Machine Learning Assignment_1

Arya Chakraborty [22MSD7020]

Questions :

1. Create arrays, Array Indexing, Slicing, Shape, Reshape Iterating, Random() using Numpy Python Packages
2. Visualize Line plot, Bar Plot, Scatter plot, Pie Chart using Matplotlib python package
3. Creating DataFrames, Reading .CSV files, Data Cleaning(eg. Finding missing values, Null values, Empty cells, Wrong Format, Wrong Data, Removing Duplicates) using Pandas python library

Ans:

Question 1

Importing Libraries

```
import numpy as np
```

creating 1D array

```
data=[1,2,3,4,5]
data_array=np.array(data)
print(type(data_array))
print("<---------->")
print(data_array)
```

✓ 0.0s

```
<class 'numpy.ndarray'>
<---------->
[1 2 3 4 5]
```

2D Array

```
data2=[[1,2],
[3,4],
[5,6]]
data_array2=np.array(data2)
print(type(data_array2))
```

✓ 0.0s

```
<class 'numpy.ndarray'>
```

Array Indexing

```python
print(data_array)
print("<---------->")
print(data_array[0])
print("<---------->")
print(data_array[3])
```

✓ 0.0s

```
[1 2 3 4 5]
<---------->
1
<---------->
4
```

```python
print(data_array2)
print("<---------->")
print(data_array2[0,0])
print("<---------->")
print(data_array2[1,1])
print("<---------->")
print(data_array2[1,0])
print("<---------->")
print(data_array2[0,1])
```

✓ 0.0s

```
[[1 2]
 [3 4]
 [5 6]]
<---------->
1
<---------->
4
<---------->
3
<---------->
2
```

Array Slicing

```python
print(data_array2[:,:1])
print("<---------->")
print(data_array2[:,-1:])
print("<---------->")
print(data_array2[:1,:])
print("<---------->")
print(data_array2[:2,:])
print("<---------->")
print(data_array2[2:,:])
```

✓ 0.0s

```
[[1]
 [3]
 [5]]
<---------->
[[2]
 [4]
 [6]]
<---------->
[[1 2]]
<---------->
[[1 2]
 [3 4]]
<---------->
[[5 6]]
```

Array Reshaping

1D to 2D

```python
# 1D ----> 2D
new_array=data_array.reshape(data_array.shape[0],1)
print(data_array,"and the shape is ",data_array.shape)
print("<---------->")
print(new_array,"and the shape is ",new_array.shape)
print("<---------->")
```

✓ 0.0s

```
[1 2 3 4 5] and the shape is  (5,)
<---------->
[[1]
 [2]
 [3]
 [4]
 [5]] and the shape is  (5, 1)
<---------->
```

2D to 3 D

```python
# 2D ------> 3D
new_array2=data_array2.reshape(data_array2.shape[0],data_array2.shape[1],1)
print(data_array2,"and the shape is ",data_array2.shape)
print("<---------->")
print(new_array2,"and the shape is ",new_array2.shape)
```

✓ 0.0s

```
[[1 2]
 [3 4]
 [5 6]] and the shape is  (3, 2)
<---------->
[[[1]
  [2]]

 [[3]
  [4]]

 [[5]
  [6]]] and the shape is  (3, 2, 1)
```

Random Function

```python
x=np.random.rand(5,2)
y=np.random.rand(4,4)
print(x)
print("<---------->")
print(y)
```

✓ 0.0s

```
[[0.92424953 0.65105919]
 [0.12031471 0.01388421]
 [0.33886668 0.085324  ]
 [0.56247457 0.19704273]
 [0.55503197 0.75023352]]
<---------->
[[0.21839891 0.9511267  0.58939008 0.73572634]
 [0.98483409 0.40575368 0.47182794 0.5579839 ]
 [0.33102127 0.13528495 0.57798673 0.80669553]
 [0.87559886 0.7808209  0.12983377 0.97113816]]
```

various use cases of random() function in python.

```
# for printing random value
a=np.random.randint(5)
a
✓ 0.0s
```

```
3
```

```
# it will give a 3 * 2 matrix where all values are integers and less than 5
b=np.random.randint(5, size=(3,2))
b
✓ 0.0s
```

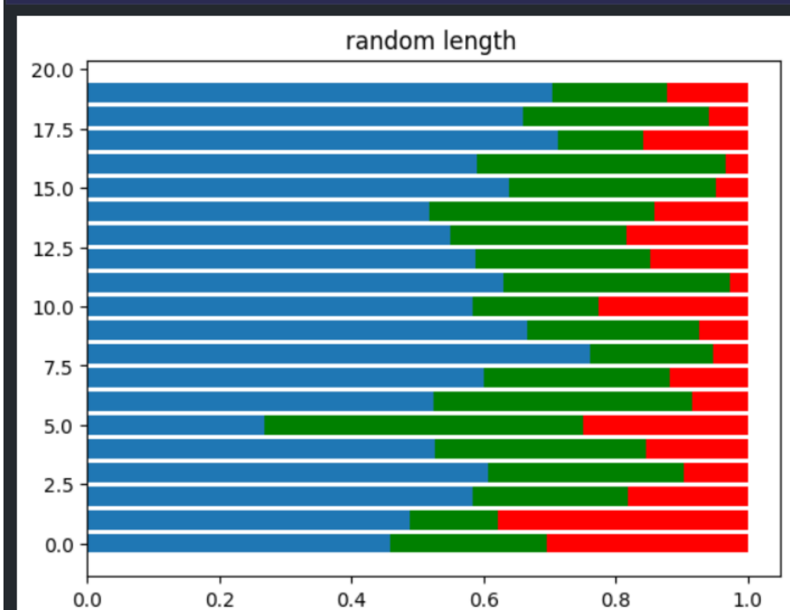
```
array([[4, 3],
       [3, 1],
       [4, 0]])
```

```
c=np.random.random((5,))
c
✓ 0.0s
```

```
array([0.70213784, 0.87623065, 0.56195869, 0.68284808, 0.11044867])
```

