IRIS FLOWER CLASSIFICATION

NAME : - SIDDHI PRASAD GAMBHIR

In [1]:
```
import pandas as pd
from sklearn.datasets import load_iris
iris = load_iris()
import warnings
warnings.filterwarnings("ignore")
import matplotlib.pylab as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report,accuracy_score
```

In [2]:
```
iris=pd.read_csv("C:\Users\Siddhi\Desktop\IRIS.csv")
iris
```

Out[2]:

	sepal_length	sepal_width	petal_length	petal_width	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
...	...	...	...	...	...
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

In [3]:
```
iris.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   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [4]:
```
iris.columns
```

Out[4]:
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

In [5]:
```
iris.head()
```

Out[5]:

	sepal_length	sepal_width	petal_length	petal_width	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 [6]:
```
iris.describe
```

Out[6]:
```
<bound method NDFrame.describe of       sepal_length  sepal_width  petal_length  petal_width         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
..             ...          ...           ...          ...             ...
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 x 5 columns]>
```

In [7]:
```
iris.shape
```

Out[7]:
```
(150, 5)
```

In [8]:
```
display(iris.head())
display(iris.shape)
```

	sepal_length	sepal_width	petal_length	petal_width	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

(150, 5)

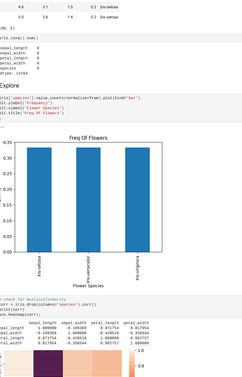
In [9]:
```
iris.isnull().sum()
```

Out[9]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

Explore

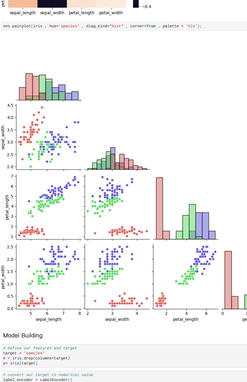
In [10]:
```
iris['species'].value_counts(normalize=True).plot(kind='bar')
plt.xlabel('Species')
plt.ylabel('Flower Species')
plt.title('Freq Of Flowers')
```

Out[10]:
```
''
```

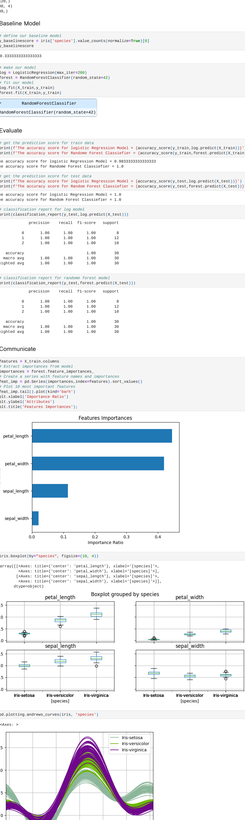


In [11]:
```
# check for multicollinearity
corr = iris.drop(columns="species").corr()
print(corr)
sns.heatmap(corr);
```

```
              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
```



In [12]:
```
sns.pairplot(iris , hue='species' , diag_kind="kde" , corner=True , palette = 'hls');
```



Model Building

In [13]:
```
# define our features and target
x = iris.drop(columns='species')
x = iris.drop(columns=target)
y= iris[target]
```

In [14]:
```
# convert our target to numerical value
le = LabelEncoder()
y = label_encoder.fit_transform(y)
```

In [15]:
```
# split our data to train and test
X_train, X_test, y_train, y_test = train_test_split(x,y,test_size=0.2,random_state=49)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(120, 4)
(120,)
(30, 4)
(30,)
```

Baseline Model

In [16]:
```
# define our baseline model
y_baselinescore = iris['species'].value_counts(normalize=True)[0]
y_baselinescore
```

Out[16]:
```
0.3333333333333333
```

In [17]:
```
# make our models
logreg = LogisticRegression(max_iter=2000)
forest = RandomForestClassifier(random_state=42)
# fit our model
logreg.fit(X_train,y_train)
forest.fit(X_train,y_train)
```

Out[17]:
```
RandomForestClassifier(random_state=42)
```

Evaluate

In [18]:
```
# get the prediction score for train data
print(f"The accuracy score for logistic Regression Model = {accuracy_score(y_train,log.predict(X_train))}")
print(f"The accuracy score for Random Forest Classifier = {accuracy_score(y_train,forest.predict(X_train))}")

The accuracy score for logistic Regression Model = 0.9833333333333333
The accuracy score for Random Forest Classifier = 1.0
```

In [19]:
```
# get the prediction score for test data
print(f"The accuracy score for logistic Regression Model = {accuracy_score(y_test,log.predict(X_test))}")
print(f"The accuracy score for Random Forest Classifier = {accuracy_score(y_test,forest.predict(X_test))}")

The accuracy score for logistic Regression Model = 1.0
The accuracy score for Random Forest Classifier = 1.0
```

In [20]:
```
# classification report for log model
print(classification_report(y_test,log.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         6
           1       1.00      1.00      1.00        12
           2       1.00      1.00      1.00        12

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

In [21]:
```
# classification report for random forest model
print(classification_report(y_test,forest.predict(X_test)))
```

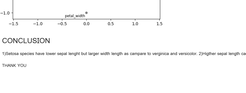
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         6
           1       1.00      1.00      1.00        12
           2       1.00      1.00      1.00        12

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

Communicate

In [22]:
```
features = X_train.columns
# get the importance of model
importances = forest.feature_importances_
# Create a series with feature names and importances
feat_imp = pd.Series(importances, index=features).sort_values()
# Plot 20 most important features
feat_imp.tail().plot(kind='barh')
plt.xlabel('Importance Ratio')
plt.ylabel('Attributes')
plt.title('Features Importances')
```


In [23]:
```
iris.boxplot(by="species", figsize=(10, 4))
```

Out[23]:
```
array([<Axes: title={'center': 'petal_length'}, xlabel='[species]'>,
       <Axes: title={'center': 'petal_width'}, xlabel='[species]'>,
       <Axes: title={'center': 'sepal_length'}, xlabel='[species]'>,
       <Axes: title={'center': 'sepal_width'}, xlabel='[species]'>],
      dtype=object)
```



In [24]:
```
pd.plotting.andrews_curves(iris, 'species')
```

Out[24]:
```
<Axes: >
```


pd.plotting.andrews_curves is useful function for visualizing high-dimensional data by representing each observation as a curve. It allows you to color the curves based on a categorical variable using the class_column paramete

In [25]:
```
pd.plotting.radviz(iris, 'species')
```

Out[25]:
```
<Axes: >
```


CONCLUSION

1)Setosa species have lower sepal lenght but larger width length as campare to verginica and versicolor. 2)Higher sepal length caontain verginica spacies, 3)In order of petal length setosa<versicolor<verginica.

THANK YOU