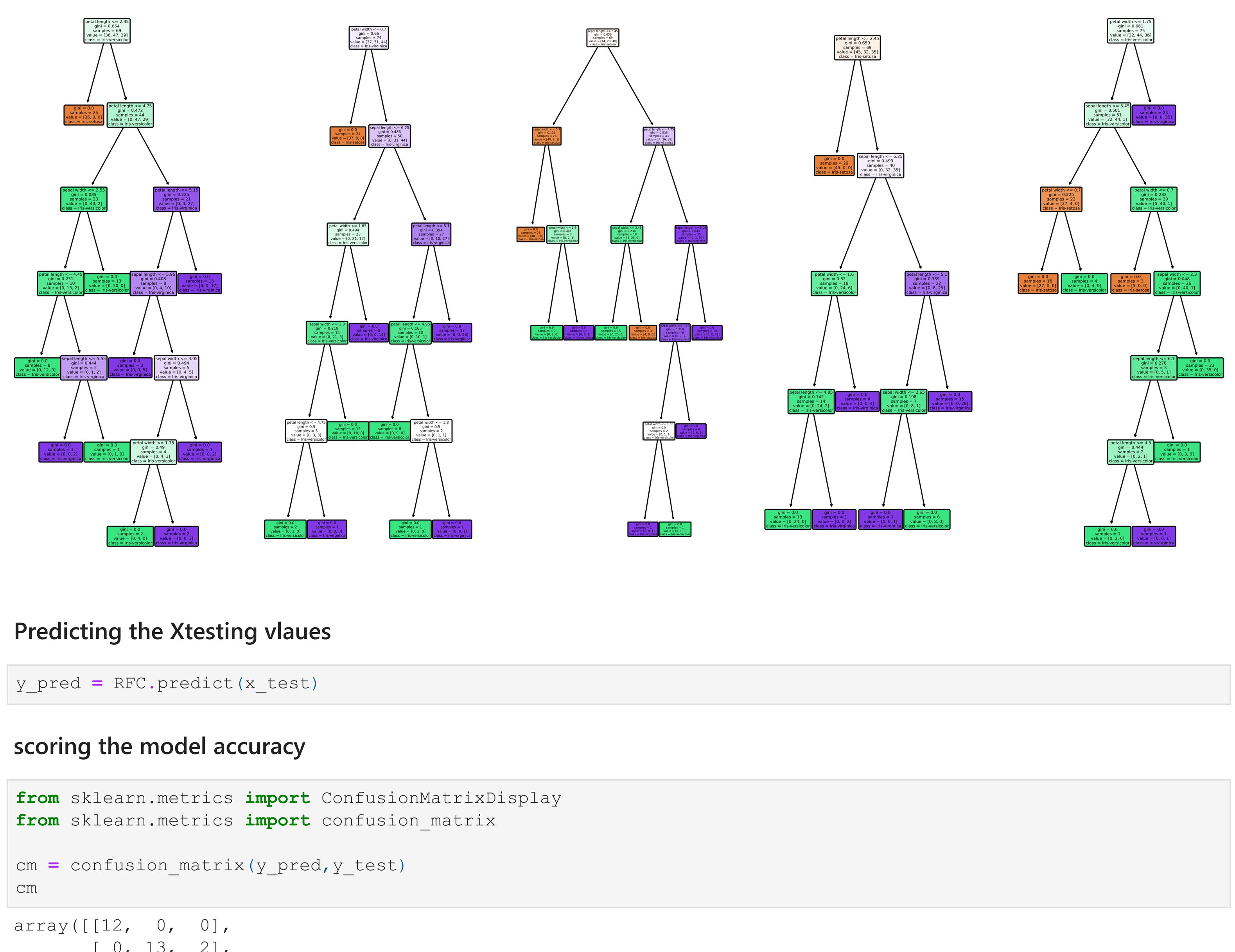
In [45]:
```
from sklearn.ensemble import RandomForestClassifier

RFC = RandomForestClassifier()
RFC.fit(x_train , y_train)
```

