IRIS CLASSIFICATION

Introduction

This project focuses on the classification of the Iris dataset, one of the most well-known datasets in the field of machine learning. The Iris dataset contains measurements of sepal length, sepal width, petal length, and petal width for three species of Iris flowers: Setosa, Versicolor, and Virginica. The goal of this project is to build various machine learning models to classify the species of a flower based on its features.

Objectives

1. Perform exploratory data analysis (EDA) to understand the dataset.
2. Preprocess the data to make it suitable for machine learning models.
3. Train and evaluate multiple machine learning models, such as:
    - Logistic Regression
    - Decision Trees
    - Random Forest
    - Support Vector Machines (SVM)
    - K-Nearest Neighbors (KNN)
4. Compare the performance of these models to identify the best-performing one.
5. Implement a live interactive dashboard using Streamlit to allow users to input flower measurements and predict the species in real-time.

This project will demonstrate the end-to-end process of building a machine learning application, from data analysis to deployment.

In [17]:
```
import pandas as pd
import numpy as np
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import RandomOverSampler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier, export_text
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm
```

In [18]:
```
iris = load_iris()
print(iris)
```

```
{'data': array([[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],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [5.2, 3.5, 1.5, 0.2],
       [5.2, 3.4, 1.4, 0.2],
       [4.7, 3.2, 1.6, 0.2],
       [4.8, 3.1, 1.6, 0.2],
       [5.4, 3.4, 1.5, 0.4],
       [5.2, 4.1, 1.5, 0.1],
       [5.5, 4.2, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.2],
       [5. , 3.2, 1.2, 0.2],
       [5.5, 3.5, 1.3, 0.2],
       [4.9, 3.6, 1.4, 0.1],
       [4.4, 3. , 1.3, 0.2],
       [5.1, 3.4, 1.5, 0.2],
       [5. , 3.5, 1.3, 0.3],
       [4.5, 2.3, 1.3, 0.3],
       [4.4, 3.2, 1.3, 0.2],
       [5. , 3.5, 1.6, 0.6],
       [5.1, 3.8, 1.9, 0.4],
       [4.8, 3. , 1.4, 0.3],
       [5.1, 3.8, 1.6, 0.2],
       [4.6, 3.2, 1.4, 0.2],
       [5.3, 3.7, 1.5, 0.2],
       [5. , 3.3, 1.4, 0.2],
       [7. , 3.2, 4.7, 1.4],
       [6.4, 3.2, 4.5, 1.5],
       [6.9, 3.1, 4.9, 1.5],
       [5.5, 2.3, 4. , 1.3],
       [6.5, 2.8, 4.6, 1.5],
       [5.7, 2.8, 4.5, 1.3],
```

```
       [6.3, 3.3, 4.7, 1.6],
       [4.9, 2.4, 3.3, 1. ],
       [6.6, 2.9, 4.6, 1.3],
       [5.2, 2.7, 3.9, 1.4],
       [5. , 2. , 3.5, 1. ],
       [5.9, 3. , 4.2, 1.5],
       [6. , 2.2, 4. , 1. ],
       [6.1, 2.9, 4.7, 1.4],
       [5.6, 2.9, 3.6, 1.3],
       [6.7, 3.1, 4.4, 1.4],
       [5.6, 3. , 4.5, 1.5],
       [5.8, 2.7, 4.1, 1. ],
       [6.2, 2.2, 4.5, 1.5],
       [5.6, 2.5, 3.9, 1.1],
       [5.9, 3.2, 4.8, 1.8],
       [6.1, 2.8, 4. , 1.3],
       [6.3, 2.5, 4.9, 1.5],
       [6.1, 2.8, 4.7, 1.2],
       [6.4, 2.9, 4.3, 1.3],
       [6.6, 3. , 4.4, 1.4],
       [6.8, 2.8, 4.8, 1.4],
       [6.7, 3. , 5. , 1.7],
       [6. , 2.9, 4.5, 1.5],
       [5.7, 2.6, 3.5, 1. ],
