In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

from dt import DecisionTreeClassifier
import random
```

In [2]:
```
df = pd.read_csv('C:\\Users\\Kemal\\Downloads\\Iris.csv')
df = df.drop("Id", axis = 1)
df_columns = df.columns.tolist()
list = df.values.tolist()
```

Exploratory Data Analysis

In [3]:
```
df["Species"].value_counts()
```

Out[3]:
```
Iris-versicolor    50
Iris-setosa        50
Iris-virginica     50
Name: Species, dtype: int64
```

In [4]:
```
df.info()
```

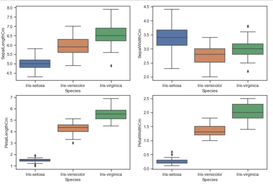
Out[4]:
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
 4   Species        150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [5]:
```
df.describe().T
```

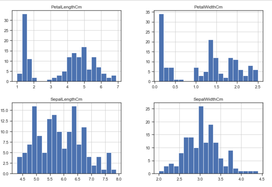
Out[5]:

	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

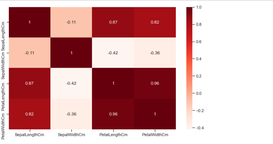
In [6]:
```
sns.set(style='ticks')
plt.figure(figsize=(12,8))
plt.subplot(2,2,1)
sns.boxplot(x='Species', y='SepalLengthCm', data = df)
plt.subplot(2,2,2)
sns.boxplot(x='Species', y='SepalWidthCm', data = df)
plt.subplot(2,2,3)
sns.boxplot(x='Species', y='PetalLengthCm', data = df)
plt.subplot(2,2,4)
sns.boxplot(x='Species', y='PetalWidthCm', data = df)
plt.show()
```



In [7]:
```
df.hist(figsize=(12,8),bins=20)
plt.show()
```

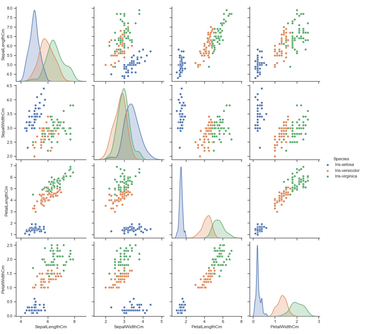


In [8]:
```
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(),cmap=plt.cm.Reds,annot=True)
plt.show()
```



In [9]:
```
sns.pairplot(df, hue="Species", height = 3)
```

Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x171be270df0>
```



In [10]:
```
df.groupby('Species').agg(['mean', 'median'])
```

Out[10]:

	SepalLengthCm		SepalWidthCm		PetalLengthCm		PetalWidthCm	
	mean	median	mean	median	mean	median	mean	median
Species								
Iris-setosa	5.006	5.0	3.418	3.4	1.464	1.50	0.244	0.2
Iris-versicolor	5.936	5.9	2.770	2.8	4.260	4.35	1.326	1.3
Iris-virginica	6.588	6.5	2.974	3.0	5.552	5.55	2.026	2.0

In [11]:
```
df.isna().sum()
```

Out[11]:
```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

Training Of The Classifier

In [17]:
```
def train_test(df, test_size):
    index = df.index.tolist()
    test_idx = random.sample(population=index, k = test_size)
    test_df = df.iloc[test_idx]
    train_df = df.drop(test_idx)
    return train_df, test_df
```

In [18]:
```
train_df, test_df = train_test(df, test_size = 20)
train_list = train_df.values.tolist()
train_list_with_columns = train_df.values.tolist()
train_list_with_columns.insert(0, df_columns)
test_list = test_df.values.tolist()
```

In [19]:
```
dtc = DecisionTreeClassifier(4)
```

In [20]:
```
tree = dtc.fit(train_list_with_columns,[])
```

Interpretation Of The Results

In [21]:
```
species, classification = dtc.predict(test_list)
```

```
Confusion Matrix :  [[5, 0, 0], [0, 5, 1], [0, 1, 8]]
Iris-setosa precision :  1.0
Iris-virginica precision :  0.8333333333333334
Iris-versicolor precision :  0.8888888888888888
Iris-setosa recall :  1.0
Iris-versicolor recall :  0.8333333333333334
Iris-virginica recall :  0.8888888888888888
Iris-setosa F1-Score :  1.0
Iris-virginica F1-Score :  0.8333333333333334
Iris-versicolor F1-Score :  0.8888888888888888
Accuracy :  90.0
```

In [23]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve, auc
from sklearn import datasets
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import LinearSVC
from sklearn.preprocessing import label_binarize
import matplotlib.pyplot as plt

n_classes = 3
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range (len(species)):
    if species[i] == "Iris-setosa":
        species[i] = 0
    if species[i] == "Iris-virginica":
        species[i] = 1
    if species[i] == "Iris-versicolor":
        species[i] = 2

for i in range (len(classification)):
    if classification[i] == "Iris-setosa":
        classification[i] = 0
    if classification[i] == "Iris-virginica":
        classification[i] = 1
    if classification[i] == "Iris-versicolor":
        classification[i] = 2
print(species)
print(classification)
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(species, classification, pos_label=species[0])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Plot of a ROC curve for a specific class
for i in range(n_classes):
    plt.figure()
    plt.plot(fpr[i], tpr[i], label='ROC curve (area = %0.2f)' % roc_auc[i])
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC CURVE')
    plt.legend(loc="lower right")
    plt.show()
```

```
[2, 2, 2, 2, 1, 1, 2, 1, 2, 0, 2, 2, 0, 1, 1, 0, 1, 0, 2, 0]
[2, 2, 2, 2, 1, 1, 2, 2, 0, 2, 2, 0, 1, 1, 0, 1, 0, 2, 0]
```