Out[45]:
```
RandomForestClassifier()
```

In [102...
```
from sklearn.tree import export_graphviz
import graphviz
```

Plotting the RandomForestRegressor trees

In [146...
```
from sklearn import tree

#fig = plt.figure(figsize=(10,8))
fig, axes = plt.subplots(nrows = 1,ncols = 5,figsize = (20,10) , dpi = 900)
a = 0
for i in range(0,100,20):

    tree.plot_tree(RFC.estimators_[i],   feature_names = names[:-1],class_names= df['class'].unique(),
              filled=True, impurity=True,rounded=True , ax = axes[a])

    axes[a].set_title('Estimators -: ' + str(i), fontsize = 11)
    a = a + 1



fig.savefig('rf_5trees.png')
```



Predicting the Xtesting vlaues

In [56]:
```
y_pred = RFC.predict(x_test)
```

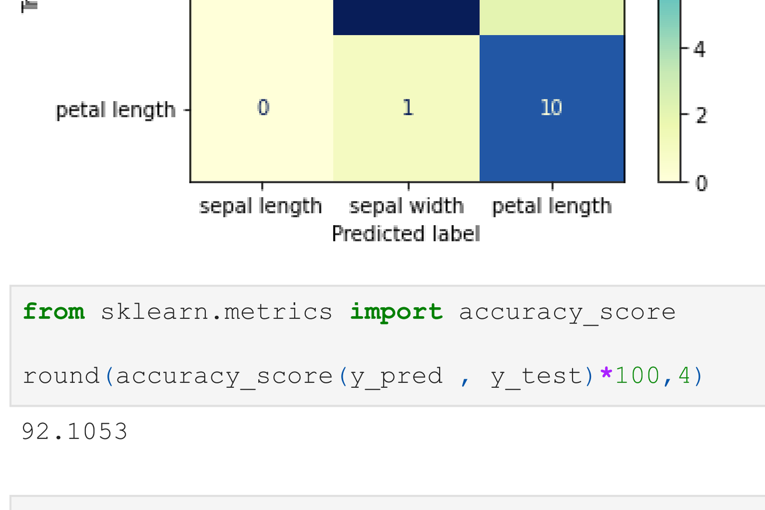
scoring the model accuracy

In [57]:
```
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_pred,y_test)
cm
```

Out[57]:
```
array([[12,  0,  0],
       [ 0, 13,  2],
       [ 0,  1, 10]], dtype=int64)
```

In [72]:
```
ConfusionMatrixDisplay.from_predictions(y_pred ,y_test , cmap = 'YlGnBu' ,display_labels = names[:-2] )
```

Out[72]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x123e6f5beb0>
```



In [77]:
```
from sklearn.metrics import accuracy_score

round(accuracy_score(y_pred , y_test)*100,4)
```

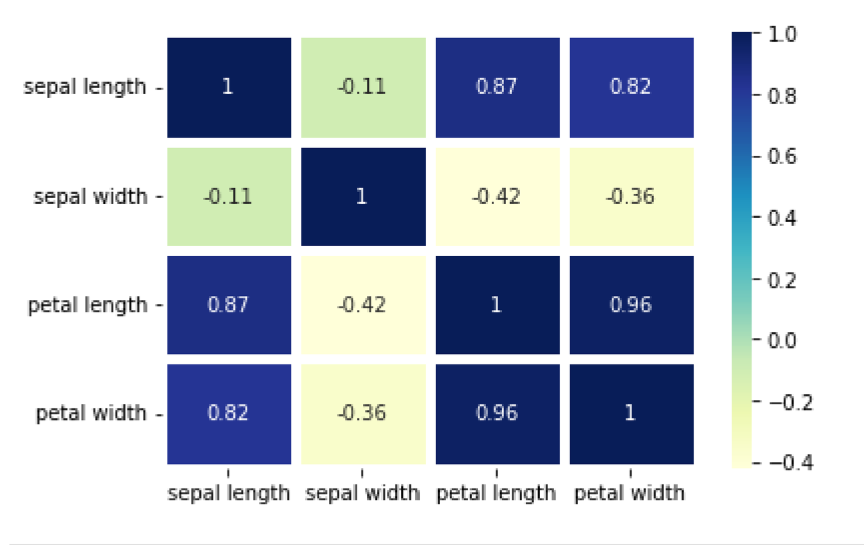
Out[77]:
```
92.1053
```

In [ ]:

In [ ]:

In [28]:
```
sns.heatmap(df.corr(), cmap = 'YlGnBu' , linewidths = 5 , annot= True)
```

Out[28]:
```
<AxesSubplot:>
```



In [36]:
```
X = df.iloc[:,:-1]
X = X.values
```