In [2]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

In [3]:
```python
data = pd.read_csv(r"D:\download\archive (8)\IRIS.csv")
```

In [4]:
```python
data.head()
```

Out[4]:

	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 [5]:
```python
data.info
```

Out[5]:
```
<bound method DataFrame.info 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 [11]:
```python
from sklearn.datasets import load_iris
data = load_iris()
print(data.DESCR)
```

```
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

.. topic:: References

   - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
     Mathematical Statistics" (John Wiley, NY, 1950).
   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
     Structure and Classification Rule for Recognition in Partially Exposed
     Environments".  IEEE Transactions on Pattern Analysis and Machine
     Intelligence, Vol. PAMI-2, No. 1, 67-71.
   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions
     on Information Theory, May 1972, 431-433.
   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
     conceptual clustering system finds 3 classes in the data.
   - Many, many more ...
```

In [13]:
```python
data.keys()
```

Out[13]: dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module', 'columns'])

In [14]:
```python
print(data.data[:5])
print(data.feature_names)
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

In [15]:
```python
print(data.target)
print(data.target_names)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
['setosa' 'versicolor' 'virginica']
```

In [17]:
```python
df = pd.DataFrame(data.data)
df.head()
```

Out[17]:

	0	1	2	3
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [18]:
```python
df.columns = ['sepal length', 'sepal width', 'petal length', 'petal width']
df.head()
```

Out[18]:

	sepal length	sepal width	petal length	petal width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [19]:
```python
target = pd.DataFrame(data.target)
target = target.rename(columns = {0: 'target'})
print(target.head())
print('all target column have 3 value: ',target.target.unique())
```

```
   target
0       0
1       0
2       0
3       0
4       0
all target column have 3 value:  [0 1 2]
```

In [20]:
```python
df = pd.concat([df, target], axis = 1)
df.head()
```

Out[20]:

	sepal length	sepal width	petal length	petal width	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [21]:
```python
df.describe()
```

Out[21]:

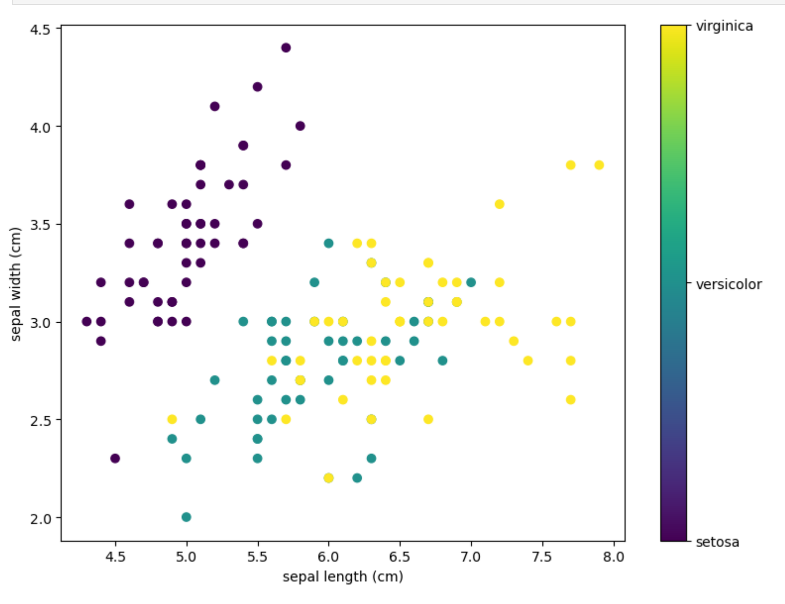
	sepal length	sepal width	petal length	petal width	target
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

In [22]:
```python
print('see what type of data you have:\n',df.dtypes)
print('<---------------------------------------------------->')
print('check for missing values:\n',df.isnull().sum())
```

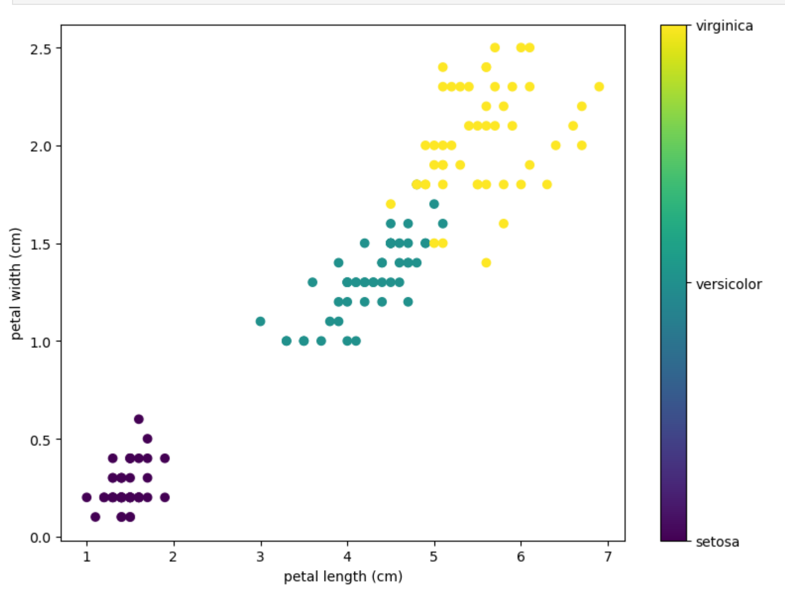
```
see what type of data you have:
 sepal length    float64
sepal width     float64
petal length    float64
petal width     float64
target            int32
dtype: object
<---------------------------------------------------->
check for missing values:
 sepal length    0
sepal width     0
petal length    0
petal width     0
target          0
dtype: int64
```