 # Dirichlet distribution

```
import matplotlib.pyplot as plt
s1 = np.random.dirichlet((10, 5, 3), 20).transpose()
plt.barh(range(20), s1[0])
plt.barh(range(20), s1[1], left=s1[0], color='g')
plt.barh(range(20), s1[2], left=s1[0]+s1[1], color='r')
plt.title("random length ")
plt.show()
✓ 0.2s
```



oiddqtjn5

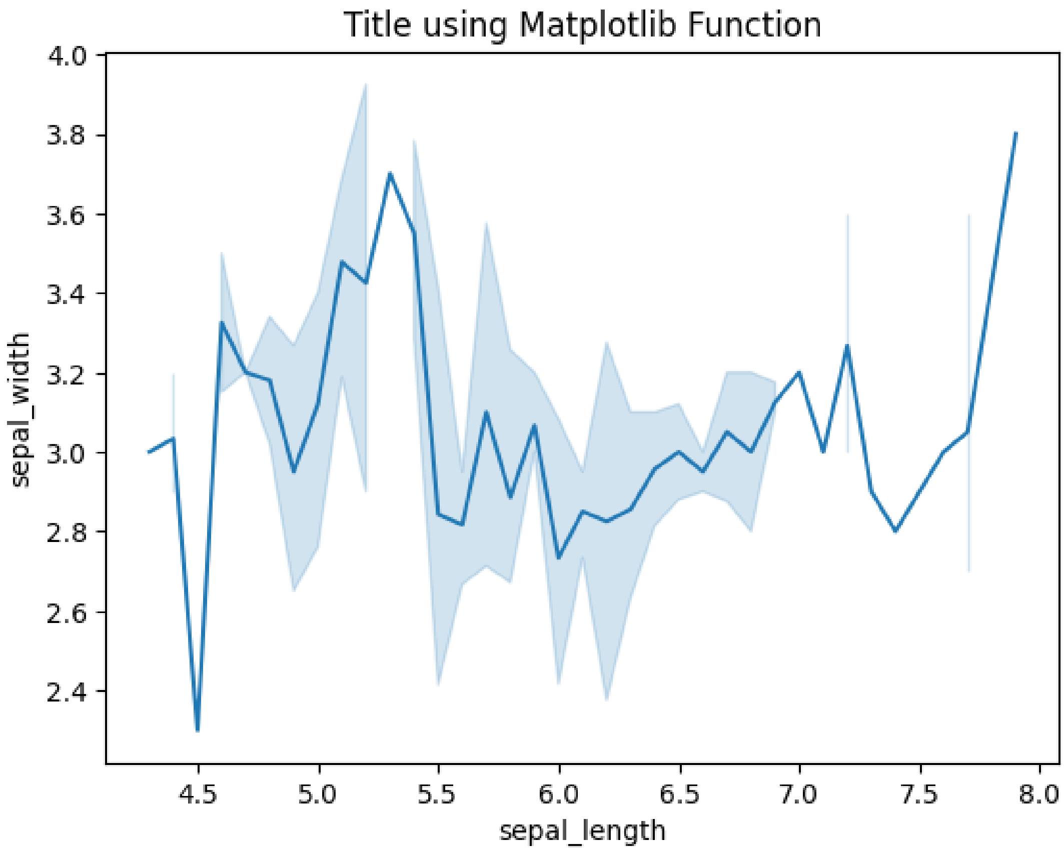
February 17, 2023

1 Arya Chakraborty

2 22MSD7020 VIT-AP

[1]:

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt

# loading dataset
data = sns.load_dataset("iris")

# draw lineplot
sns.lineplot(x="sepal_length", y="sepal_width", data=data)
plt.title('Title using Matplotlib Function')
plt.show()
```



[2]:
```
data.head(20)
```

[2]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa
10	5.4	3.7	1.5	0.2	setosa
11	4.8	3.4	1.6	0.2	setosa
12	4.8	3.0	1.4	0.1	setosa
13	4.3	3.0	1.1	0.1	setosa
14	5.8	4.0	1.2	0.2	setosa
15	5.7	4.4	1.5	0.4	setosa
16	5.4	3.9	1.3	0.4	setosa

17	5.1	3.5	1.4	0.3	setosa
18	5.7	3.8	1.7	0.3	setosa
19	5.1	3.8	1.5	0.3	setosa

[3]:

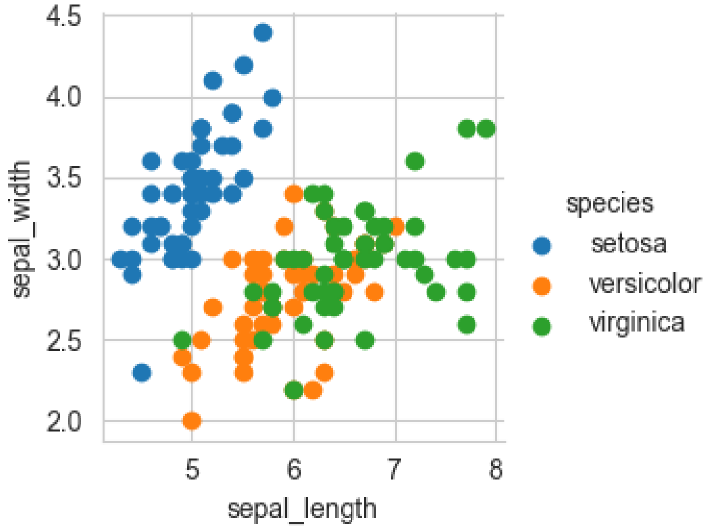
```
data.tail(20)
```

[3]:

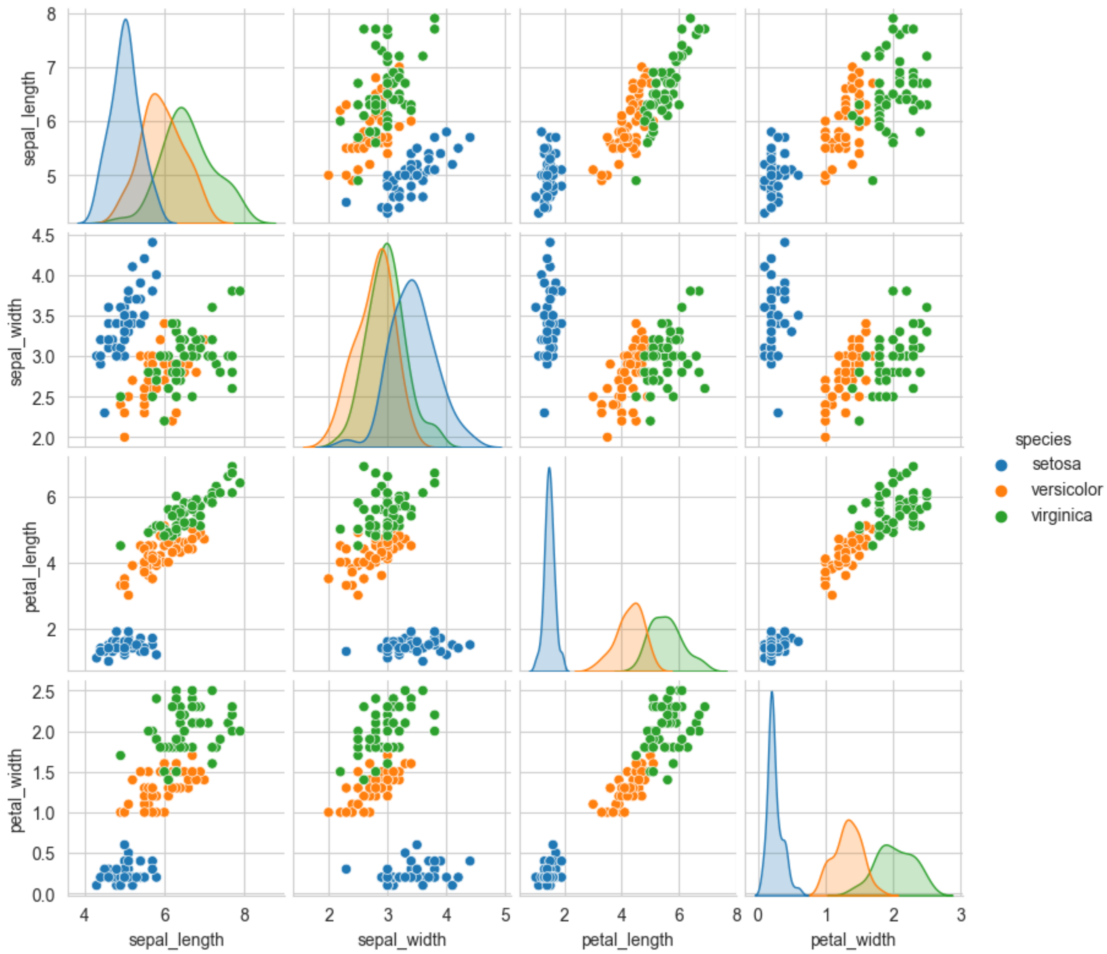
	sepal_length	sepal_width	petal_length	petal_width	species
130	7.4	2.8	6.1	1.9	virginica
131	7.9	3.8	6.4	2.0	virginica
132	6.4	2.8	5.6	2.2	virginica
133	6.3	2.8	5.1	1.5	virginica
134	6.1	2.6	5.6	1.4	virginica
135	7.7	3.0	6.1	2.3	virginica
136	6.3	3.4	5.6	2.4	virginica
137	6.4	3.1	5.5	1.8	virginica
138	6.0	3.0	4.8	1.8	virginica
139	6.9	3.1	5.4	2.1	virginica
140	6.7	3.1	5.6	2.4	virginica
141	6.9	3.1	5.1	2.3	virginica
142	5.8	2.7	5.1	1.9	virginica
143	6.8	3.2	5.9	2.3	virginica
144	6.7	3.3	5.7	2.5	virginica
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

[4]:

```
sns.set_style("whitegrid");
sns.FacetGrid(data,hue="species") \
   .map(plt.scatter, "sepal_length","sepal_width") \
   .add_legend();
plt.show()
```



[5]:
```
sns.set_style("whitegrid");
sns.pairplot(data,hue="species",height=2);
plt.show()
```



1). petal_length and petal_width are the most useful features to identify various flower types.

2). While Setosa can be easily identified (linearly seperable), Virnica and Versicolor have some overlap (almost linearly seperable).

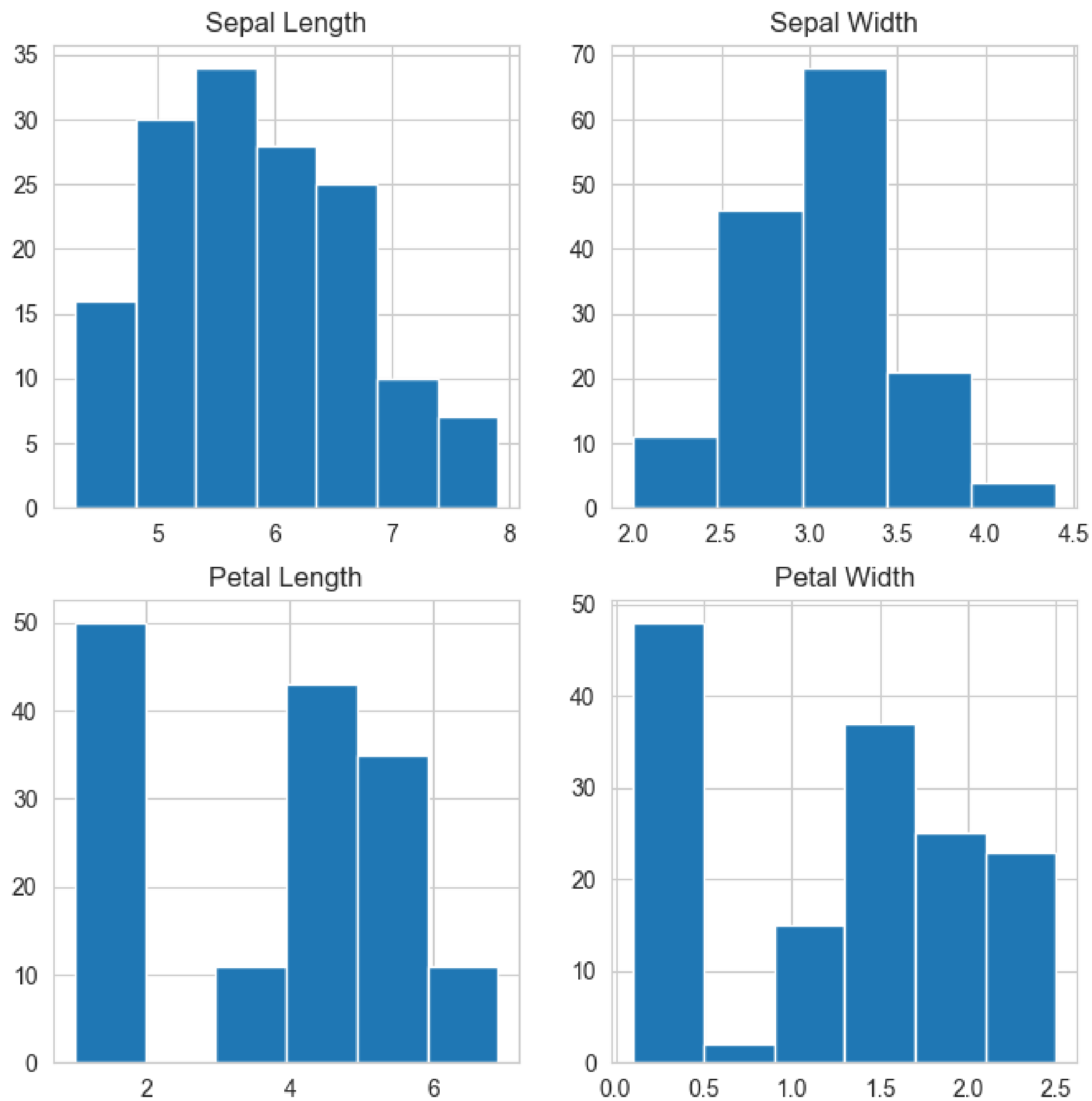
3). We can find "lines" and "if-else" conditions to build a simple model to classify the flower types.

3 HISTOGRAM & PDF

4 A histogram is a bar graph of raw data that creates a picture of the data distribution. The bars represent the frequency of occurrence by classes of data. A histogram shows basic information about the data set, such as central location , width of spread , and shape.

[6]:
```
import numpy as np
iris_setosa = data.loc[data["species"]=="setosa"];
iris_virginica = data.loc[data["species"]=="virginica"];
iris_versicolor = data.loc[data["species"]=="versicolor"]
```

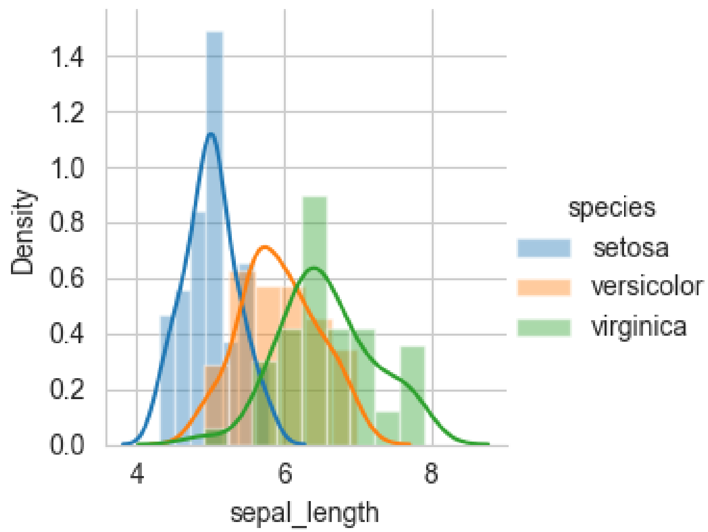
[7]:
```
fig, axes = plt.subplots(2, 2, figsize=(8,8))

axes[0,0].set_title("Sepal Length")
axes[0,0].hist(data['sepal_length'], bins=7)

axes[0,1].set_title("Sepal Width")
axes[0,1].hist(data['sepal_width'], bins=5);

axes[1,0].set_title("Petal Length")
axes[1,0].hist(data['petal_length'], bins=6);

axes[1,1].set_title("Petal Width")
axes[1,1].hist(data['petal_width'], bins=6);
```

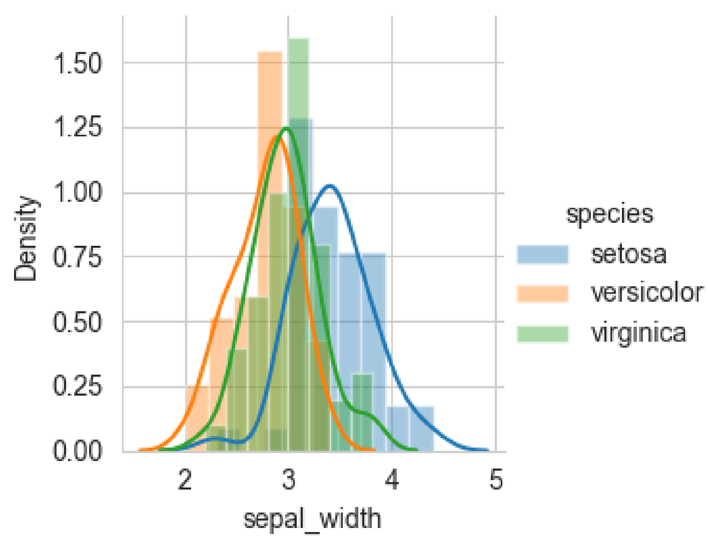


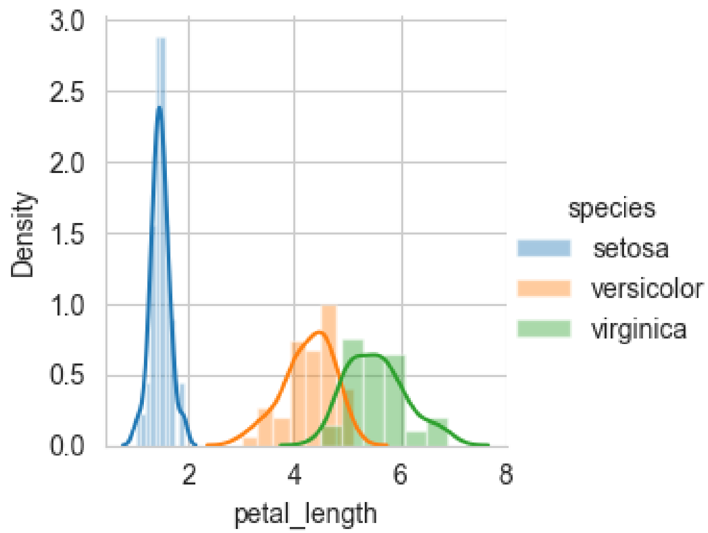
The highest frequency of the sepal length is between 30 and 35 which is between 5.5 and 6 The highest frequency of the sepal Width is around 70 which is between 3.0 and 3.5 The highest frequency of the petal length is around 50 which is between 1 and 2 The highest frequency of the petal width is between 40 and 50 which is between 0.0 and 0.5

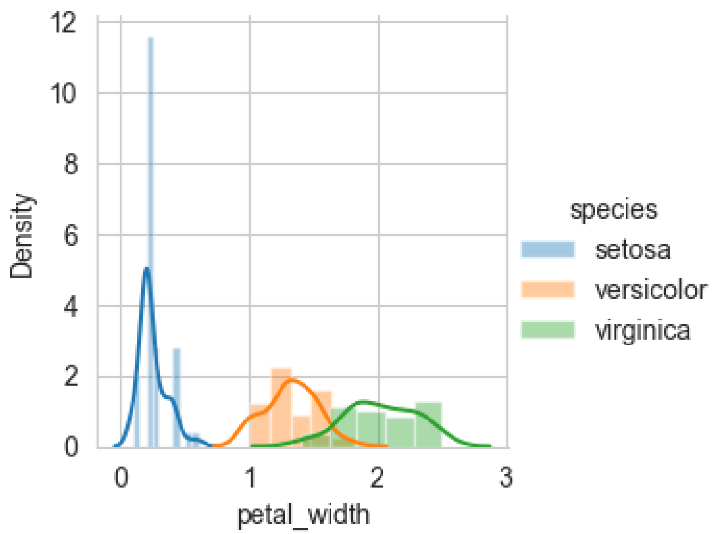
[8]:
```
import warnings
warnings.filterwarnings("ignore")
plot = sns.FacetGrid(data, hue="species")
plot.map(sns.distplot, "sepal_length").add_legend()