       [5.5, 2.4, 3.8, 1.1],
       [5.5, 2.4, 3.7, 1. ],
       [5.8, 2.7, 3.9, 1.2],
       [6. , 2.7, 5.1, 1.6],
       [5.4, 3. , 4.5, 1.5],
       [6. , 3.4, 4.5, 1.6],
       [6.7, 3.1, 4.7, 1.5],
       [6.3, 2.3, 4.4, 1.3],
       [5.6, 3. , 4.1, 1.3],
       [5.5, 2.5, 4. , 1.3],
       [5.5, 2.6, 4.4, 1.2],
       [6.1, 3. , 4.6, 1.4],
       [5.8, 2.6, 4. , 1.2],
       [5. , 2.3, 3.3, 1. ],
       [5.6, 2.7, 4.2, 1.3],
       [5.7, 3. , 4.2, 1.2],
       [5.7, 2.9, 4.2, 1.3],
       [6.2, 2.9, 4.3, 1.3],
       [5.1, 2.5, 3. , 1.1],
       [5.7, 2.8, 4.1, 1.3],
       [6.3, 3.3, 6. , 2.5],
       [5.8, 2.7, 5.1, 1.9],
       [7.1, 3. , 5.9, 2.1],
       [6.3, 2.9, 5.6, 1.8],
       [6.5, 3. , 5.8, 2.2],
       [7.6, 3. , 6.6, 2.1],
       [4.9, 2.5, 4.5, 1.7],
       [7.3, 2.9, 6.3, 1.8],
       [6.7, 2.5, 5.8, 1.8],
       [7.2, 3.6, 6.1, 2.5],
       [6.5, 3.2, 5.1, 2. ],
       [6.4, 2.7, 5.3, 1.9],
```

```
       [6.8, 3. , 5.5, 2.1],
       [5.7, 2.5, 5. , 2. ],
       [5.8, 2.8, 5.1, 2.4],
       [6.4, 3.2, 5.3, 2.3],
       [6.5, 3. , 5.5, 1.8],
       [7.7, 3.8, 6.7, 2.2],
       [7.7, 2.6, 6.9, 2.3],
       [6. , 2.2, 5. , 1.5],
       [6.9, 3.2, 5.7, 2.3],
       [5.6, 2.8, 4.9, 2. ],
       [7.7, 2.8, 6.7, 2. ],
       [6.3, 2.7, 4.9, 1.8],
       [6.7, 3.3, 5.7, 2.1],
       [7.2, 3.2, 6. , 1.8],
       [6.2, 2.8, 4.8, 1.8],
       [6.1, 3. , 4.9, 1.8],
       [6.4, 2.8, 5.6, 2.1],
       [7.2, 3. , 5.8, 1.6],
       [7.4, 2.8, 6.1, 1.9],
       [7.9, 3.8, 6.4, 2. ],
       [6.4, 2.8, 5.6, 2.2],
       [6.3, 2.8, 5.1, 1.5],
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [6.3, 3.4, 5.6, 2.4],
       [6.4, 3.1, 5.5, 1.8],
       [6. , 3. , 4.8, 1.8],
       [6.9, 3.1, 5.4, 2.1],
       [6.7, 3.1, 5.6, 2.4],
       [6.9, 3.1, 5.1, 2.3],
       [5.8, 2.7, 5.1, 1.9],
       [6.8, 3.2, 5.9, 2.3],
       [6.7, 3.3, 5.7, 2.5],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.5, 5. , 1.9],
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]]), 'target': array([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, 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, 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]), 'frame': None, 'targe
t_names': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'), 'DESCR': '.. _
iris_dataset:\n\nIris plants dataset\n--------------------\n\n**Data Set Characteris
tics:**\n\n:Number of Instances: 150 (50 in each of three classes)\n:Number of Attri
butes: 4 numeric, predictive attributes and the class\n:Attribute Information:\n
- sepal length in cm\n    - sepal width in cm\n    - petal length in cm\n    - petal
width in cm\n    - class:\n            - Iris-Setosa\n            - Iris-Versicolour
\n            - Iris-Virginica\n\n:Summary Statistics:\n\n============== ==== ==== =
====== ===== ====================\n                Min  Max   Mean    SD   Class Cor
relation\n============== ==== ==== ======= ===== ====================\nsepal length:
4.3  7.9   5.84   0.83    0.7826\nsepal width:    2.0  4.4   3.05   0.43   -0.4194\n
petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)\npetal width:    0.1  2.5
```

1.20   0.76   0.9565 (high!)\n============== ==== ==== ======= ===== ============
=======\n\n:Missing Attribute Values: None\n:Class Distribution: 33.3% for each of 3
classes.\n:Creator: R.A. Fisher\n:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.
gov)\n:Date: July, 1988\n\nThe famous Iris database, first used by Sir R.A. Fisher.
The dataset is taken\nfrom Fisher\'s paper. Note that it\'s the same as in R, but no
t as in the UCI\nMachine Learning Repository, which has two wrong data points.\n\nTh
is is perhaps the best known database to be found in the\npattern recognition litera
ture. Fisher\'s paper is a classic in the field and\nis referenced frequently to th
is day. (See Duda & Hart, for example.) The\ndata set contains 3 classes of 50 ins
tances each, where each class refers to a\ntype of iris plant. One class is linearl
y separable from the other 2; the\nlatter are NOT linearly separable from each othe
r.\n\n.. dropdown:: References\n\n - Fisher, R.A. "The use of multiple measurements
in taxonomic problems"\n   Annual Eugenics, 7, Part II, 179-188 (1936); also in "Co
ntributions to\n   Mathematical Statistics" (John Wiley, NY, 1950).\n - Duda, R.
O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.\n   (Q327.D83) J
ohn Wiley & Sons. ISBN 0-471-22361-1. See page 218.\n - Dasarathy, B.V. (1980) "N
osing Around the Neighborhood: A New System\n   Structure and Classification Rule f
or Recognition in Partially Exposed\n   Environments". IEEE Transactions on Patter
n Analysis and Machine\n   Intelligence, Vol. PAMI-2, No. 1, 67-71.\n - Gates, G.
W. (1972) "The Reduced Nearest Neighbor Rule". IEEE Transactions\n   on Informatio
n Theory, May 1972, 431-433.\n - See also: 1988 MLC Proceedings, 54-64. Cheeseman
et al"s AUTOCLASS II\n   conceptual clustering system finds 3 classes in the dat
a.\n - Many, many more ...\n', 'feature_names': ['sepal length (cm)', 'sepal width
(cm)', 'petal length (cm)', 'petal width (cm)'], 'filename': 'iris.csv', 'data_modul
e': 'sklearn.datasets.data'}

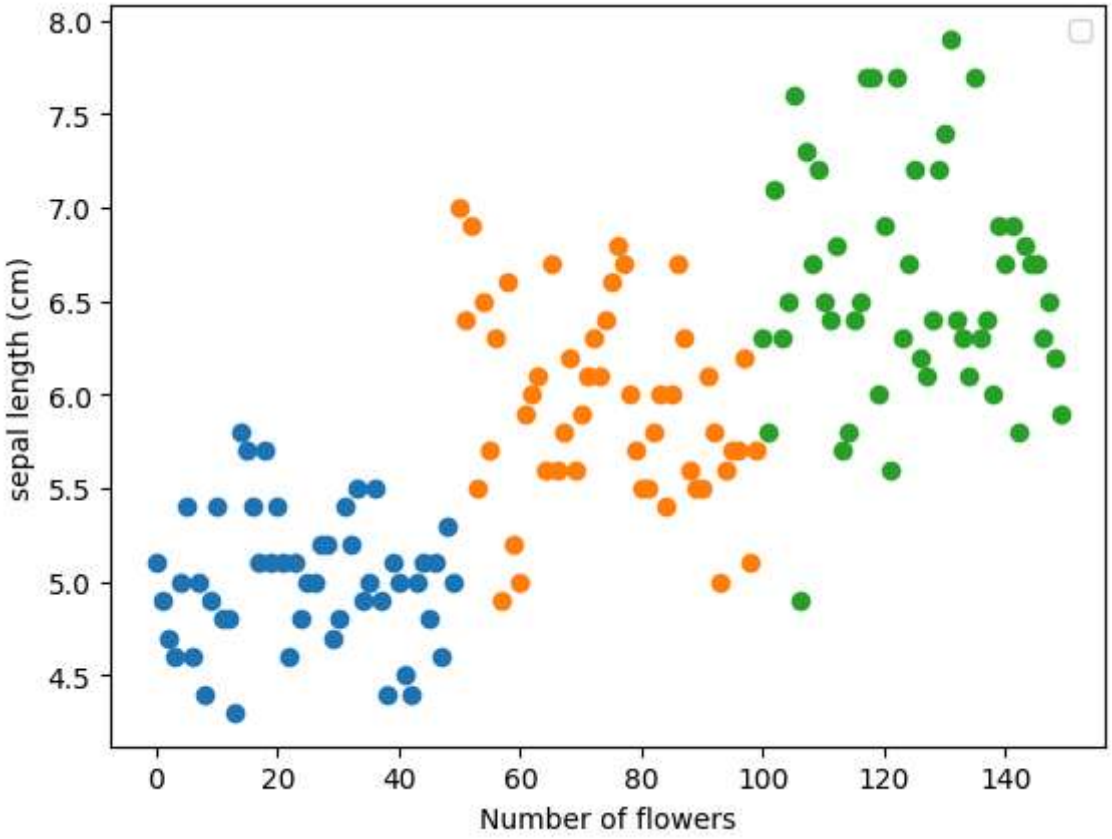
In [19]:
```
data = pd.DataFrame(data=iris.data,columns=iris.feature_names)
col = iris.feature_names
data['species'] = pd.Categorical.from_codes(iris.target, iris.target_names)
data.head()
```

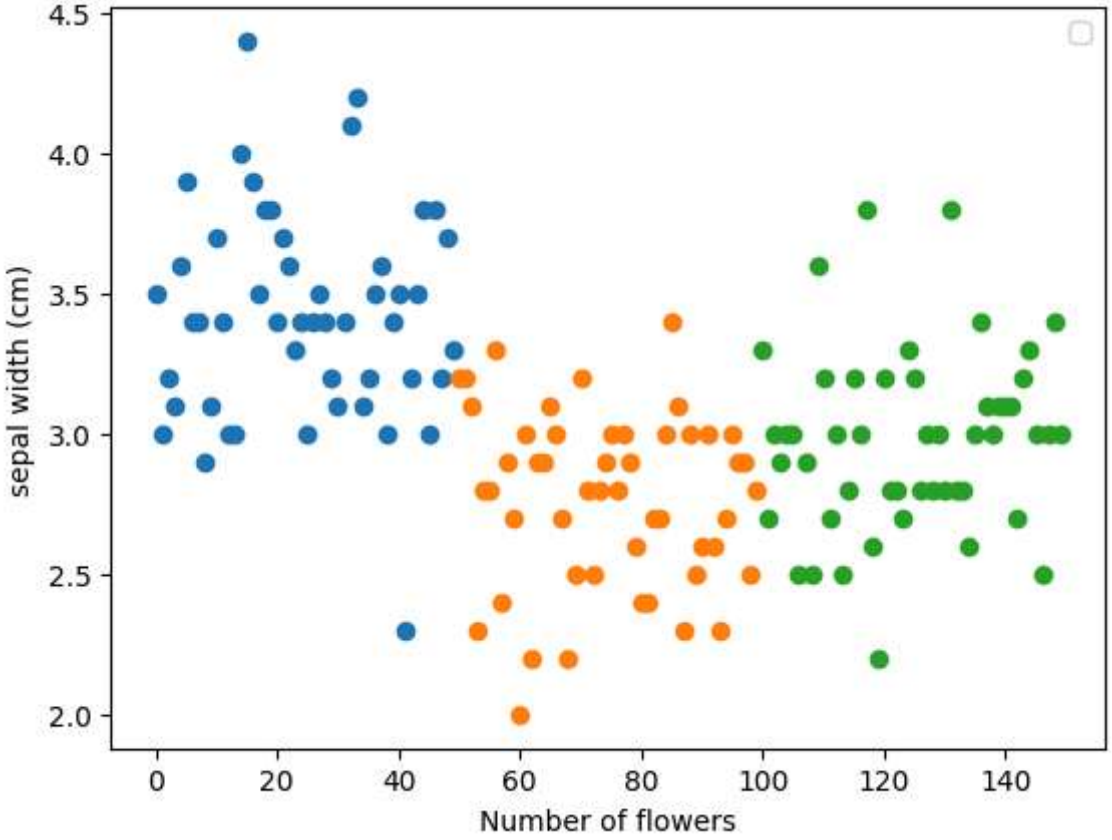
Out[19]:

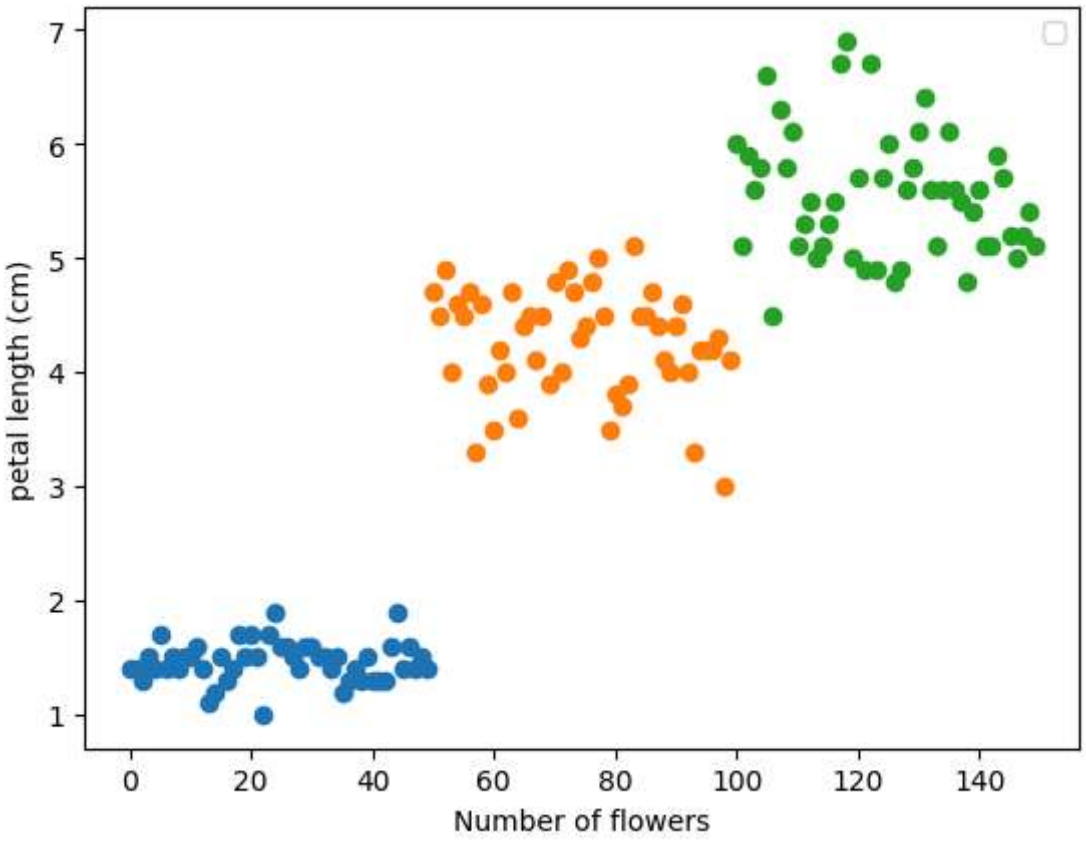
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