In [23]:
```python
plt.rcParams['figure.figsize'] = [10,5]
sns.heatmap(df.corr(), annot = True);
```

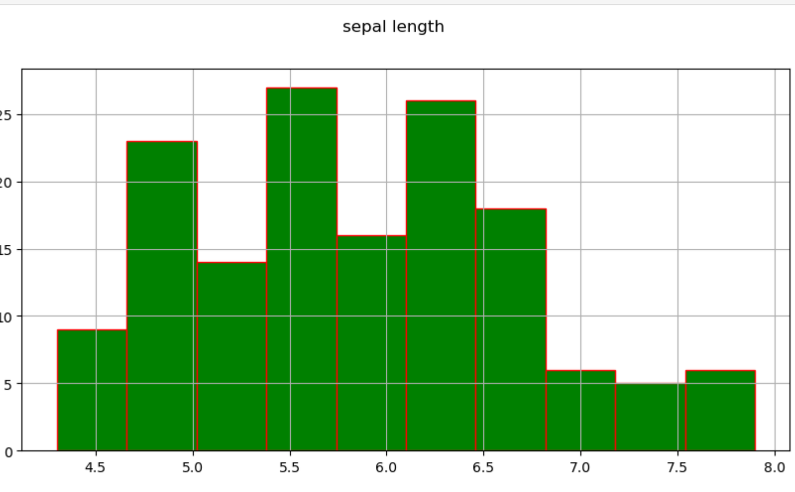
In [24]:
```python
x_index = 0
y_index = 1

formatter = plt.FuncFormatter(lambda i, *args:data.target_names[int(i)])
plt.figure(figsize=(8,6))
plt.scatter(data.data[:, x_index], data.data[:, y_index], c=data.target)
plt.colorbar(ticks=[0, 1, 2], format=formatter)
plt.xlabel(data.feature_names[x_index])
plt.ylabel(data.feature_names[y_index])
plt.tight_layout()
plt.show()
```

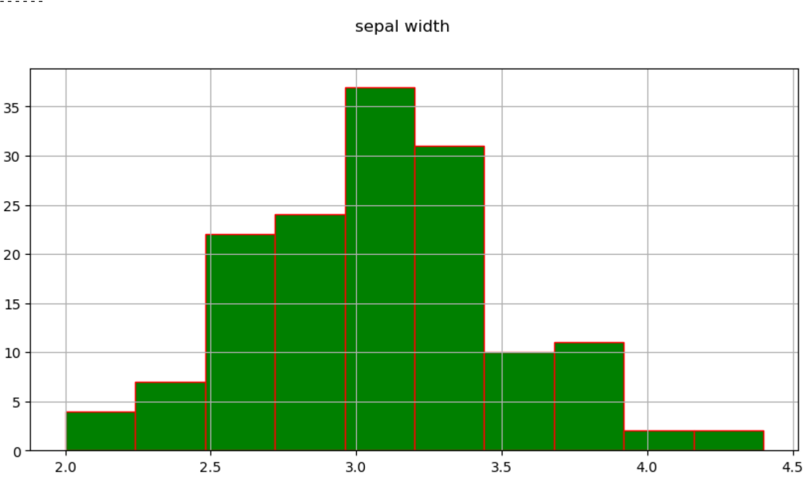


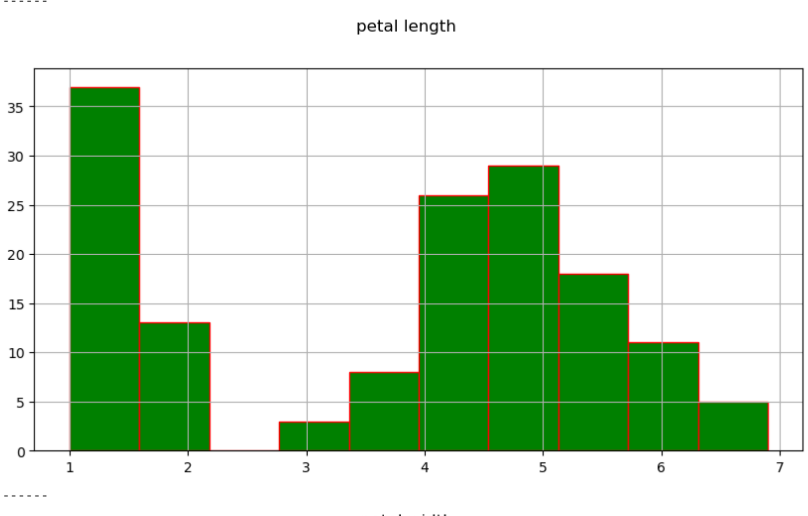
In [25]:
```python
x_index = 2
y_index = 3

formatter = plt.FuncFormatter(lambda i, *args:data.target_names[int(i)])
plt.figure(figsize=(8,6))
plt.scatter(data.data[:, x_index], data.data[:, y_index], c=data.target)
plt.colorbar(ticks=[0, 1, 2], format=formatter)
plt.xlabel(data.feature_names[x_index])
plt.ylabel(data.feature_names[y_index])
plt.tight_layout()
plt.show()
```