plot = sns.FacetGrid(data, hue="species")
plot.map(sns.distplot, "sepal_width").add_legend()
```

```
plot = sns.FacetGrid(data, hue="species")
plot.map(sns.distplot, "petal_length").add_legend()

plot = sns.FacetGrid(data, hue="species")
plot.map(sns.distplot, "petal_width").add_legend()

plt.show()
```









In the case of Sepal Length, there is a huge amount of overlapping. In the case of Sepal Width also, there is a huge amount of overlapping. In the case of Petal Length, there is a very little amount of overlapping. In the case of Petal Width also, there is a very little amount of overlapping. So we can use Petal Length and Petal Width as the classification feature.

[9]:
```
data.corr(method='pearson')
```

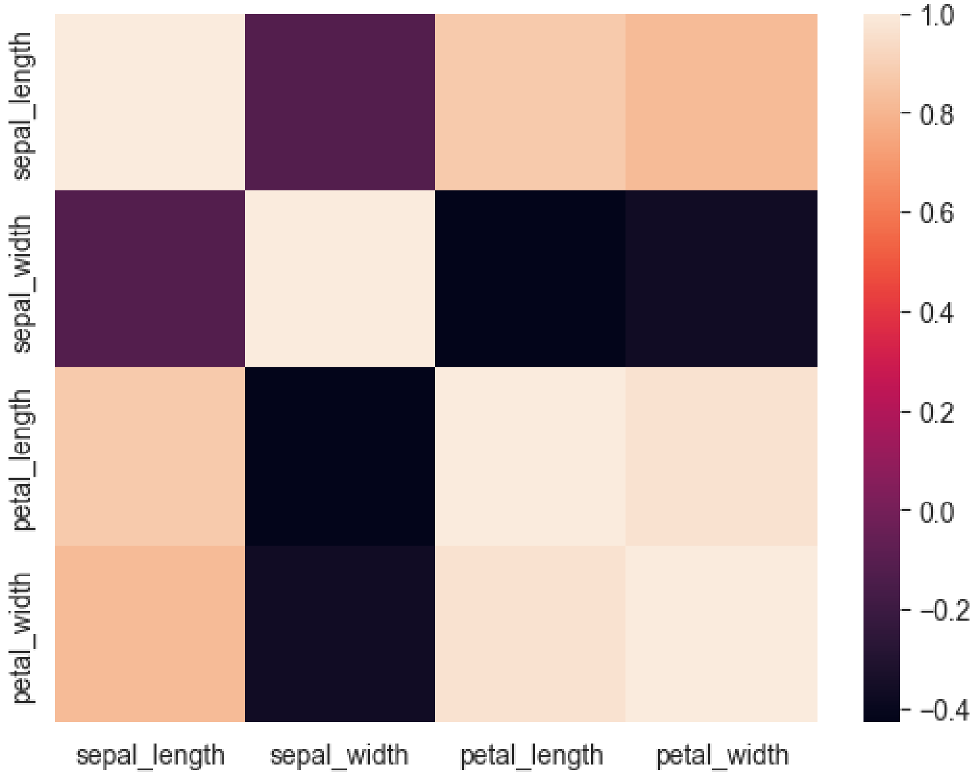
[9]:

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.000000	-0.117570	0.871754	0.817941
sepal_width	-0.117570	1.000000	-0.428440	-0.366126

```
petal_length      0.871754     -0.428440      1.000000      0.962865
petal_width       0.817941     -0.366126      0.962865      1.000000
```

5 HEATMAPS

[10]:
```python
sns.heatmap(data.corr(method='pearson'));

plt.show()
```

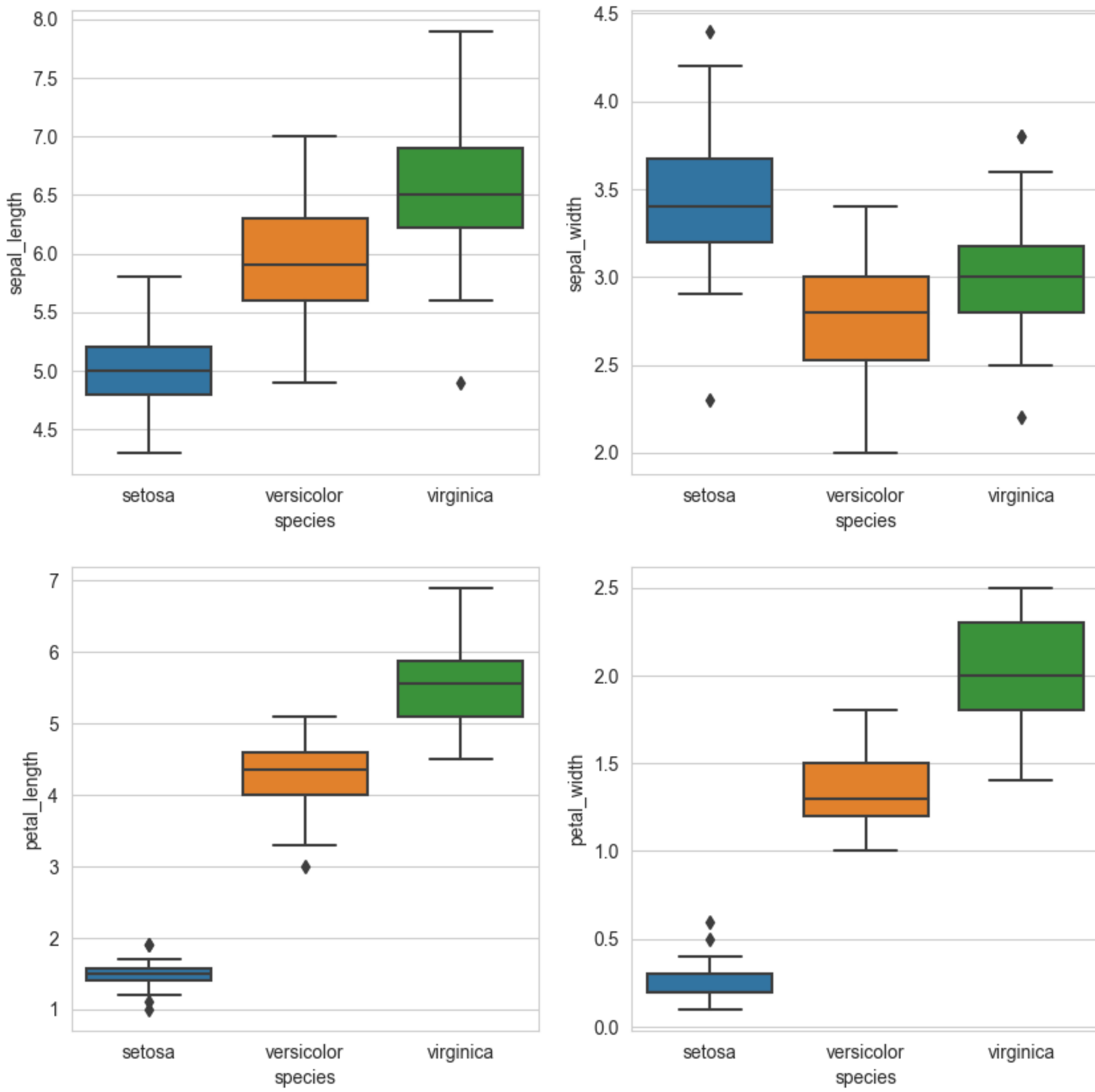


Petal width and petal length have high correlations. Petal length and sepal width have good correlations. Petal Width and Sepal length have good correlations.

6 Box-plot can be visualized as a PDF on the side-ways.

[11]:
```python
def graph(y):
    sns.boxplot(x="species", y=y, data=data)

plt.figure(figsize=(10,10))

plt.subplot(221)
```

```
graph('sepal_length')

plt.subplot(222)
graph('sepal_width')

plt.subplot(223)
graph('petal_length')

plt.subplot(224)
graph('petal_width')

plt.show()
```



Species Setosa has the smallest features and less distributed with some outliers. Species Versicolor has the average features. Species Virginica has the highest features

7 Handling Outliers

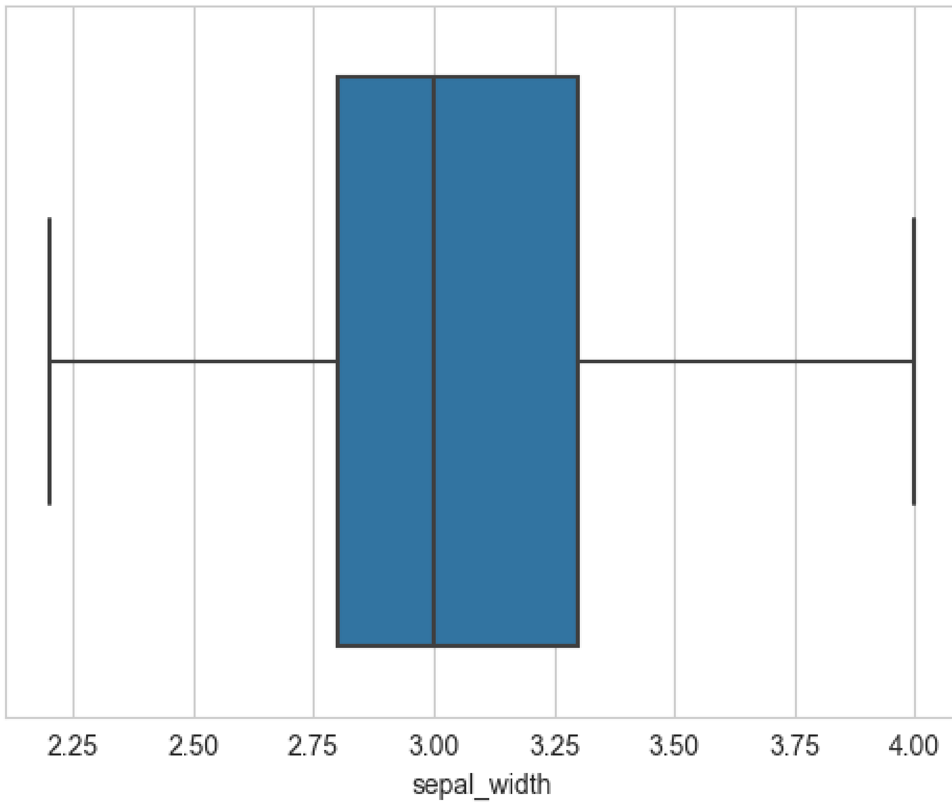
An Outlier is a data-item/object that deviates significantly from the rest of the (so-called normal)objects. They can be caused by measurement or execution errors. The analysis for outlier detection is referred to as outlier mining. There are many ways to detect the outliers, and the removal process is the data frame same as removing a data item from the panda's dataframe.

For removing the outlier, one must follow the same process of removing an entry from the dataset using its exact position in the dataset because in all the above methods of detecting the outliers end result is the list of all those data items that satisfy the outlier definition according to the method used.

[12]:
```
Q1 = np.percentile(data['sepal_width'], 25,
                interpolation = 'midpoint')

