In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import itertools
```

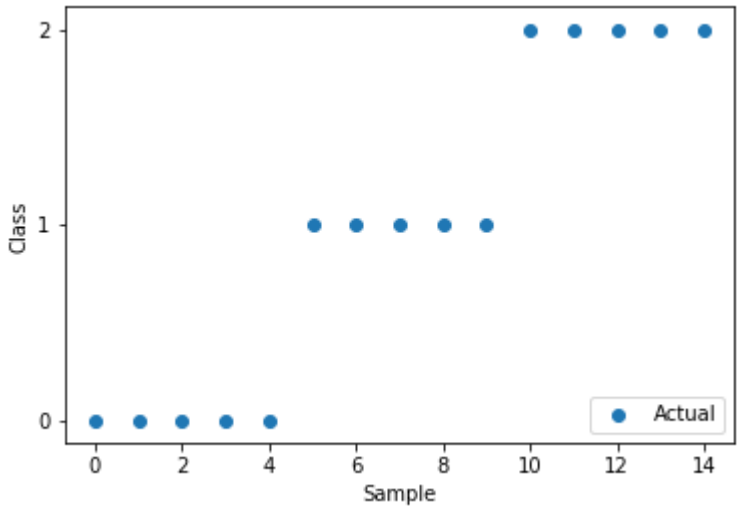
In [3]:
```
models = ['Model 1','Model 2', 'Model 3', 'Model 4']

# Labeled Classes
labels=[0,1,2]

# Class Names
# Setosa = 0, Versicolor = 1, Virginica = 2
classes = ['Setosa','Versicolor','Virginica']

df = pd.read_csv(r'C:\Users\309962\Desktop\IrisSample.csv')
```

In [4]:
```
df.head(15)
```

Out[4]:

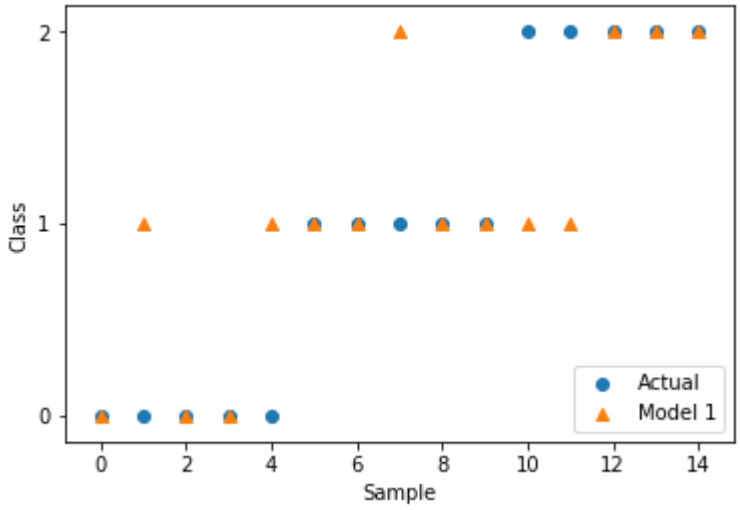
	sepal_length	sepal_width	petal_length	petal_width	class	NumericClass	Model1_Prediction
0	5.8	4.0	1.2	0.2	Iris-setosa	0	0
1	5.7	4.4	1.3	0.4	Iris-setosa	0	0
2	5.1	3.4	1.5	0.2	Iris-setosa	0	0
3	5.4	3.9	1.7	0.4	Iris-setosa	0	0
4	4.3	3.0	1.1	0.1	Iris-setosa	0	0
5	4.9	2.4	3.3	1.0	Iris-versicolor	1	1
6	5.9	3.0	4.2	1.5	Iris-versicolor	1	1
7	6.6	3.0	4.4	1.4	Iris-versicolor	1	2
8	5.0	2.3	3.3	1.0	Iris-versicolor	1	1
9	6.2	2.9	4.3	1.3	Iris-versicolor	1	1
10	5.8	2.7	5.1	1.9	Iris-virginica	2	1
11	7.2	3.6	6.1	2.5	Iris-virginica	2	1
12	6.4	3.2	5.3	2.3	Iris-virginica	2	2
13	7.4	2.8	6.1	1.9	Iris-virginica	2	2
14	6.1	2.6	5.6	1.4	Iris-virginica	2	2

In [5]:
```
plt.figure()
plt.scatter(df.index,df['NumericClass'],label='Actual')
plt.legend(loc=4)
plt.yticks([0,1,2])
plt.xlabel('Sample')
plt.ylabel('Class')
plt.show()
```

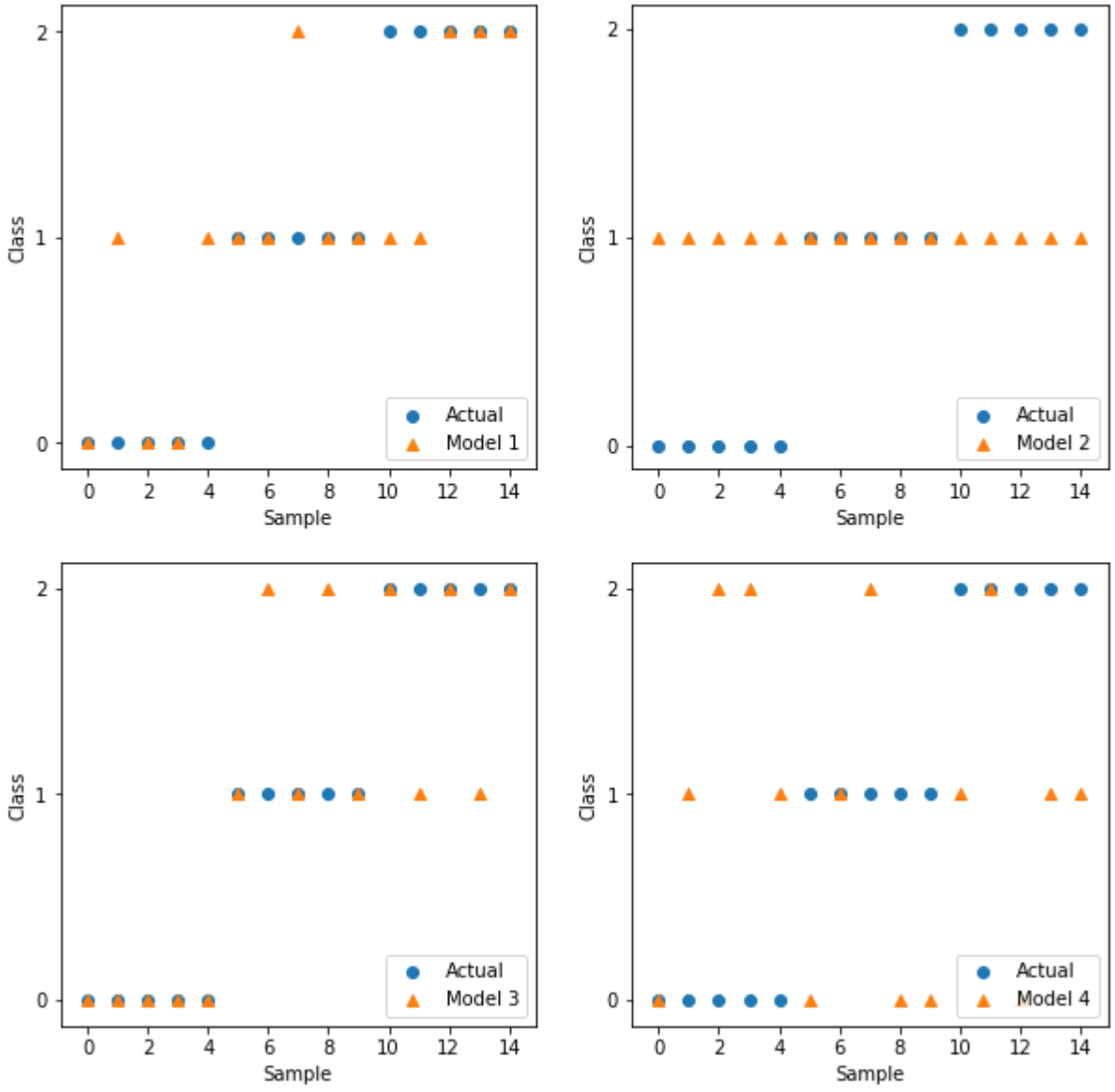


In [6]:

```
# Compare performance of Actual and Model 1 Prediction
plt.figure()
plt.scatter(df.index,df['NumericClass'],label='Actual')
plt.scatter(df.index,df['Model1_Prediction'],label='Model 1',marker='^')
plt.legend(loc=4)
plt.yticks([0,1,2])
plt.xlabel('Sample')
plt.ylabel('Class')
plt.show()
```



In [7]:
```
plt.figure(figsize=(10,10))

for idx, model in enumerate(models):
    plt.subplot(2,2,idx+1)

    plt.scatter(df.index,df['NumericClass'],label='Actual')
    plt.scatter(df.index,df[model.replace(' ','') + '_Prediction'],
                label=model,marker='^')
    plt.yticks([0,1,2])
    plt.legend(loc=4)
    plt.xlabel('Sample')
    plt.ylabel('Class')
```



In [8]:
```
from sklearn.metrics import classification_report,confusion_matrix
```

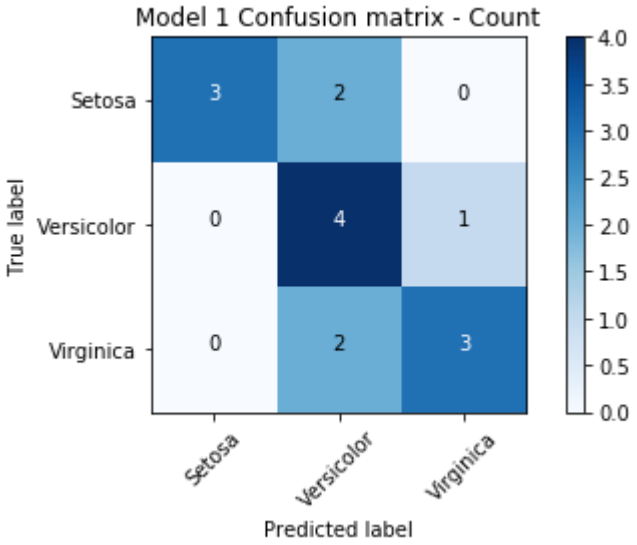
In [9]:
```
# Reference:
# https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_ma
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        #print("Normalized confusion matrix")
    #else:
    #    print('Confusion matrix, without normalization')

    #print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()
```

In [10]:
```
# Compute confusion matrix
cnf_matrix = confusion_matrix(df['NumericClass'],
                              df['Model1_Prediction'],labels=labels)
```

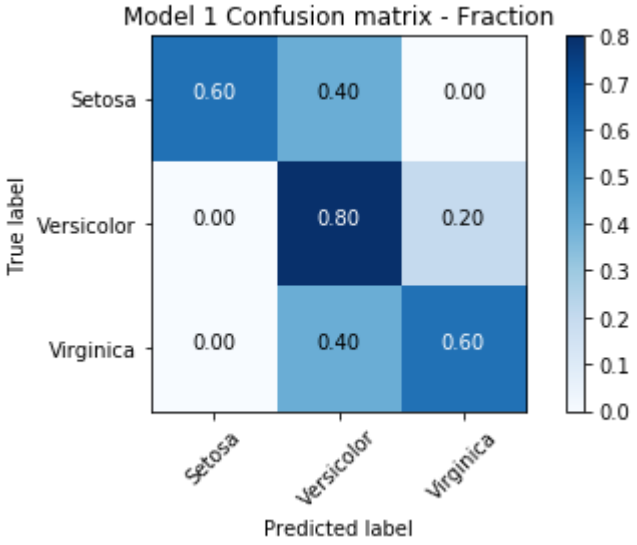
In [11]:
```
cnf_matrix
```

Out[11]:
```
array([[3, 2, 0],
       [0, 4, 1],
       [0, 2, 3]], dtype=int64)
```

In [12]:
```
# Plot confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=classes,
                      title='Model 1 Confusion matrix - Count')
```

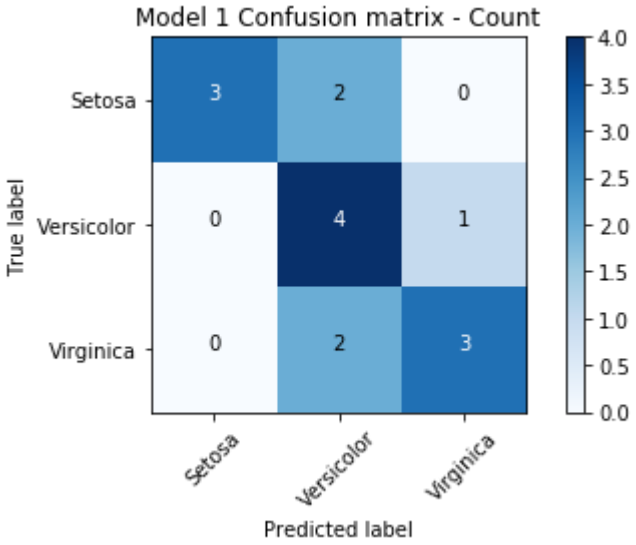


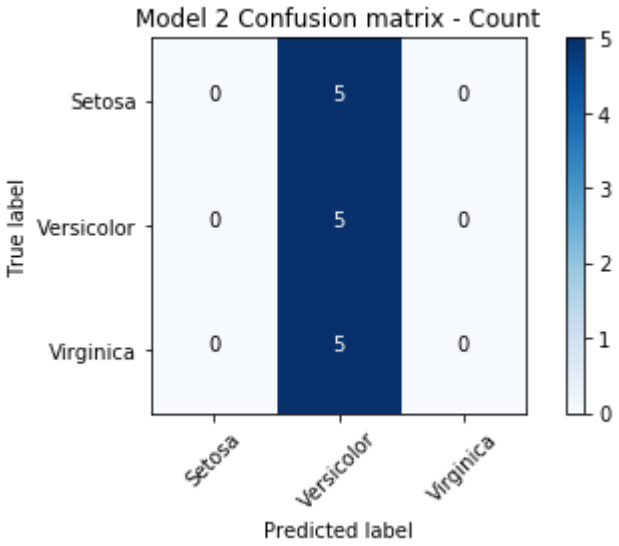
In [13]:
```
# Plot normalized confusion matrix (ratio)
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=classes,
                      title='Model 1 Confusion matrix - Fraction',normalize=True)
```

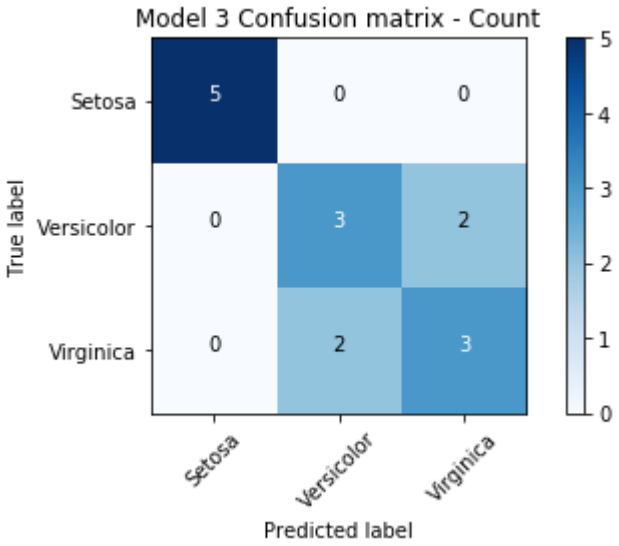


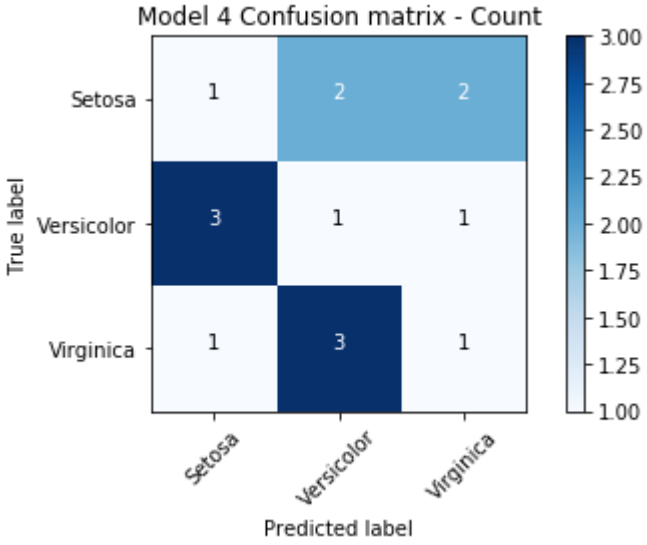
In [14]:

```
# Plot confusion matrix
# Show actual counts
for model in models:
    #print(model)
    cnf_matrix = confusion_matrix(df['NumericClass'],
                                  df[model.replace(' ','') + '_Prediction'],
                                  labels=labels)
    np.set_printoptions(precision=2)