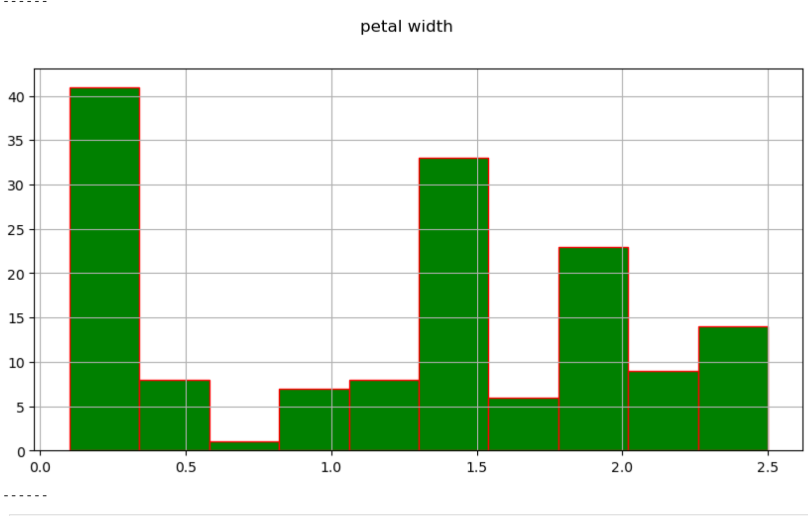


In [28]:
```python
for feat in ['sepal length', 'sepal width', 'petal length', 'petal width']:
    df[feat].hist(ec='red',color='green')
    plt.suptitle(feat)
    plt.show()
    print('------')
```









In [30]:
```python
X = df.copy()
y = X.pop('target')
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1, stratify = y)
```

In [31]:
```python
print('How split: \n')
print('X train', X_train.shape)
print('y train', y_train.shape)
print('X test', X_test.shape)
print('y test', y_test.shape)
```

```
How split:

X train (120, 4)
y train (120,)
X test (30, 4)
y test (30,)
```

In [32]:
```python
scaler = StandardScaler()
X_train = pd.DataFrame(scaler.fit_transform(X_train), columns=X_train.columns)
X_test = pd.DataFrame(scaler.transform(X_test), columns=X_test.columns)
```

In [33]:
```python
print(df.target.value_counts(normalize= True))
print('\n The baseline prediction for this model is 1/3')
```

```
0    0.333333
1    0.333333
2    0.333333
Name: target, dtype: float64

 The baseline prediction for this model is 1/3
```

Logstic Regression model

In [34]:
```python
lg = LogisticRegression()
lg.fit(X_train, y_train)
print('accuracy of model: ',lg.score(X_test, y_test))
cvs = cross_val_score(lg, X_train, y_train, cv=10)
print('accuracy of model after CVS: ',np.mean(cvs))
```

```
accuracy of model:  0.9666666666666667
accuracy of model after CVS:  0.9499999999999998
```

In [35]:
```python
df_coef = pd.DataFrame(lg.coef_, columns=X_train.columns)
df_coef
```

Out[35]:

	sepal length	sepal width	petal length	petal width
0	-1.102746	1.001818	-1.836891	-1.667978
1	0.402982	-0.323432	-0.277761	-0.650011
2	0.699764	-0.678386	2.114653	2.317989

In [36]:
```python
predict = lg.predict(X_test)
compar = pd.DataFrame({'actual': y_test, 'predicted': predict})
compar = compar.reset_index(drop = True)
compar[:10]
```

Out[36]:

	actual	predicted
0	2	2
1	0	0
2	1	1
3	0	0
4	0	0
5	0	0
6	2	2
7	2	2
8	2	2
9	1	1

In [37]:
```python
pd.DataFrame(confusion_matrix(y_test, predict, labels=[2, 1, 0]),
             index=[2, 1, 0], columns=[2, 1, 0])
```

Out[37]:

	2	1	0
2	9	1	0
1	0	10	0
0	0	0	10

In [38]:
```python
print(classification_report(y_test, predict))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       0.91      1.00      0.95        10
           2       1.00      0.90      0.95        10

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
```

```
weighted avg       0.97      0.97      0.97        30
```

In [ ]: