In [6]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [7]:

```
iris=pd.read_csv("Downloads/Iris.csv")
```

In [8]:

```
iris.head()
```

Out[8]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [9]:

```
iris.shape
```

Out[9]:

```
(150, 6)
```

In [10]:

```
iris.isnull().sum()
```

Out[10]:

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [11]:

```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [12]:

```
iris["Species"].unique()
```

Out[12]:

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

In [13]:

```
iris.drop('Id',axis=1,inplace=True)
```

In [14]:

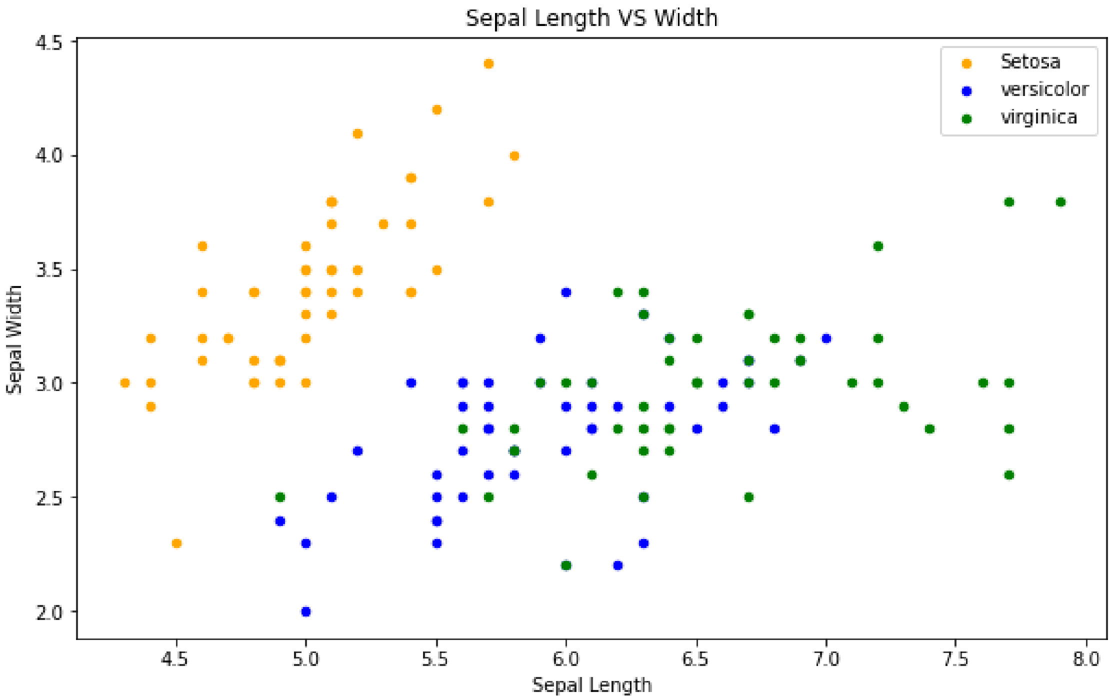
```
iris.head()
```

Out[14]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
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

In [15]:

```
fig = iris[iris.Species=='Iris-setosa'].plot(kind='scatter',x='SepalLengthCm',y='SepalWidthCm',color='orange', label='Setosa')
iris[iris.Species=='Iris-versicolor'].plot(kind='scatter',x='SepalLengthCm',y='SepalWidthCm',color='blue', label='versicolor',ax=fig)
iris[iris.Species=='Iris-virginica'].plot(kind='scatter',x='SepalLengthCm',y='SepalWidthCm',color='green', label='virginica', ax=fig)
fig.set_xlabel("Sepal Length")
fig.set_ylabel("Sepal Width")
fig.set_title("Sepal Length VS Width")
fig=plt.gcf()
fig.set_size_inches(10,6)
plt.show()
fig.savefig("Sepal Length VS Width.png")
```



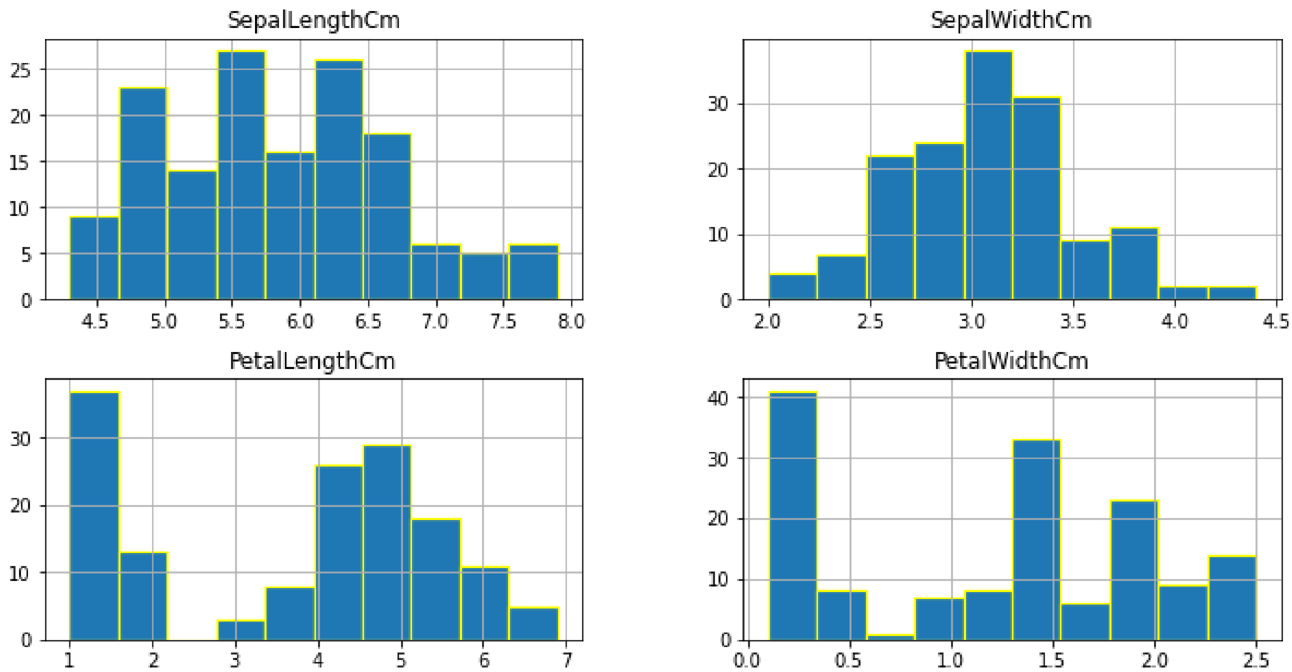
In [16]:

```
fig = iris[iris.Species=='Iris-setosa'].plot.scatter(x='PetalLengthCm',y='PetalWidthCm',
color='orange', label='Setosa')
```

```
iris[iris.Species=='Iris-versicolor'].plot.scatter(x='PetalLengthCm',y='PetalWidthCm',co
lor='blue', label='versicolor',ax=fig)
iris[iris.Species=='Iris-virginica'].plot.scatter(x='PetalLengthCm',y='PetalWidthCm',col
or='green', label='virginica', ax=fig)
fig.set_xlabel("Petal Length")
fig.set_ylabel("Petal Width")
fig.set_title(" Petal Length VS Width")
fig=plt.gcf()
fig.set_size_inches(10,6)
plt.show()
fig.savefig("Petal Length VS Width.png")
```

In [17]:

```
iris.hist(edgecolor='Yellow', linewidth=1.2)
fig=plt.gcf()
fig.set_size_inches(12,6)
plt.show()
```



In [44]:

```
iris.describe().T
```

Out[44]:

	count	mean	std	min	25%	50%	75%	max
SepalLengthCm	150.0	5.843333	0.828066	4.3	5.1	5.80	6.4	7.9
SepalWidthCm	150.0	3.054000	0.433594	2.0	2.8	3.00	3.3	4.4
PetalLengthCm	150.0	3.758667	1.764420	1.0	1.6	4.35	5.1	6.9
PetalWidthCm	150.0	1.198667	0.763161	0.1	0.3	1.30	1.8	2.5

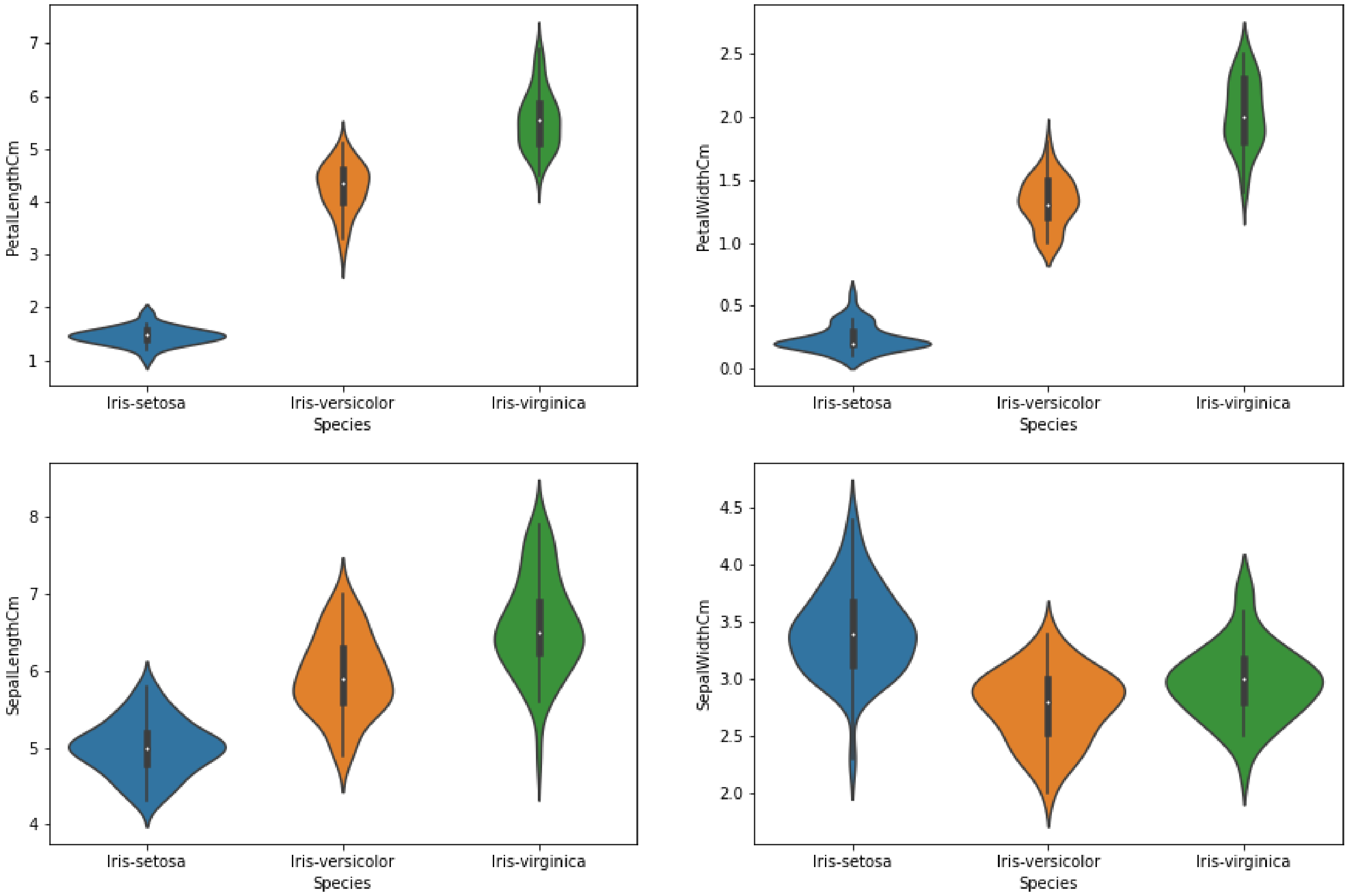
In [ ]:

```

```

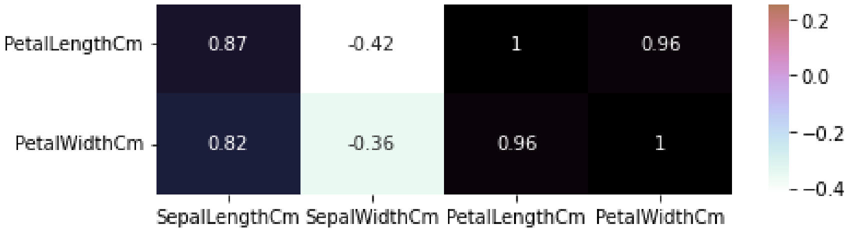
In [18]:

```
plt.figure(figsize=(15,10))
plt.subplot(2,2,1)
sns.violinplot(x='Species',y='PetalLengthCm',data=iris)
plt.subplot(2,2,2)
sns.violinplot(x='Species',y='PetalWidthCm',data=iris)
plt.subplot(2,2,3)
sns.violinplot(x='Species',y='SepalLengthCm',data=iris)
plt.subplot(2,2,4)
sns.violinplot(x='Species',y='SepalWidthCm',data=iris)
fig.savefig("variable with species.png")
```



In [19]:

```
plt.figure(figsize=(7,4))
sns.heatmap(iris.corr(),annot=True,cmap='cubehelix_r')
plt.show()
```



In [56]:

```
x=iris.iloc[:, :-1].values
```

In [49]:

```
y=iris.iloc[:, -1].values
```

In [53]:

```
iris.dtypes
```

Out[53]:

```
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

In [54]:

```
iris['Species']=iris['Species'].astype('category')
iris.dtypes
```

Out[54]:

```
SepalLengthCm     float64
SepalWidthCm      float64
PetalLengthCm     float64
PetalWidthCm      float64
Species          category
dtype: object
```

In [57]:

```
#random forest classifier method:
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, random_state =
123)
```

In [58]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [59]:

```
from sklearn.ensemble import RandomForestClassifier
Rf_classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_
state = 0)
Rf_classifier.fit(x_train, y_train)
```

Out[59]:

```
RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)
```

In [60]:

```
y_pred = Rf_classifier.predict(x_test)
```

In [83]:

```
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score

print("Accuracy Score:\t",accuracy_score(y_test,y_pred))
print('-'*80)
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))
print('-'*80)
print("Classification Report:\t",classification_report(y_test,y_pred))
print('-'*80)
```

```
Accuracy Score:  0.9
--------------------------------------------------------------------------------
Confusion Matrix:
 [[13  0  0]
 [ 0  5  1]
 [ 0  2  9]]
--------------------------------------------------------------------------------
Classification Report:                      precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        13
Iris-versicolor       0.71      0.83      0.77         6
 Iris-virginica       0.90      0.82      0.86        11

       accuracy                           0.90        30
      macro avg       0.87      0.88      0.88        30
   weighted avg       0.91      0.90      0.90        30

--------------------------------------------------------------------------------
```

In [62]:

```
print('The accuracy of the Random forest is:',metrics.accuracy_score(y_pred,y_test))
```

```
The accuracy of the Random forest is: 0.9
```

In [71]:

```
#Logistic Regression
xl = iris.iloc[:, [0,1,2, 3]].values
yl = iris.iloc[:, 4].values
from sklearn.model_selection import train_test_split
x_ltrain, x_ltest, y_ltrain, y_ltest = train_test_split(xl, yl, test_size = 0.25, random
_state = 0)
```

In [72]:

```
print("Shape of training samples:\t",x_ltrain.shape)
print("Shape of testing samples:\t",x_ltest.shape)
```

```
Shape of training samples:  (112, 4)
Shape of testing samples:  (38, 4)
```

In [73]:

```
iris.dtypes
```

Out[73]:

```
SepalLengthCm     float64
SepalWidthCm      float64
PetalLengthCm     float64
PetalWidthCm      float64
Species          category
dtype: object
```

In [74]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_ltrain = sc.fit_transform(x_ltrain)
```

```
x_ltest = sc.transform(x_ltest)
```

In [75]:

```
from sklearn.linear_model import LogisticRegression
l_classifier = LogisticRegression(random_state = 0, solver='lbfgs', multi_class='auto')
l_classifier.fit(x_ltrain, y_ltrain)
```

Out[75]:

```
LogisticRegression(random_state=0)
```

In [76]:

```
y_lpred = l_classifier.predict(x_ltest)
```

In [77]:

```
probs_y=l_classifier.predict_proba(x_ltest)
probs_y = np.round(probs_y, 2)
res = "{:<10} | {:<10} | {:<10} | {:<13} | {:<5}".format("y_test", "y_pred", "Setosa(%)"
, "versicolor(%)", "virginica(%)\n")
res += "-"*65+"\n"
res += "\n".join("{:<10} | {:<10} | {:<10} | {:<13} | {:<10}".format(x, y, a, b, c) for
x, y, a, b, c in zip(y_test, y_pred, probs_y[:,0], probs_y[:,1], probs_y[:,2]))
res += "\n"+"-"*65+"\n"
print(res)
```

```
y_test     | y_pred     | Setosa(%)  | versicolor(%) | virginica(%)
-----------------------------------------------------------------
Iris-versicolor | Iris-virginica | 0.0        | 0.03          | 0.97
Iris-virginica | Iris-virginica | 0.01       | 0.95          | 0.04
Iris-virginica | Iris-virginica | 1.0        | 0.0           | 0.0
Iris-versicolor | Iris-versicolor | 0.0        | 0.08          | 0.92
Iris-setosa | Iris-setosa | 0.98       | 0.02          | 0.0
Iris-virginica | Iris-versicolor | 0.0        | 0.01          | 0.99
Iris-versicolor | Iris-versicolor | 0.98       | 0.02          | 0.0
Iris-setosa | Iris-setosa | 0.01       | 0.71          | 0.28
Iris-setosa | Iris-setosa | 0.0        | 0.73          | 0.27
Iris-versicolor | Iris-versicolor | 0.02       | 0.89          | 0.08
Iris-virginica | Iris-virginica | 0.0        | 0.44          | 0.56
Iris-setosa | Iris-setosa | 0.02       | 0.76          | 0.22
Iris-versicolor | Iris-versicolor | 0.01       | 0.85          | 0.13
Iris-virginica | Iris-virginica | 0.0        | 0.69          | 0.3
Iris-virginica | Iris-virginica | 0.01       | 0.75          | 0.24
Iris-virginica | Iris-virginica | 0.95       | 0.05          | 0.0
Iris-setosa | Iris-setosa | 0.02       | 0.72          | 0.26
Iris-setosa | Iris-setosa | 0.03       | 0.86          | 0.11
Iris-versicolor | Iris-versicolor | 0.94       | 0.06          | 0.0
Iris-setosa | Iris-setosa | 0.99       | 0.01          | 0.0
Iris-setosa | Iris-setosa | 0.0        | 0.17          | 0.83
Iris-virginica | Iris-versicolor | 0.04       | 0.71          | 0.25
Iris-setosa | Iris-setosa | 0.98       | 0.02          | 0.0
Iris-virginica | Iris-virginica | 0.96       | 0.04          | 0.0
Iris-setosa | Iris-setosa | 0.0        | 0.35          | 0.65
Iris-setosa | Iris-setosa | 1.0        | 0.0           | 0.0
Iris-setosa | Iris-setosa | 0.99       | 0.01          | 0.0
Iris-virginica | Iris-virginica | 0.02       | 0.87          | 0.11
Iris-virginica | Iris-virginica | 0.09       | 0.9           | 0.02
Iris-setosa | Iris-setosa | 0.97       | 0.03          | 0.0
-----------------------------------------------------------------
```

In [82]:

```
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score

print("Accuracy Score:\t",accuracy_score(y_ltest,y_lpred))
print('-'*80)
print("Confusion Matrix:\n",confusion_matrix(y_ltest,y_lpred))
print('-'*80)
```

```
print("Classification Report:\t",classification_report(y_ltest,y_lpred))
print('-'*80)
```

```
Accuracy Score:  0.9736842105263158
--------------------------------------------------------------------------------
Confusion Matrix:
 [[13  0  0]
 [ 0 15  1]
 [ 0  0  9]]
--------------------------------------------------------------------------------
Classification Report:                    precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        13
Iris-versicolor       1.00      0.94      0.97        16
 Iris-virginica       0.90      1.00      0.95         9

       accuracy                           0.97        38
      macro avg       0.97      0.98      0.97        38
   weighted avg       0.98      0.97      0.97        38

--------------------------------------------------------------------------------
```

In [80]:

```
print("Accuracy: ", l_classifier.score(x_ltest, y_ltest) * 100)
```

```
Accuracy:  97.36842105263158
```

Comparing both models we can say that accuracy of logistic regression model is better than random forest model.