Assignment 3 Part 2

CS4172 Machine Learning Lab

Name: Abhiroop Mukherjee

Enrolment Number: 510519109

Task 6

Download the Forest Cover Type dataset (https://www.kaggle.com/uciml/forest-cover-type-dataset) and pre-process the dummy variables to create training, test, and development set. Reduce the train data size if the system unable to process the whole dataset.

In [ ]:

```python
import pandas as pd

_FILE_PATH = './../ML_DRIVE/Assign_3/covtype/covtype.csv'

cov_df = pd.read_csv(_FILE_PATH)

cov_df
```

Out[ ]:

	Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrology	Horizontal_Distance_To_Roadways	Hillshade_9am	Hillshade_Noo
0	2596	51	3	258	0	510	221	23
1	2590	56	2	212	-6	390	220	23
2	2804	139	9	268	65	3180	234	23
3	2785	155	18	242	118	3090	238	23
4	2595	45	2	153	-1	391	220	23
...	...	...	...	...	...	...	...	.
581007	2396	153	20	85	17	108	240	23
581008	2391	152	19	67	12	95	240	23
581009	2386	159	17	60	7	90	236	24
581010	2384	170	15	60	5	90	230	24
581011	2383	165	13	60	4	67	231	24

581012 rows × 55 columns

In [ ]:

```
cov_df.columns
```

Out[ ]:

```
Index(['Elevation', 'Aspect', 'Slope', 'Horizontal_Distance_To_Hydrology',
       'Vertical_Distance_To_Hydrology', 'Horizontal_Distance_To_Roadways',
       'Hillshade_9am', 'Hillshade_Noon', 'Hillshade_3pm',
       'Horizontal_Distance_To_Fire_Points', 'Wilderness_Area1',
       'Wilderness_Area2', 'Wilderness_Area3', 'Wilderness_Area4',
       'Soil_Type1', 'Soil_Type2', 'Soil_Type3', 'Soil_Type4', 'Soil_Type5',
       'Soil_Type6', 'Soil_Type7', 'Soil_Type8', 'Soil_Type9', 'Soil_Type10',
       'Soil_Type11', 'Soil_Type12', 'Soil_Type13', 'Soil_Type14',
       'Soil_Type15', 'Soil_Type16', 'Soil_Type17', 'Soil_Type18',
       'Soil_Type19', 'Soil_Type20', 'Soil_Type21', 'Soil_Type22',
       'Soil_Type23', 'Soil_Type24', 'Soil_Type25', 'Soil_Type26',
       'Soil_Type27', 'Soil_Type28', 'Soil_Type29', 'Soil_Type30',
       'Soil_Type31', 'Soil_Type32', 'Soil_Type33', 'Soil_Type34',
       'Soil_Type35', 'Soil_Type36', 'Soil_Type37', 'Soil_Type38',
       'Soil_Type39', 'Soil_Type40', 'Cover_Type'],
      dtype='object')
```

In [ ]:

```
from sklearn.preprocessing import StandardScaler

def standardize(df: "pd.DataFrame", col_name: "str") -> "pd.DataFrame":
    scaler = StandardScaler()

    df[[col_name]] = pd.DataFrame(
        data=scaler.fit_transform(df[[col_name]]),
        index=df.index,
        columns=[col_name]
    )
    return df
```

In [ ]:

```
_columns_to_scale = ['Elevation', 'Aspect', 'Slope', 'Horizontal_Distance_To_Hydrology',
                     'Vertical_Distance_To_Hydrology', 'Horizontal_Distance_To_Roadways',
                     'Hillshade_9am', 'Hillshade_Noon', 'Hillshade_3pm',
                     'Horizontal_Distance_To_Fire_Points']

for _col in _columns_to_scale:
    cov_df = standardize(cov_df, _col)

cov_df
```

Out[ ]:

	Elevation	Aspect	Slope	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrology	Horizontal_Distance_To_Roadways	Hillshade_9am	Hillshad
0	-1.297805	-0.935157	-1.482820	-0.053767	-0.796273	-1.180146	0.330743	0
1	-1.319235	-0.890480	-1.616363	-0.270188	-0.899197	-1.257106	0.293388	0
2	-0.554907	-0.148836	-0.681563	-0.006719	0.318742	0.532212	0.816364	0
3	-0.622768	-0.005869	0.520322	-0.129044	1.227908	0.474492	0.965786	0
4	-1.301377	-0.988770	-1.616363	-0.547771	-0.813427	-1.256464	0.293388	0
...	...	...	...	...	...	...	...	
581007	-2.012130	-0.023740	0.787408	-0.867697	-0.504653	-1.437962	1.040496	0
581008	-2.029988	-0.032675	0.653865	-0.952383	-0.590424	-1.446299	1.040496	0
581009	-2.047847	0.029873	0.386780	-0.985317	-0.676194	-1.449506	0.891075	0
581010	-2.054990	0.128163	0.119694	-0.985317	-0.710502	-1.449506	0.666942	1
581011	-2.058562	0.083486	-0.147392	-0.985317	-0.727656	-1.464256	0.704298	1

581012 rows × 55 columns

In [ ]:

```
cov_df[['Cover_Type']].value_counts()
```

Out[ ]:
```
Cover_Type
2             283301
1             211840
3              35754
7              20510
6              17367
5               9493
4               2747
dtype: int64
```

In [ ]:
```python
# NOTE: class imbalance is present but removing it will
# remove the data that cover_type 2 is the most common data in world

cov_df = cov_df.sample(frac=0.1)

X = cov_df.drop('Cover_Type', axis=1)
y = cov_df[['Cover_Type']]
```

In [ ]:
```python
y.value_counts()
```

Out[ ]:
```
Cover_Type
2             28395
1             20996
3              3653
7              2064
6              1766
5               958
4               269
dtype: int64
```

In [ ]:
```python
# 80% as train
# 10% as validation
# 10% as train

from sklearn.model_selection import train_test_split

X_train, _X_rest, y_train, _y_rest = train_test_split(X, y, train_size=0.8)
X_val, X_val, y_val, y_val = train_test_split(_X_rest, _y_rest, train_size=0.5)
```

Task 7

Train the one vs rest and one-vs-one SVM model on the above dataset for multiclass classification. Plot and Analyze the Confusion matrix for the above models. Show the accuracy in the graph. State the difference of the two approaches using the model parameters.

In [ ]:

```
# hyper parameter tuning

from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

def display_confusion_matrix(X_test: "pd.DataFrame",
                             y_test: "pd.DataFrame",
                             model: "SVC"):
    y_predict = model.predict(X_test)
    matrix = confusion_matrix(y_test, y_predict)
    fig = plt.figure(figsize=(10,10))
    sns.heatmap(
        matrix,
        xticklabels=range(1,8),
        yticklabels=range(1,8),
        linewidth=0.5,
        cmap='coolwarm',
        annot=True,
        cbar=True,
        square=True)
    plt.title('HeatMap for the model')
    plt.ylabel('Actual Value')
    plt.xlabel('Predicted Value')
    plt.show()
```

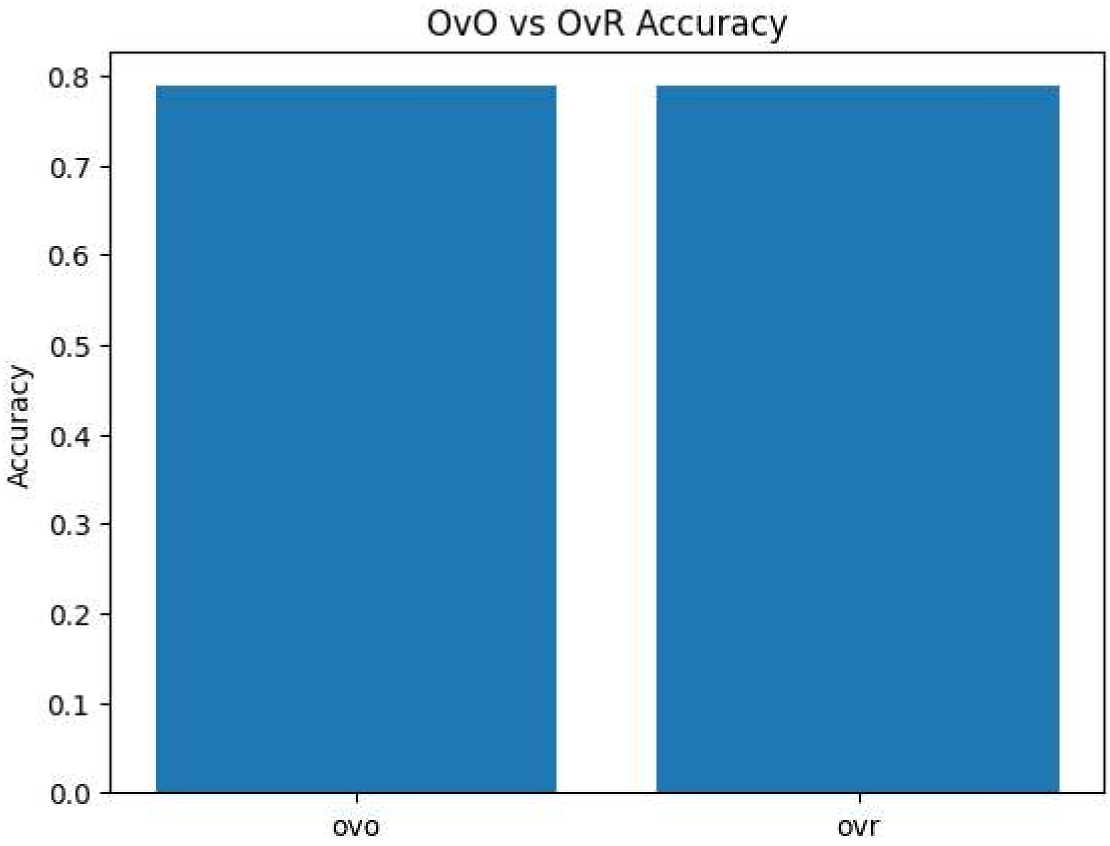
In [ ]:

```
decision_function_shapes = ['ovo', 'ovr']

models = [
    SVC(decision_function_shape=shape).fit(X_train, y_train.iloc[:, 0])
    for shape in decision_function_shapes
]
```

In [ ]:

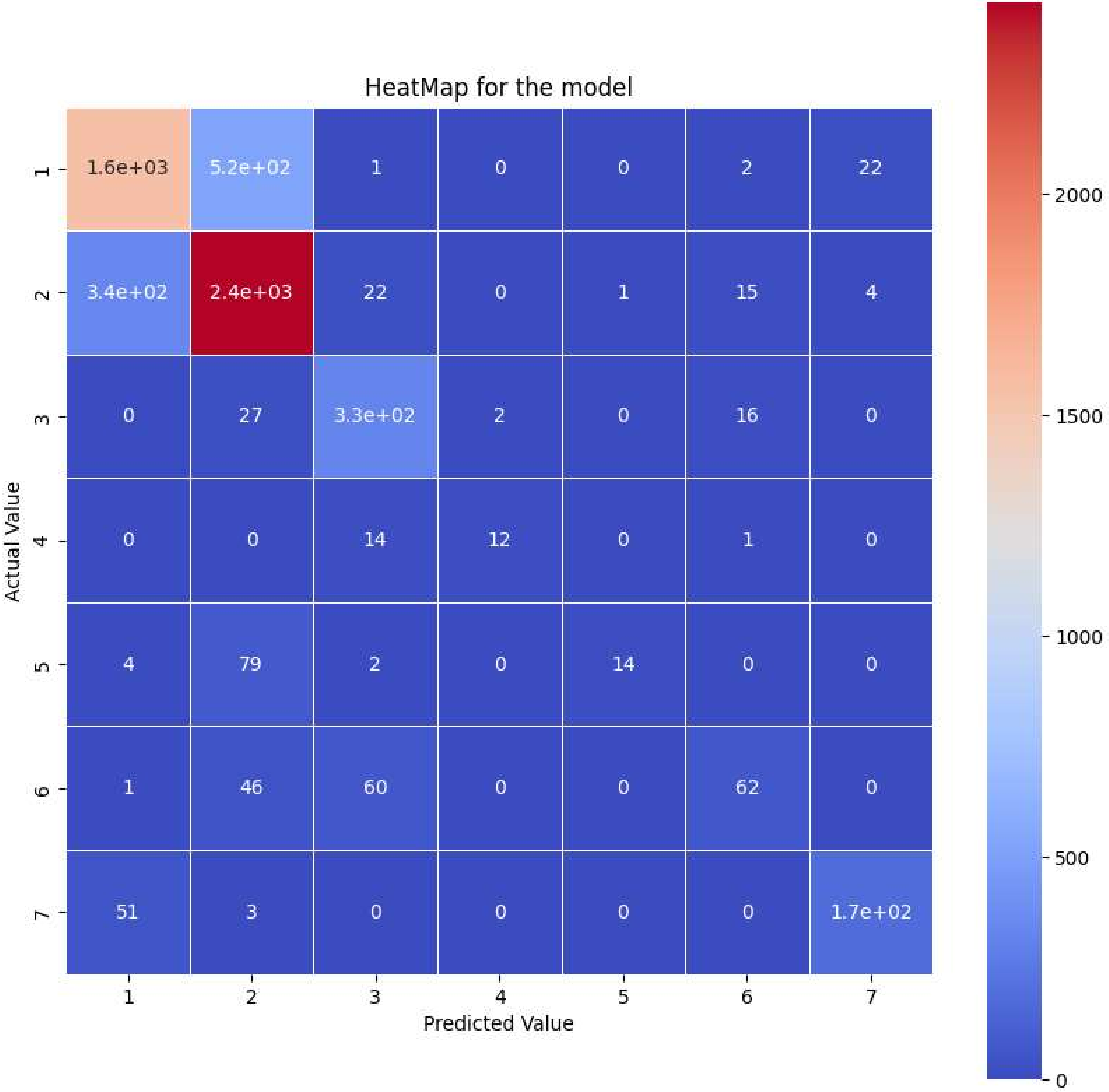
```
# Accuracies

accuracies = [model.score(X_val, y_val) for model in models]


print(pd.DataFrame(columns=['decision_function_shape', 'Accuracy'],
             data=zip(decision_function_shapes, accuracies)))

plt.bar(range(0, len(accuracies)), accuracies)
plt.title('OvO vs OvR Accuracy')
plt.ylabel('Accuracy')
plt.xticks(ticks=[0,1], labels=['ovo', 'ovr'])
plt.show()
```

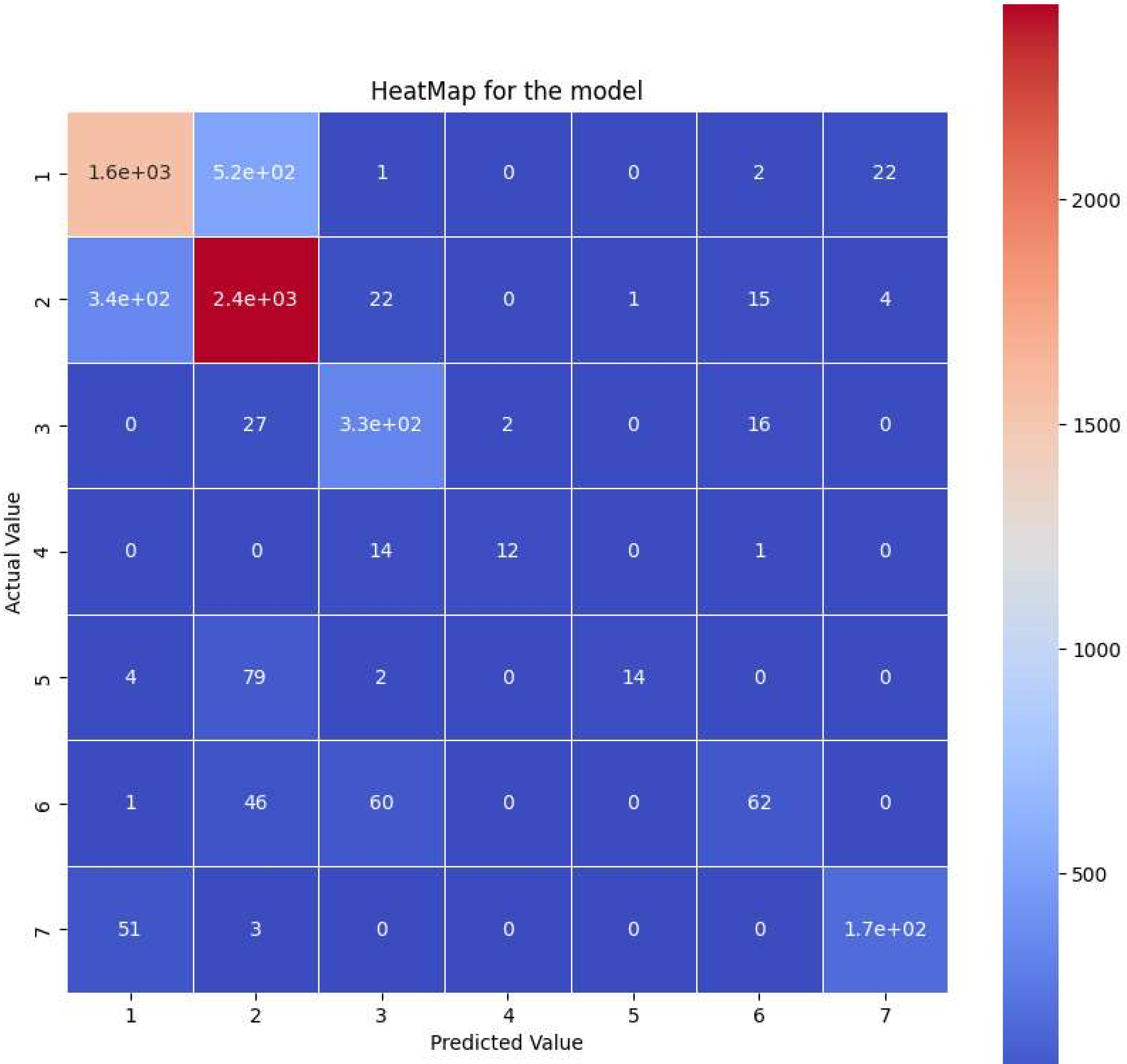
```
  decision_function_shape  Accuracy
0                     ovo   0.78816
1                     ovr   0.78816
```



In [ ]:

```
for model in models:
    display_confusion_matrix(X_val, y_val, model)
```


HeatMap for the model
Actual Value
Predicted Value
1
2
3
4
5
6
7
1.6e+03
5.2e+02
1
0
0
2
22
3.4e+02
2.4e+03
22
0
1
15
4
0
27
3.3e+02
2
0
16
0
0
0
14
12
0
1
0
4
79
2
0
14
0
0
1
46
60
0
0
62
0
51
3
0
0
0
0
1.7e+02
2000
1500
1000
500
0


HeatMap for the model
Actual Value
Predicted Value
1
2
3
4
5
6
7
1.6e+03
5.2e+02
1
0
0
2
22
3.4e+02
2.4e+03
22
0
1
15
4
0
27
3.3e+02
2
0
16
0
0
0
14
12
0
1
0
4
79
2
0
14
0
0
1
46
60
0
0
62
0
51
3
0
0
0
0
1.7e+02
2000
1500
1000
500
0