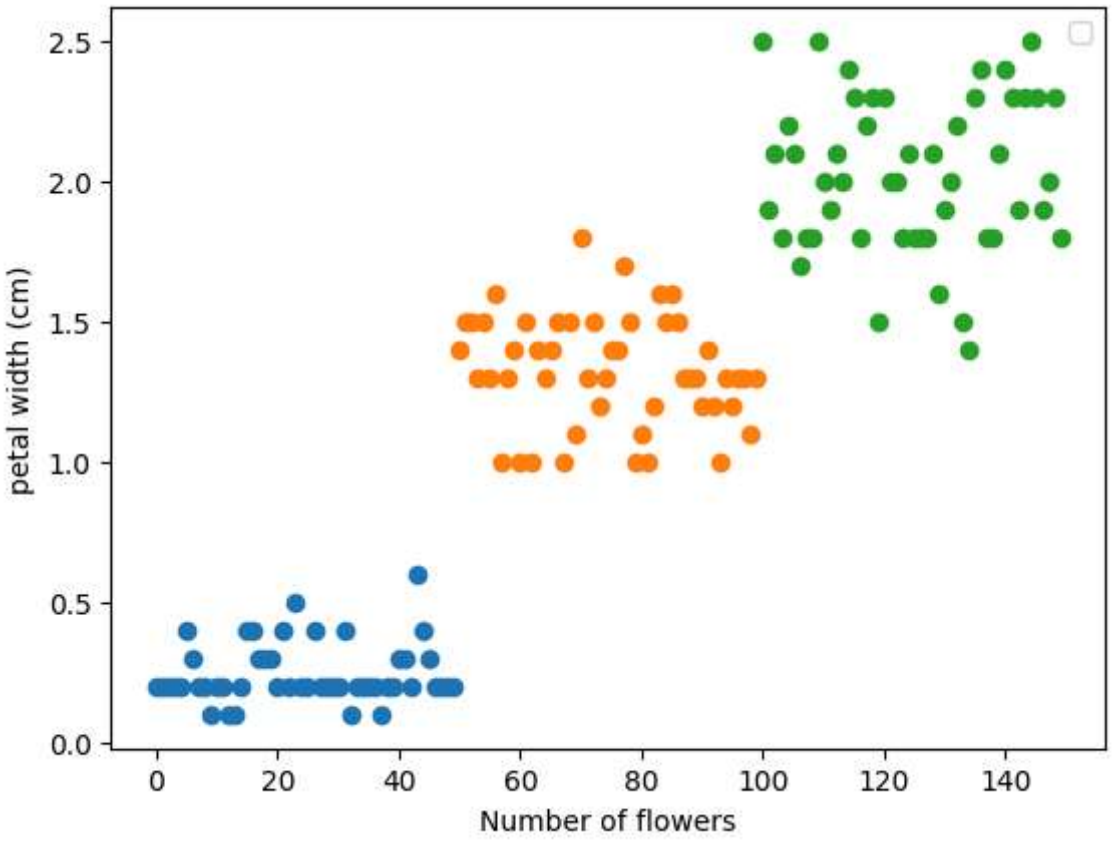
In [20]:
```
for labels in col:
    plt.scatter(data[data['species']=='setosa'].index, data[data['species']=='setos
    plt.scatter(data[data['species']=='versicolor'].index, data[data['species']=='v
    plt.scatter(data[data['species']=='virginica'].index, data[data['species']=='vi
    plt.legend()
    plt.xlabel('Number of flowers')
    plt.ylabel(f'{labels}')
    plt.show()
```

```
C:\Users\apoor\AppData\Local\Temp\ipykernel_33896\3748782141.py:5: UserWarning: No a
rtists with labels found to put in legend.  Note that artists whose label start with
an underscore are ignored when legend() is called with no argument.
  plt.legend()
```


8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
sepal length (cm)
0
20
40
60
80
100
120
140
Number of flowers


4.5
4.0
3.5
3.0
2.5
2.0
sepal width (cm)
0
20
40
60
80
100
120
140
Number of flowers





In [21]:
```
data.describe()
```

Out[21]:

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

In [22]:
```
train, valid, test = np.split(data.sample(frac=1), [int(0.6*len(data)),int(0.8*len(
print(train.shape)
print(valid.shape)
print(test.shape)
```

```
(90, 5)
(30, 5)
(30, 5)
```

```
C:\Users\apoor\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfr
a8p0\LocalCache\local-packages\Python313\site-packages\numpy\_core\fromnumeric.py:5
7: FutureWarning: 'DataFrame.swapaxes' is deprecated and will be removed in a future
version. Please use 'DataFrame.transpose' instead.
  return bound(*args, **kwds)
```

In [23]:
```
print(train[train['species']=="setosa"].shape)
print(train[train['species']=="versicolor"].shape)
print(train[train['species']=="virginica"].shape)
```

```
(35, 5)
(26, 5)
(29, 5)
```

Scaling Data

Scaling of the database is very important as sometimes the distribution of the values of different features can have different ranges which can lead to one feature influence the model and create biases. Scaling destributes the database to a common scale and prevents the influence of a particular or a group of features in the model. Scaling is very crutial when it comes to distance based Machine Learning algorithms like N-nearest and SVM's.

In [24]:
```
def scale_data(data, overfitting = False):
    x = data[data.columns[0:-1]]
    y = data[data.columns[-1]]
    scale = StandardScaler()
    x = scale.fit_transform(x)
    if overfitting:
```

```
        ros = RandomOverSampler()
        x,y = ros.fit_resample(x,y)
    data_scaled = np.hstack((x, np.reshape(y,(-1,1))))

    return data_scaled, x, y
```

In [25]:
```
train, x_train, y_train = scale_data(train, overfitting=True)
valid, x_valid, y_valid = scale_data(valid, overfitting=False)
test, x_test, y_test = scale_data(test, overfitting=False)
```

Logistic Regression

This section uses Logistic Regression for identifying the species.

In [26]:
```
lg = LogisticRegression(multi_class='multinomial', solver='lbfgs',max_iter=200)
lg.fit(x_train,y_train)

score_croos_valid = cross_val_score(lg, x_valid, y_valid, cv=5)
print(score_croos_valid)
print("Cross Validation Score = "+f'{score_croos_valid.mean()}')
```

```
[0.83333333 1.         1.         1.         0.83333333]
Cross Validation Score = 0.9333333333333333
```

```
C:\Users\apoor\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfr
a8p0\LocalCache\local-packages\Python313\site-packages\sklearn\linear_model\_logisti
c.py:1247: FutureWarning: 'multi_class' was deprecated in version 1.5 and will be re
moved in 1.7. From then on, it will always use 'multinomial'. Leave it to its defaul
t value to avoid this warning.
  warnings.warn(
C:\Users\apoor\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfr
a8p0\LocalCache\local-packages\Python313\site-packages\sklearn\linear_model\_logisti
c.py:1247: FutureWarning: 'multi_class' was deprecated in version 1.5 and will be re
moved in 1.7. From then on, it will always use 'multinomial'. Leave it to its defaul
t value to avoid this warning.
  warnings.warn(
C:\Users\apoor\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfr
a8p0\LocalCache\local-packages\Python313\site-packages\sklearn\linear_model\_logisti
c.py:1247: FutureWarning: 'multi_class' was deprecated in version 1.5 and will be re
moved in 1.7. From then on, it will always use 'multinomial'. Leave it to its defaul
t value to avoid this warning.
  warnings.warn(
C:\Users\apoor\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfr
a8p0\LocalCache\local-packages\Python313\site-packages\sklearn\linear_model\_logisti
c.py:1247: FutureWarning: 'multi_class' was deprecated in version 1.5 and will be re
moved in 1.7. From then on, it will always use 'multinomial'. Leave it to its defaul
t value to avoid this warning.
  warnings.warn(
C:\Users\apoor\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfr
a8p0\LocalCache\local-packages\Python313\site-packages\sklearn\linear_model\_logisti
c.py:1247: FutureWarning: 'multi_class' was deprecated in version 1.5 and will be re
moved in 1.7. From then on, it will always use 'multinomial'. Leave it to its defaul
t value to avoid this warning.
  warnings.warn(
C:\Users\apoor\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfr
a8p0\LocalCache\local-packages\Python313\site-packages\sklearn\linear_model\_logisti
c.py:1247: FutureWarning: 'multi_class' was deprecated in version 1.5 and will be re
moved in 1.7. From then on, it will always use 'multinomial'. Leave it to its defaul
t value to avoid this warning.
  warnings.warn(
```

In [27]:

```python
y_pred = lg.predict(x_test)
score = accuracy_score(y_test,y_pred)
print(y_pred)
print("Accuracy Score of Logistic Regression = "+f'{score}')
```

```
['versicolor' 'versicolor' 'setosa' 'versicolor' 'versicolor' 'versicolor'
 'setosa' 'virginica' 'versicolor' 'versicolor' 'versicolor' 'virginica'
 'versicolor' 'setosa' 'setosa' 'versicolor' 'versicolor' 'virginica'
 'versicolor' 'setosa' 'setosa' 'virginica' 'versicolor' 'virginica'
 'versicolor' 'setosa' 'virginica' 'versicolor' 'setosa' 'versicolor']
Accuracy Score of Logistic Regression = 0.8333333333333334
```

Decision Trees

This section uses Decision Trees for identifying the species.