    # Plot non-normalized confusion matrix
    plt.figure()
    plot_confusion_matrix(cnf_matrix, classes=classes,
                      title= model + ' Confusion matrix - Count', normalize=False
```



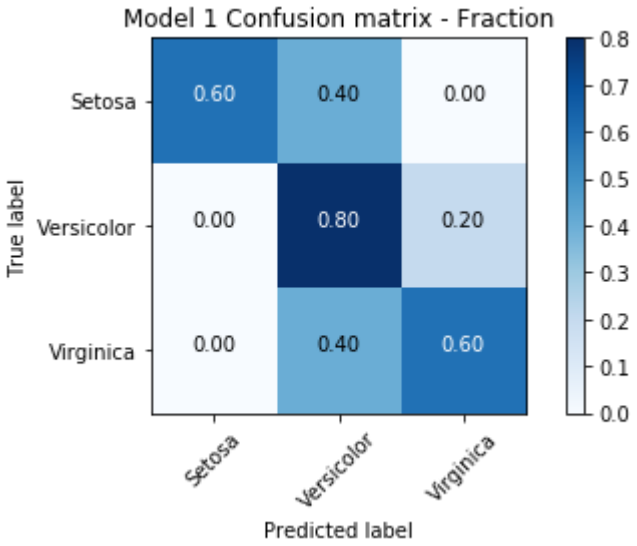


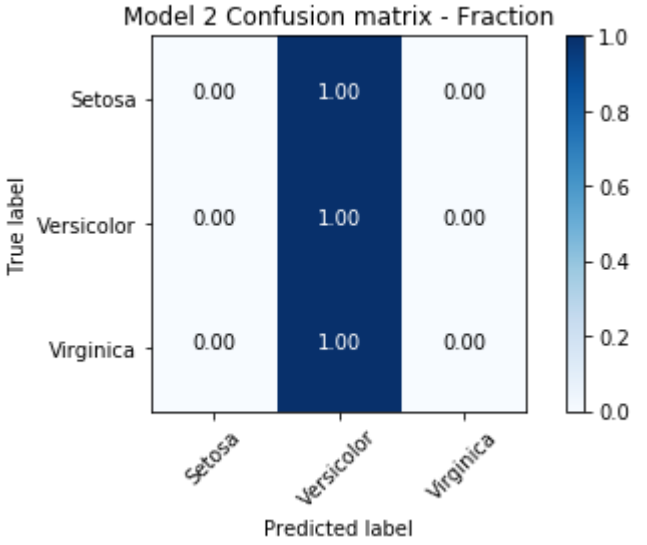

Model 3 Confusion matrix - Count
Setosa
Versicolor
Virginica
True label
5
0
0
0
3
2
0
2
3
Setosa
Versicolor
Virginica
Predicted label
5
4
3
2
1
0

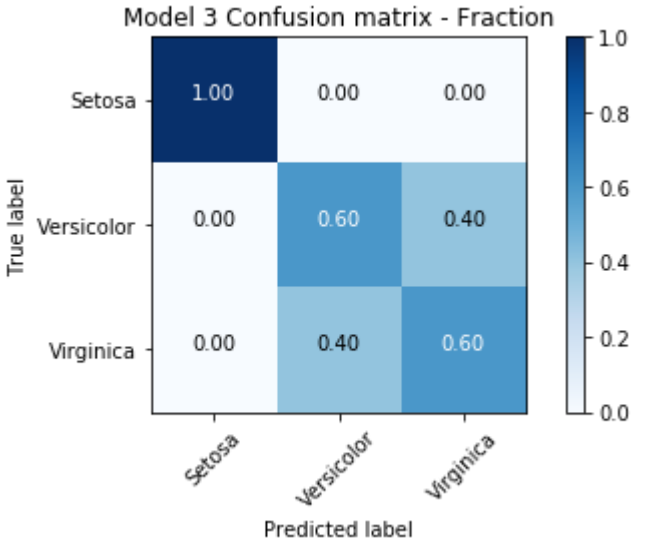

Model 4 Confusion matrix - Count
Setosa
Versicolor
Virginica
True label
1
2
2
3
1
1
1
3
1
Setosa
Versicolor
Virginica
Predicted label
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00

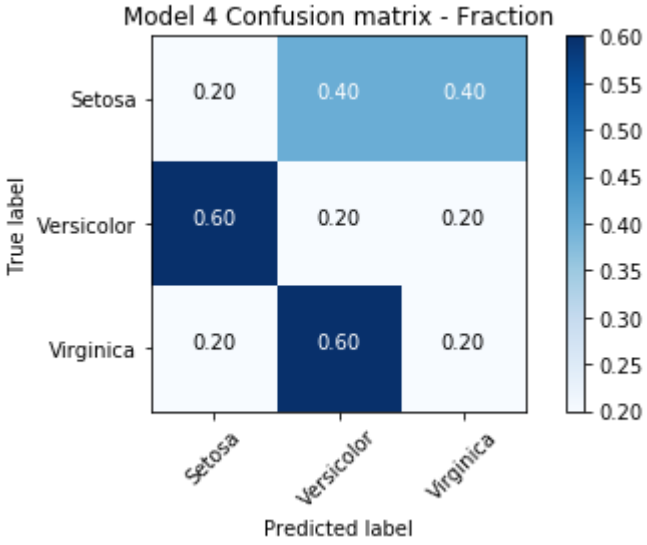
In [15]:

```
# Compute confusion matrix
# Show Ratio
for model in models:
    #print(model)
    cnf_matrix = confusion_matrix(df['NumericClass'],
                                  df[model.replace(' ','') + '_Prediction'],label
    np.set_printoptions(precision=2)

    # Plot non-normalized confusion matrix
    plt.figure()
    plot_confusion_matrix(cnf_matrix, classes=classes,
                          title= model + ' Confusion matrix - Fraction', normalize=Tr
```




Model 2 Confusion matrix - Fraction
Setosa
Versicolor
Virginica
True label
0.00
1.00
0.00
0.00
1.00
0.00
0.00
1.00
0.00
1.0
0.8
0.6
0.4
0.2
0.0
Setosa
Versicolor
Virginica
Predicted label


Model 3 Confusion matrix - Fraction
Setosa
Versicolor
Virginica
True label
1.00
0.00
0.00
0.00
0.60
0.40
0.00
0.40
0.60
1.0
0.8
0.6
0.4
0.2
0.0
Setosa
Versicolor
Virginica
Predicted label


Model 4 Confusion matrix - Fraction
Setosa
Versicolor
Virginica
True label
0.20
0.40
0.40
0.60
0.20
0.20
0.20
0.60
0.20
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
Setosa
Versicolor
Virginica
Predicted label

In [16]:
```
# Using SKLearn classification report
# Micro vs Macro Average
# Reference:
#  https://datascience.stackexchange.com/questions/15989/micro-average-vs-macro-a
for model in models:
    print(model)
    print(classification_report(
    df['NumericClass'],
    df[model.replace(' ','') + '_Prediction'],
    labels=labels,
    target_names=classes))
```

```
Model 1
             precision    recall  f1-score   support

     Setosa       1.00      0.60      0.75         5
 Versicolor       0.50      0.80      0.62         5
  Virginica       0.75      0.60      0.67         5

avg / total       0.75      0.67      0.68        15

Model 2
             precision    recall  f1-score   support

     Setosa       0.00      0.00      0.00         5
 Versicolor       0.33      1.00      0.50         5
  Virginica       0.00      0.00      0.00         5

avg / total       0.11      0.33      0.17        15

Model 3
             precision    recall  f1-score   support

     Setosa       1.00      1.00      1.00         5
 Versicolor       0.60      0.60      0.60         5
  Virginica       0.60      0.60      0.60         5

avg / total       0.73      0.73      0.73        15

Model 4
             precision    recall  f1-score   support

     Setosa       0.20      0.20      0.20         5
 Versicolor       0.17      0.20      0.18         5
  Virginica       0.25      0.20      0.22         5

avg / total       0.21      0.20      0.20        15


C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:11
35: UndefinedMetricWarning: Precision and F-score are ill-defined and being set
to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

```
Summary
```

Macro average = Treat all classes equally. Average of individual class scores

Weighted average = Take frequency of the classes into consideration.

Weighted Average is recommended if there is uneven class distribution

In this example, Weighted Average of Model 1 and Model 3 are highest.

Reference:
https://datascience.stackexchange.com/questions/15989/micro-average-vs-macro-average-performance-in-a-multiclass-classification-settin

https://docs.aws.amazon.com/machine-learning/latest/dg/multiclass-classification.html