Q3 = np.percentile(data['sepal_width'], 75,
                interpolation = 'midpoint')
IQR = Q3 - Q1

print("Old Shape: ", data.shape)

# Upper bound
upper = np.where(data['sepal_width'] >= (Q3+1.5*IQR))

# Lower bound
lower = np.where(data['sepal_width'] <= (Q1-1.5*IQR))

# Removing the Outliers
data.drop(upper[0], inplace = True)
data.drop(lower[0], inplace = True)

print("New Shape: ", data.shape)

sns.boxplot(x='sepal_width', data=data)
```

```
Old Shape:  (150, 5)
New Shape:  (146, 5)
```

[12]:
```
<AxesSubplot: xlabel='sepal_width'>
```



8 Violen plot

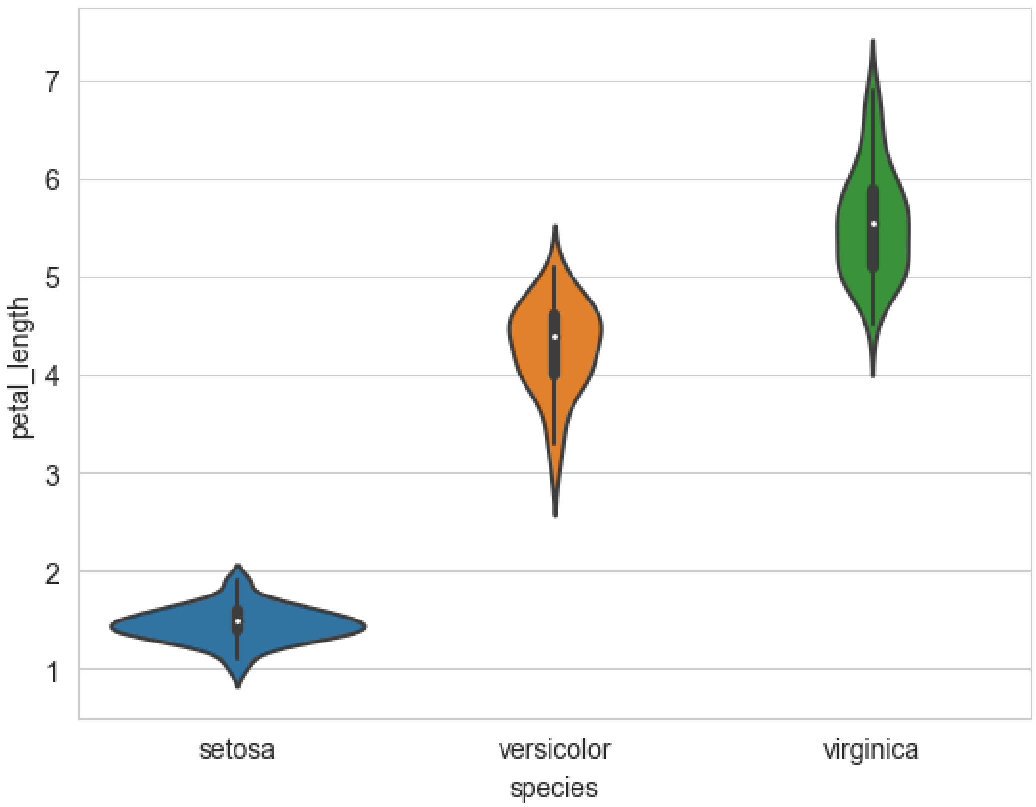
9 A violin plot combines the benefits of the previous two plots

10 and simplifies them

11 Denser regions of the data are fatter, and sparser ones thinner

12 in a violin plot

[13]:
```
sns.violinplot(x="species", y="petal_length", data=data, height=8)
plt.show()
```



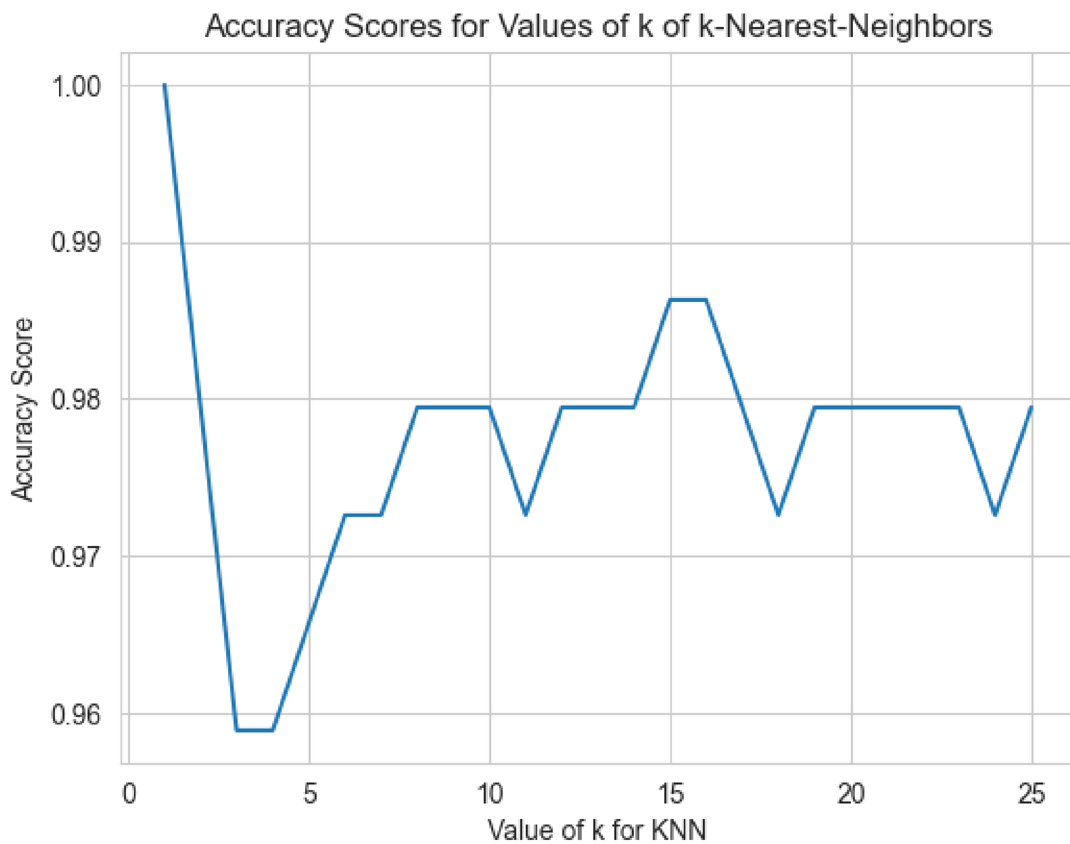
[14]:
```
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

[15]:
```
X = data.filter(['sepal_length','petal_length','sepal_width','petal_width'],␣
 ↪axis=1)
y = data['species']

print(X.shape)
print(y.shape)
```

```
(146, 4)
(146,)
```

[16]:
```
k_range = list(range(1,26))
scores = []
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X, y)
```

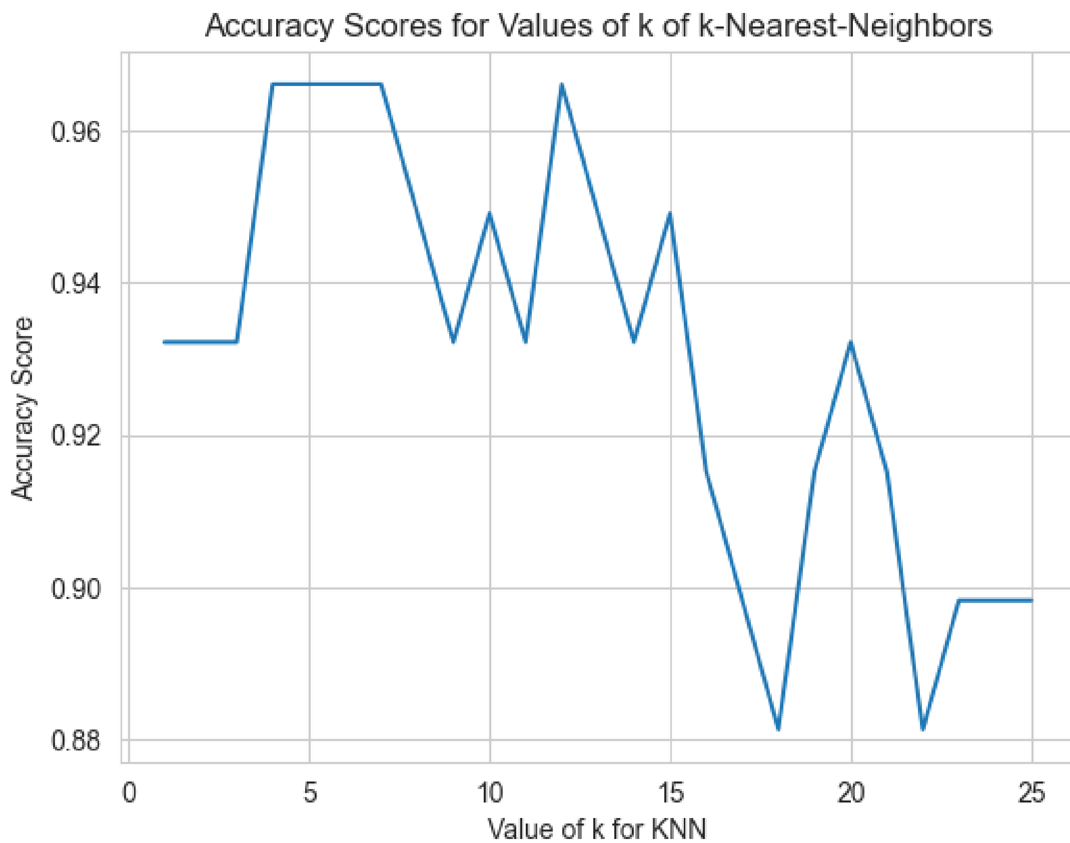
```
    y_pred = knn.predict(X)
    scores.append(metrics.accuracy_score(y, y_pred))

plt.plot(k_range, scores)
plt.xlabel('Value of k for KNN')
plt.ylabel('Accuracy Score')
plt.title('Accuracy Scores for Values of k of k-Nearest-Neighbors')
plt.show()
```



[22]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,␣
 ↪random_state=5)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(87, 4)
(87,)
(59, 4)
(59,)
```

[23]:
```
k_range = list(range(1,26))
scores = []
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    scores.append(metrics.accuracy_score(y_test, y_pred))

plt.plot(k_range, scores)
plt.xlabel('Value of k for KNN')
plt.ylabel('Accuracy Score')
plt.title('Accuracy Scores for Values of k of k-Nearest-Neighbors')
plt.show()
```



[25]:
```
knn = KNeighborsClassifier(n_neighbors=12)
knn.fit(X, y)

# make a prediction for an example of an out-of-sample observation
knn.predict([[6, 3, 4, 2]])
```

[25]: array(['versicolor'], dtype=object)

[33]:
```
Species=data["species"].unique().tolist()
l=[x for x in range (len(Species))]

#replacing species names with numbers
data["species"].replace(Species,l,inplace=True)
data["species"].unique()

lr=LogisticRegression()

Z=data[['sepal_length','sepal_width','petal_length','petal_width']]
lr.fit(Z,data['species'])
Yhat=lr.predict(Z)
print("Accuracy Score:- ",metrics.accuracy_score(data["species"],Yhat))

print(lr.predict([[5.0,3.6,1.4,0.2]]))
```

```
Accuracy Score:-  0.9726027397260274
[0]
```

[34]:
```
lr.predict([[5.0,3.6,1.4,0.2]])
```

[34]: array([0], dtype=int64)