In [28]:
```
dtc = DecisionTreeClassifier()
dtc.fit(x_train,y_train)
y_pred = dtc.predict(x_test)
score = accuracy_score(y_test,y_pred)

print(y_pred)
print("Accuracy Score of Descision trees = "+f'{score}')
```

```
['virginica' 'virginica' 'setosa' 'versicolor' 'versicolor' 'versicolor'
 'setosa' 'virginica' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'setosa' 'setosa' 'versicolor' 'versicolor' 'virginica'
 'versicolor' 'setosa' 'setosa' 'virginica' 'versicolor' 'virginica'
 'versicolor' 'setosa' 'virginica' 'setosa' 'setosa' 'versicolor']
Accuracy Score of Descision trees = 0.8333333333333334
```

In [29]:
```
export_tree = export_text(dtc, feature_names=iris.feature_names)
print(export_tree)
```

```
|--- petal length (cm) <= -0.50
|   |--- class: setosa
|--- petal length (cm) >  -0.50
|   |--- petal length (cm) <= 0.70
|   |   |--- petal width (cm) <= 0.65
|   |   |   |--- class: versicolor
|   |   |--- petal width (cm) >  0.65
|   |   |   |--- sepal length (cm) <= 0.27
|   |   |   |   |--- class: versicolor
|   |   |   |--- sepal length (cm) >  0.27
|   |   |   |   |--- class: virginica
|   |--- petal length (cm) >  0.70
|   |   |--- sepal length (cm) <= 0.27
|   |   |   |--- petal width (cm) <= 0.52
|   |   |   |   |--- class: virginica
|   |   |   |--- petal width (cm) >  0.52
|   |   |   |   |--- class: versicolor
|   |   |--- sepal length (cm) >  0.27
|   |   |   |--- class: virginica
```

Random Forest

This section uses Random Forest for identifying the species.

In [30]:
```
rfc = RandomForestClassifier()
rfc.fit(x_train, y_train)
y_pred = rfc.predict(x_test)
score = accuracy_score(y_pred,y_test)
print(y_pred)
print("Accuracy Score of Random Forest = "+f'{score}')
```

```
['virginica' 'versicolor' 'setosa' 'versicolor' 'versicolor' 'versicolor'
 'setosa' 'virginica' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'setosa' 'setosa' 'versicolor' 'versicolor' 'virginica'
 'versicolor' 'setosa' 'setosa' 'virginica' 'versicolor' 'virginica'
 'versicolor' 'setosa' 'virginica' 'versicolor' 'setosa' 'versicolor']
Accuracy Score of Random Forest = 0.8333333333333334
```

K-Nearest Neighbour

This section uses K-Nearest Neighbour for identifying the species.

In [31]:
```
knn = KNeighborsClassifier()
knn.fit(x_train, y_train)
y_pred = knn.predict(x_test)
score = accuracy_score(y_pred,y_test)
print(y_pred)
print("Accuracy Score of K-Nearest Neighbour = "+f'{score}')
```

```
['versicolor' 'versicolor' 'setosa' 'versicolor' 'versicolor' 'versicolor'
 'setosa' 'virginica' 'versicolor' 'versicolor' 'versicolor' 'virginica'
 'versicolor' 'setosa' 'setosa' 'versicolor' 'versicolor' 'virginica'
 'versicolor' 'setosa' 'setosa' 'virginica' 'versicolor' 'virginica'
 'versicolor' 'setosa' 'virginica' 'versicolor' 'setosa' 'versicolor']
Accuracy Score of K-Nearest Neighbour = 0.8333333333333334
```

Support Vecor Machines

This section uses Support Vecor Machines for identifying the species.

In [32]:
```
svm = svm.SVC(kernel="linear")
svm.fit(x_train,y_train)
y_pred = svm.predict(x_test)
score = accuracy_score(y_pred,y_test)
print(y_pred)
print("Accuracy Score of Support Vecor Machines = "+f'{score}')
```

```
['versicolor' 'versicolor' 'setosa' 'versicolor' 'versicolor' 'versicolor'
 'setosa' 'virginica' 'versicolor' 'versicolor' 'versicolor' 'virginica'
 'versicolor' 'setosa' 'setosa' 'versicolor' 'versicolor' 'virginica'
 'virginica' 'setosa' 'setosa' 'virginica' 'versicolor' 'virginica'
 'versicolor' 'setosa' 'virginica' 'versicolor' 'setosa' 'versicolor']
Accuracy Score of Support Vecor Machines = 0.8666666666666667

Accuracy Score of Support Vecor Machines = 0.8666666666666667
```