NAIVE BAYERS CLASSIFICATION FOR DWM EXPERIEMNT 2

In [4]:
```
import numpy as np
import pandas as pd #FOR DATA PROCESSING
import matplotlib.pyplot as plt # FOR DISPLAYING CHARTS
import seaborn as sns
```

In [5]:
```
iris = pd.read_csv('iris.data')
```

In [6]:
```
iris.head(10)
```

Out[6]:

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa
5	4.6	3.4	1.4	0.3	Iris-setosa
6	5.0	3.4	1.5	0.2	Iris-setosa
7	4.4	2.9	1.4	0.2	Iris-setosa
8	4.9	3.1	1.5	0.1	Iris-setosa
9	5.4	3.7	1.5	0.2	Iris-setosa

In [7]:
```
headerList = ["SepalLengthCm", "SepalWidthCm", "PetalLengthCm", "PetalWidthCm", "Spe
```

In [8]:
```
iris.to_csv("header_iris.csv", header=headerList, index=False)
```

In [9]:
```
iris = pd.read_csv("header_iris.csv")
```

In [10]:
```
iris.head()
```

Out[10]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa

In [11]:
```
len(iris['Species'].unique())
```

Out[11]: 3

In [12]:
```
iris['Species'].unique()
```

Out[12]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [13]:
```
iris.describe(include="all")
```

Out[13]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
count	149.000000	149.000000	149.000000	149.000000	149
unique	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	Iris-virginica
freq	NaN	NaN	NaN	NaN	50
mean	5.848322	3.051007	3.774497	1.205369	NaN
std	0.828594	0.433499	1.759651	0.761292	NaN
min	4.300000	2.000000	1.000000	0.100000	NaN
25%	5.100000	2.800000	1.600000	0.300000	NaN
50%	5.800000	3.000000	4.400000	1.300000	NaN
75%	6.400000	3.300000	5.100000	1.800000	NaN
max	7.900000	4.400000	6.900000	2.500000	NaN

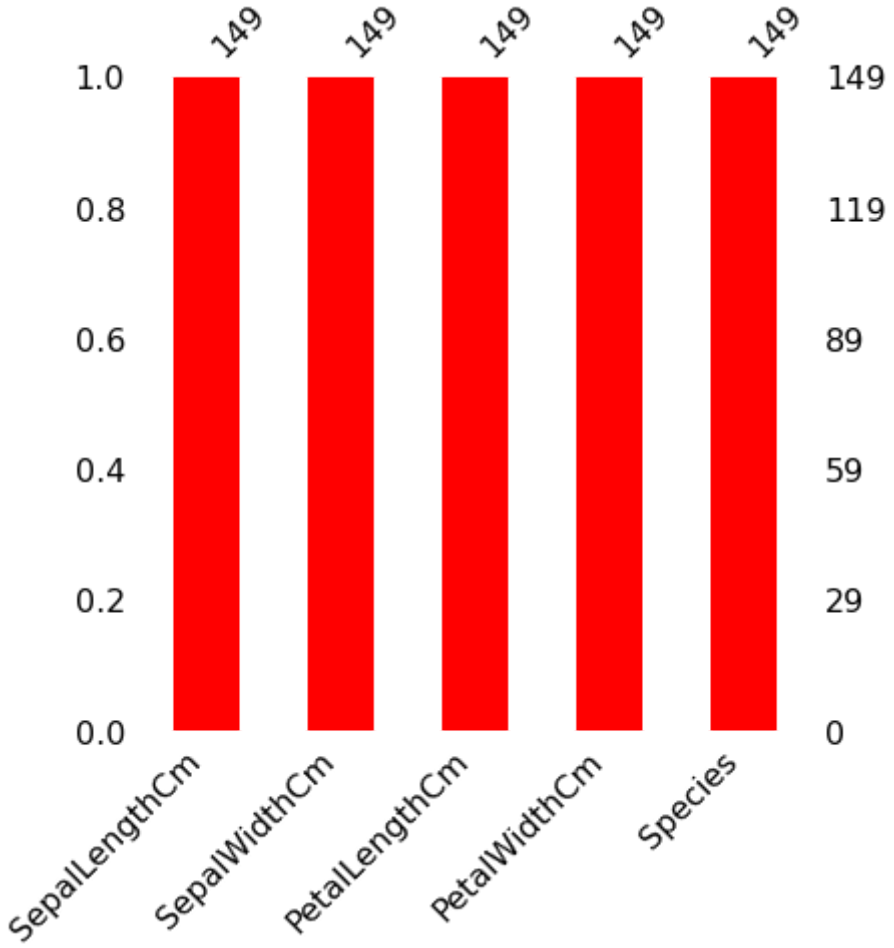
In [14]:
```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149 entries, 0 to 148
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   SepalLengthCm  149 non-null    float64
 1   SepalWidthCm   149 non-null    float64
 2   PetalLengthCm  149 non-null    float64
 3   PetalWidthCm   149 non-null    float64
 4   Species        149 non-null    object 
dtypes: float64(4), object(1)
memory usage: 5.9+ KB
```

In [15]:
```
iris.isnull().sum()
```

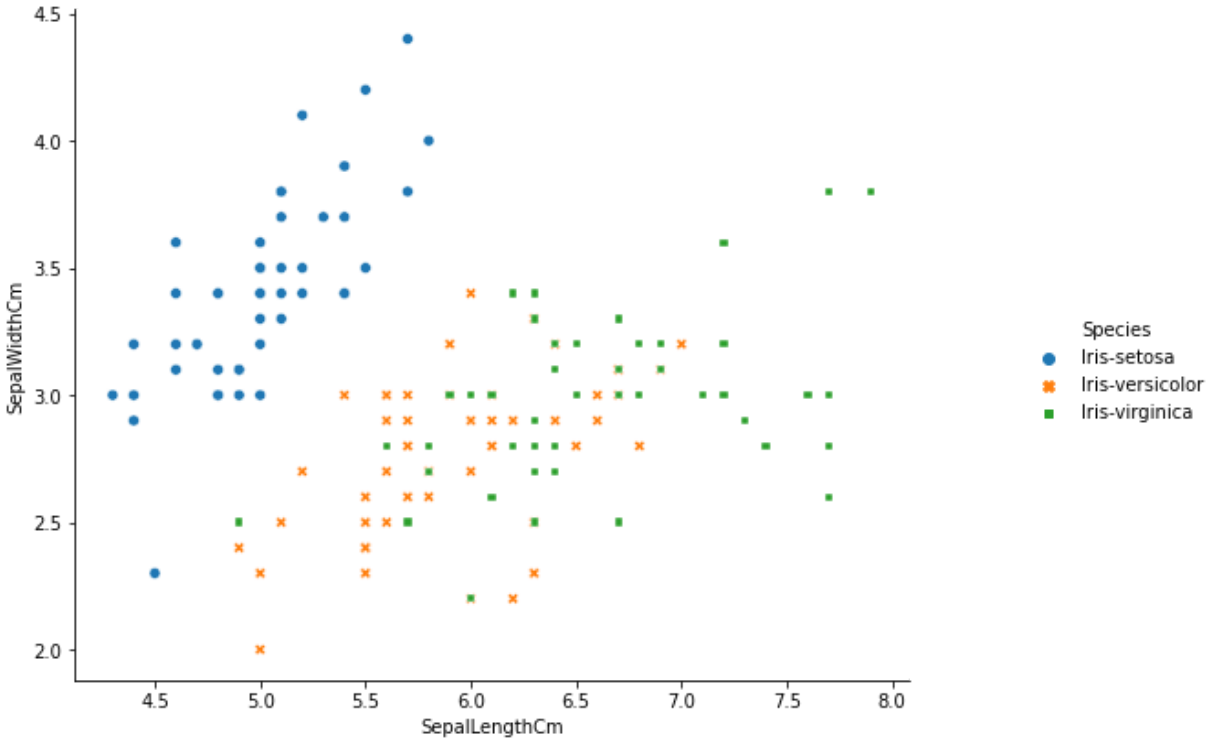
Out[15]:
```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [16]:
```
import missingno as msno
msno.bar(iris, figsize=(6,6), color="red")
plt.show()
```



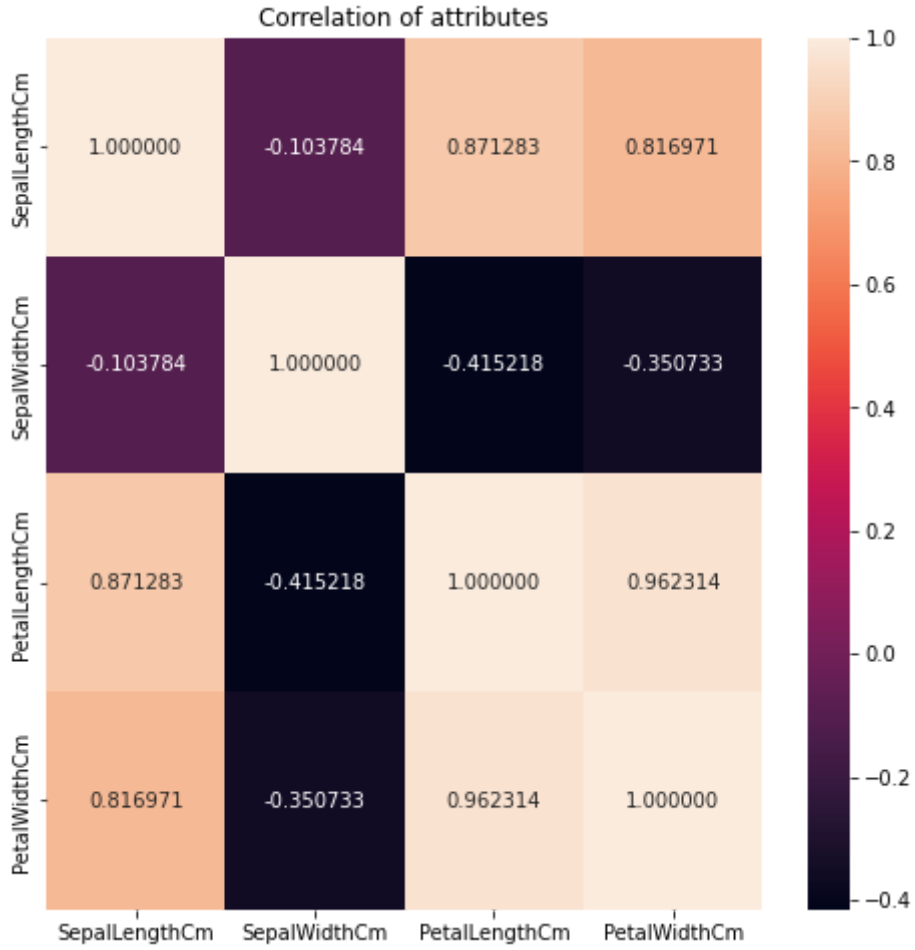
In [17]:

