In [1]:

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
```

In [9]:

```
df = pd.read_csv('Iris-checkpoint.csv').drop('Id', axis=1)
df
```

Out[9]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	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 × 5 columns

In [10]:

```
X = df.drop('Species', axis=1)
Y = df['Species']
```

In [11]:

```
x = X.values
y = Y.values
```

In [12]:

```
x.shape, y.shape
```

Out[12]:

```
((150, 4), (150,))
```

In [13]:

```
E = LabelEncoder()
# E.fit(y)
E.fit(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'])
y_encoded = E.transform(y)
```

In [14]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y_encoded, test_size=.2)
```

In [15]:

```
E.classes_
```

Out[15]:

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype='<U15')
```

In [16]:

```
E.transform(['Iris-versicolor'])
```

Out[16]:

```
array([1])
```

In [29]:

```
E.inverse_transform([2])
```

Out[29]:

```
array(['Iris-virginica'], dtype='<U15')
```

In [30]:

```
model = LogisticRegression(max_iter=300)
model.fit(x_train, y_train)
model.score(x_train, y_train)
```

Out[30]:

```
0.9666666666666667
```

In [31]:

```
p_test = model.predict(x_test)
p_train = model.predict(x_train)
```

In [32]:

```
f1_score(y_train, p_train, average='micro')
```

Out[32]:

```
0.9666666666666667
```

In [33]:

```
f1_score(y_test, p_test, average='micro')
```

Out[33]:

```
0.9333333333333333
```

In [34]:

```
c_train = confusion_matrix(y_train, p_train, normalize='pred')
c_train
```

Out[34]:

```
array([[1.        , 0.        , 0.        ],
       [0.        , 0.95121951, 0.05555556],
       [0.        , 0.04878049, 0.94444444]])
```

In [35]:

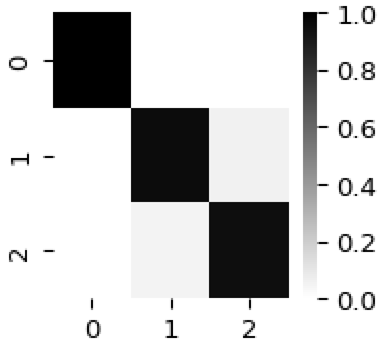
```
c_test = confusion_matrix(y_test, p_test, normalize='pred')
c_test
```

Out[35]:

```
array([[1.        , 0.        , 0.        ],
       [0.        , 0.88888889, 0.07142857],
       [0.        , 0.11111111, 0.92857143]])
```

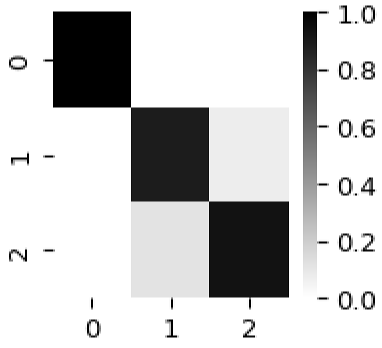
In [36]:

```
# Inches, dpi=100
plt.figure(figsize=(2,2), dpi=95)
plot = sns.heatmap(c_train, vmin=0, vmax=1, cmap='binary');
plot.get_figure().savefig('heatmap_iris_log.png')
```



In [37]:

```
# Inches, dpi=100
plt.figure(figsize=(2,2), dpi=95)
plot = sns.heatmap(c_test, vmin=0, vmax=1, cmap='binary');
```



In [38]:

```
print(classification_report(y_train, p_train))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        43
           1       0.95      0.95      0.95        41
           2       0.94      0.94      0.94        36

    accuracy                           0.97       120
   macro avg       0.97      0.97      0.97       120
weighted avg       0.97      0.97      0.97       120
```

In [39]:

```
print(classification_report(y_test, p_test))
```

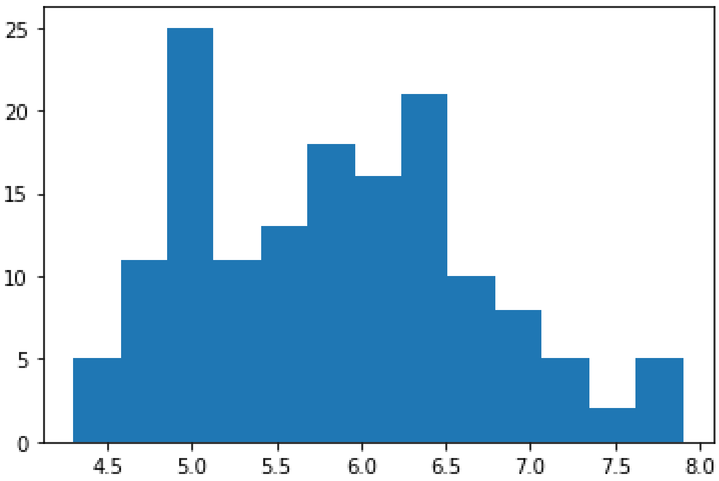
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         7
           1       0.89      0.89      0.89         9
           2       0.93      0.93      0.93        14

    accuracy                           0.93        30
   macro avg       0.94      0.94      0.94        30
weighted avg       0.93      0.93      0.93        30
```

In [40]:

```
plt.hist(x[:, 0], bins='sqrt');
```



In [ ]:

In [ ]:

In [ ]: