In [151]:
```
# Name : Mohmmed Khaled AL-zhrani
# 2041606

#SVM vs LR vs RF
# on Iris dataset
```

In [152]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [153]:
```
# read the data by uasing Pandas, Note : We can use load data from sklearn but i
df = pd.read_csv("Iris.csv")
df.head(15)
```

Out[153]:

	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
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa
10	11	5.4	3.7	1.5	0.2	Iris-setosa
11	12	4.8	3.4	1.6	0.2	Iris-setosa
12	13	4.8	3.0	1.4	0.1	Iris-setosa
13	14	4.3	3.0	1.1	0.1	Iris-setosa
14	15	5.8	4.0	1.2	0.2	Iris-setosa

In [154]:
```
# try to understand the data
df.describe()
```

Out[154]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [155]:
```
df.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 [156]:
```
# make sure there's no missing data into dataset
df.isnull().sum()
```

Out[156]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [157]:
```
# counts the flowers
df['Species'].value_counts()
```

Out[157]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

In [158]:
```
from sklearn.preprocessing import LabelEncoder
le_Species = LabelEncoder()
```

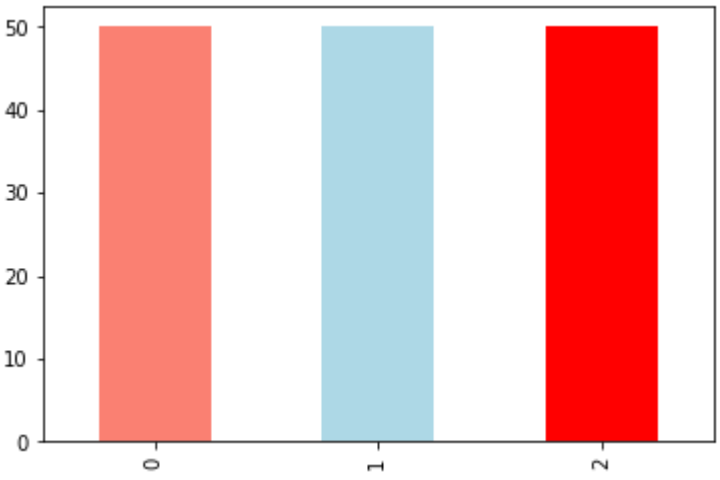
In [159]:
```
df['Species'] = le_Species.fit_transform(df['Species'])
```

In [160]:
```
# ensure it's 0,1,2
df.head(10)
```

Out[160]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	0
1	2	4.9	3.0	1.4	0.2	0
2	3	4.7	3.2	1.3	0.2	0
3	4	4.6	3.1	1.5	0.2	0
4	5	5.0	3.6	1.4	0.2	0
5	6	5.4	3.9	1.7	0.4	0
6	7	4.6	3.4	1.4	0.3	0
7	8	5.0	3.4	1.5	0.2	0
8	9	4.4	2.9	1.4	0.2	0
9	10	4.9	3.1	1.5	0.1	0

In [161]:
```
# 50 for each flower !
df['Species'].value_counts().plot(kind="bar", color=["salmon", "lightblue", "red"
```



In [179]:
```
# separate between features and target(label)
x = df.drop('Species',axis='columns')
y = df.Species
x.head()
```

Out[179]:

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

In [180]:
```
y.head()
```

Out[180]:
```
0    0
1    0
2    0
3    0
4    0
Name: Species, dtype: int32
```

In [181]:
```
# Split the dataset into train (80%) and test (20%)
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)
```

In [182]:
```
# Import the RandomForestClassifier, create the RF classifier, and train the mode
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=20)
model.fit(X_train, Y_train)
```

Out[182]:
```
RandomForestClassifier(n_estimators=20)
```

In [183]:
```
# Calculate the score of the model on the testing data
model.score(X_test, Y_test)
```

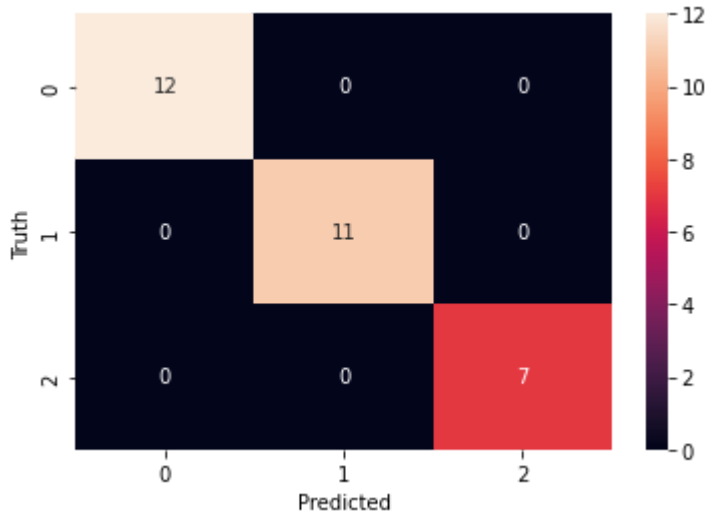
Out[183]:
```
1.0
```

In [184]:
```
# Confusion Matrix of the predicted vales of the model and the ground truth (real
y_pred = model.predict(X_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, y_pred)
cm
```

Out[184]:
```
array([[12,  0,  0],
       [ 0, 11,  0],
       [ 0,  0,  7]], dtype=int64)
```

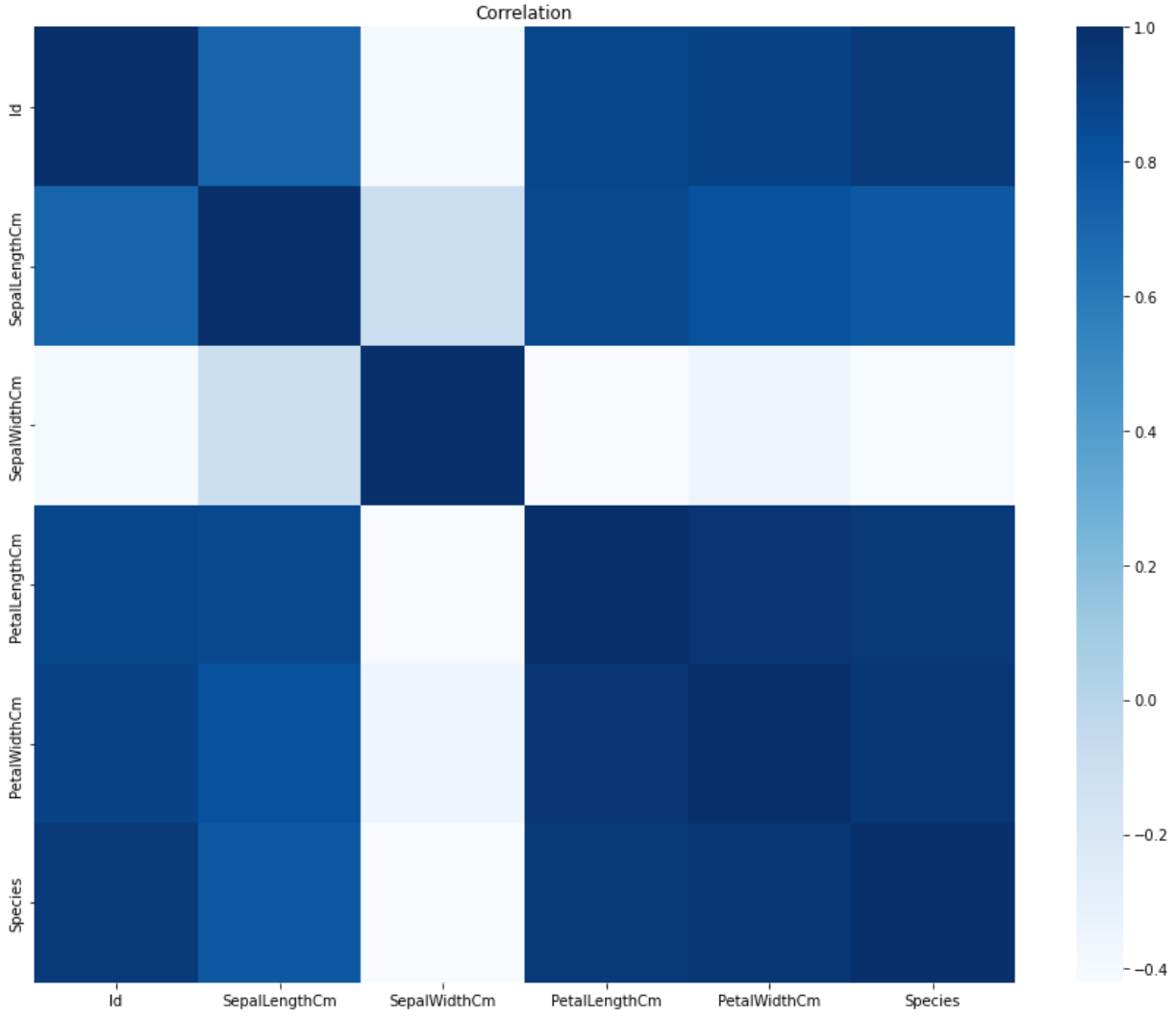
In [185]:
```
# Plot the confusion matrix
import matplotlib.pyplot as plt
import seaborn as sn
plt.figure()
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[185]: Text(33.0, 0.5, 'Truth')



In [186]:
```
# Plot the dataset to get an idea of the data and how it is distributed
import matplotlib.pyplot as plt
import seaborn as sns
corr = df.corr()
fig = plt.figure(figsize=(15,12) )
r = sns.heatmap(corr, cmap='Blues')
r.set_title("Correlation")
```

Out[186]: Text(0.5, 1.0, 'Correlation')



In [187]:
```
# Now we we will use SVM and LR !
```

In [188]:
```
# Import the logistic regression package and create linear regression object.
from sklearn.linear_model import LogisticRegression
lgrgmodel = LogisticRegression()
lgrgmodel.fit(X_train, Y_train)
```

```
C:\Users\PC\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[188]:
```
LogisticRegression()
```

In [189]:
```
# Make predictions using the testing set and show the results
lgrg_pred =lgrgmodel.predict(X_test)
lgrg_pred
```

Out[189]:
```
array([0, 1, 0, 2, 2, 0, 1, 0, 1, 0, 1, 0, 0, 0, 2, 1, 1, 0, 2, 1, 1, 2,
       2, 0, 0, 1, 1, 1, 0, 2])
```

In [190]:
```
lgrgmodel.score(X_test,Y_test)
```

Out[190]:
```
1.0
```

In [191]:
```
lgrgmodel.score(X_train,Y_train)
```

Out[191]:
```
1.0
```

In [192]:
```
# Import the SVC which makes support vector classification by using SVM and creat
from sklearn.svm import SVC
svm = SVC(random_state = 1)
```

In [193]:
```
#Train the model using the training sets.
svm.fit(X_train,Y_train)
```

Out[193]:
```
SVC(random_state=1)
```

In [194]:
```
#calc the accuracy
print("train accuracy: ",svm.score(X_train,Y_train))
print("test accuracy: ",svm.score(X_test,Y_test))
```

```
train accuracy:  0.9916666666666667
test accuracy:  1.0
```

In [195]:
```
# compare scores between Random Forest , logistic Regression, SVM [RF and LR is b

print("Random Forest")
print("train accuracy: ",model.score(X_test, Y_test))
print("test accuracy:  ",model.score(X_test, Y_test))

print("logistic Regression")
print("train accuracy: ",lgrgmodel.score(X_train,Y_train))
print("test accuracy:  ",lgrgmodel.score(X_test,Y_test))

print("=====SVM")
print("train accuracy: ",svm.score(X_train,Y_train))
print("test accuracy:  ",svm.score(X_test,Y_test))


print("logistic Regression and Random Forest is better")
```

```
Random Forest
train accuracy:  1.0
test accuracy:   1.0
logistic Regression
train accuracy:  1.0
test accuracy:   1.0
=====SVM
train accuracy:  0.9916666666666667
test accuracy:   1.0
logistic Regression and Random Forest is better
```