```
scatter = sns.relplot(x="SepalLengthCm", y="SepalWidthCm", data=iris, hue="Species",
scatter.fig.set_size_inches(10,6)
plt.show()
```



In [18]:

```
plt.subplots(figsize = (8,8))
sns.heatmap(iris.corr(), annot=True, fmt="f").set_title("Correlation of attributes")
plt.show()
```



In [19]:
```
from sklearn.model_selection import train_test_split
```

In [20]:
```
X = iris.iloc[:,0:4].values
y = iris.iloc[:,4].values
```

In [21]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
```

With test_size = 0.3, efficiency = 89%

In [22]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_stat
```

In [23]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

NAIVE BAYERS

In [24]:
```
from sklearn.naive_bayes import GaussianNB
```

In [25]:
```
nvclassifier = GaussianNB()
nvclassifier.fit(X_train, y_train)
```

Out[25]:
```
GaussianNB()
```

In [26]:
```
y_pred = nvclassifier.predict(X_test)
print(y_pred)

from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test,y_pred)
print(accuracy)
```

```
[2 2 1 1 0 2 2 1 1 1 0 0 1 0 0 1 2 1 0 0 0 0 1 0 1 1 1 0 0 2 0 1 1 2 0 1 2
 1 1 0 2 0 1 2 2 2 1 1 1 2 2 1 2 2 1 2 1 0 1 1 1 1 2 1 2 0 0 2 1 0 0 1 0 2
 2]
0.9333333333333333
```

In [27]:
```
#Metrics
from sklearn.metrics import make_scorer, accuracy_score,precision_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score ,precision_score,recall_score,f1_score

#Model Select
from sklearn.model_selection import KFold,train_test_split,cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.linear_model import  LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn import linear_model
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.naive_bayes import GaussianNB
```

In [28]:
```
gaussian = GaussianNB()
gaussian.fit(X_train, y_train)
Y_pred = gaussian.predict(X_test)
accuracy_nb=round(accuracy_score(y_test,Y_pred)* 100, 2)
acc_gaussian = round(gaussian.score(X_train, y_train) * 100, 2)

cm = confusion_matrix(y_test, Y_pred)
accuracy = accuracy_score(y_test,Y_pred)
precision =precision_score(y_test, Y_pred,average='micro')
recall =  recall_score(y_test, Y_pred,average='micro')
f1 = f1_score(y_test,Y_pred,average='micro')
print('Confusion matrix for Naive Bayes\n',cm)
print('accuracy_Naive Bayes: %.3f' %accuracy)
print('precision_Naive Bayes: %.3f' %precision)
print('recall_Naive Bayes: %.3f' %recall)
print('f1-score_Naive Bayes : %.3f' %f1)
```

```
Confusion matrix for Naive Bayes
 [[22  0  0]
 [ 0 28  2]
 [ 0  3 20]]
accuracy_Naive Bayes: 0.933
precision_Naive Bayes: 0.933
recall_Naive Bayes: 0.933
f1-score_Naive Bayes : 0.933
```

DECISION TREE

In [30]:
```
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
Y_pred = decision_tree.predict(X_test)
accuracy_dt=round(accuracy_score(y_test,Y_pred)* 100, 2)
acc_decision_tree = round(decision_tree.score(X_train, y_train) * 100, 2)

cm = confusion_matrix(y_test, Y_pred)
accuracy = accuracy_score(y_test,Y_pred)
precision =precision_score(y_test, Y_pred,average='micro')
recall =  recall_score(y_test, Y_pred,average='micro')
f1 = f1_score(y_test,Y_pred,average='micro')
print('Confusion matrix for DecisionTree\n',cm)
print('accuracy_DecisionTree: %.3f' %accuracy)
print('precision_DecisionTree: %.3f' %precision)
print('recall_DecisionTree: %.3f' %recall)
print('f1-score_DecisionTree : %.3f' %f1)
```

```
Confusion matrix for DecisionTree
 [[22  0  0]
 [ 0 28  2]
 [ 0  2 21]]
accuracy_DecisionTree: 0.947
precision_DecisionTree: 0.947
recall_DecisionTree: 0.947
f1-score_DecisionTree : 0.947
```