University of Science and Technology of Hanoi

Classification I Report

Machine learning and Data mining II

Nguyễn Mạnh Hưng - 22BI13183

Nguyễn Trọng Minh - 22BI13304

TABLE OF CONTENT

I. K-nearest neighbor classification

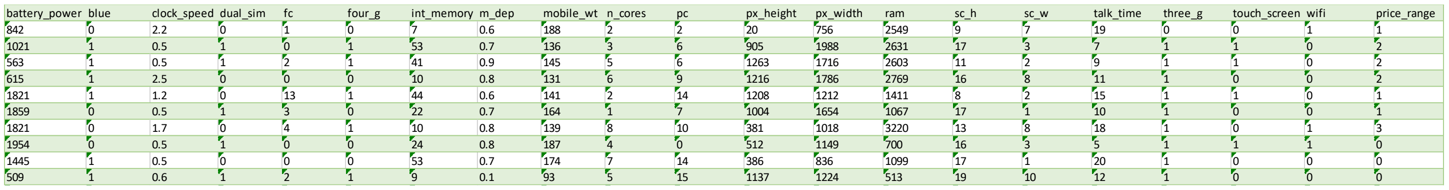
1. Perform K-NN

In this lab work, we chose 2 data sets both are binary classified. One contains data about 100 mushroom individuals and classified into edible or not.

cap-diameter	cap-shape	gill-attachment	gill-color	stem-height	stem-width	stem-color	season	class
1372	2	2	10	3.807466754	1545	11	1.804272709	1
1461	2	2	10	3.807466754	1557	11	1.804272709	1
1371	2	2	10	3.612496295	1566	11	1.804272709	1
1261	6	2	10	3.78757181	1566	11	1.804272709	1
1305	6	2	10	3.711971019	1464	11	0.943194554	1
1337	6	2	10	3.775634843	1520	11	0.943194554	1
1300	2	2	10	3.835319677	1563	11	1.804272709	1
1354	6	2	10	3.676160118	1532	11	0.888450288	1
1222	6	2	10	3.771655854	1476	11	0.943194554	1
1085	6	2	10	3.775634843	1581	11	0.888450288	1

Data about mushroom

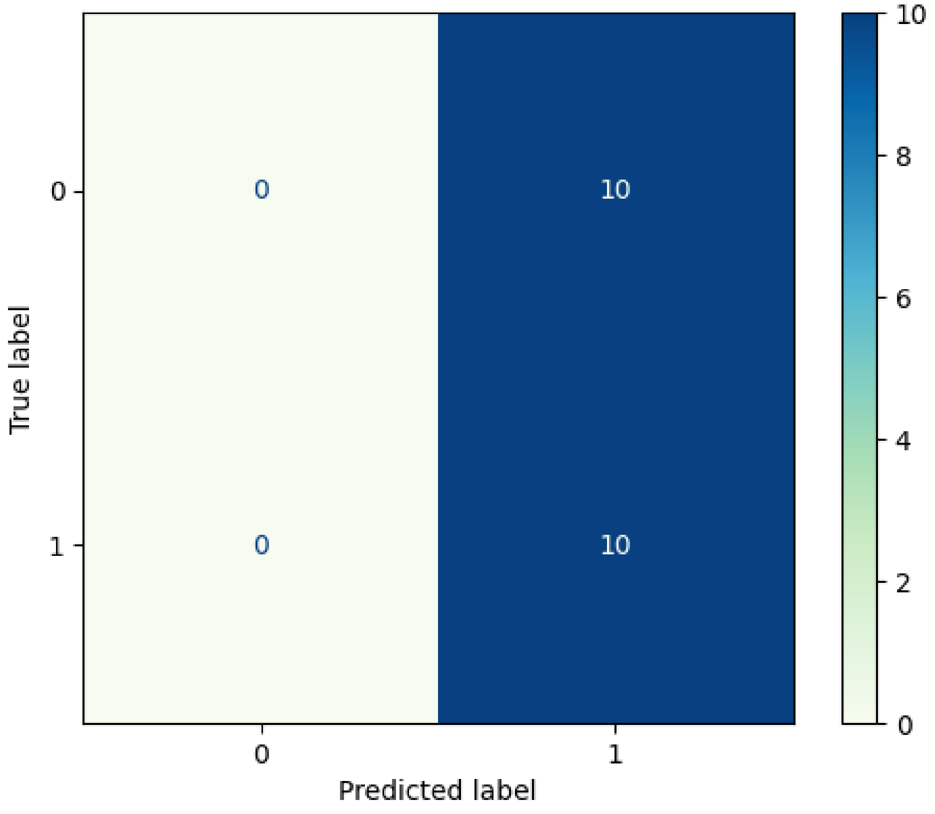
The other one is about 100 phones with their spec, divided into 4 price range.



battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
842	0	2.2	0	1	0	7	0.6	188	2	2	20	756	2549	9	7	19	0	0	1	1
1021	1	0.5	1	0	1	53	0.7	136	3	6	905	1988	2631	17	3	7	1	1	0	2
563	1	0.5	1	2	1	41	0.9	145	5	6	1263	1716	2603	11	2	9	1	1	0	2
615	1	2.5	0	0	0	10	0.8	131	6	9	1216	1786	2769	16	8	11	1	0	0	2
1821	1	1.2	0	13	1	44	0.6	141	2	14	1208	1212	1411	8	2	15	1	1	0	1
1859	0	0.5	1	3	0	22	0.7	164	1	7	1004	1654	1067	17	1	10	1	0	0	1
1821	0	1.7	0	4	1	10	0.8	139	8	10	381	1018	3220	13	8	18	1	0	1	3
1954	0	0.5	1	0	0	24	0.8	187	4	0	512	1149	700	16	3	5	1	1	1	0
1445	1	0.5	0	0	0	53	0.7	174	7	14	386	836	1099	17	1	20	1	0	0	0
509	1	0.6	1	2	1	9	0.1	93	5	15	1137	1224	513	19	10	12	1	0	0	0

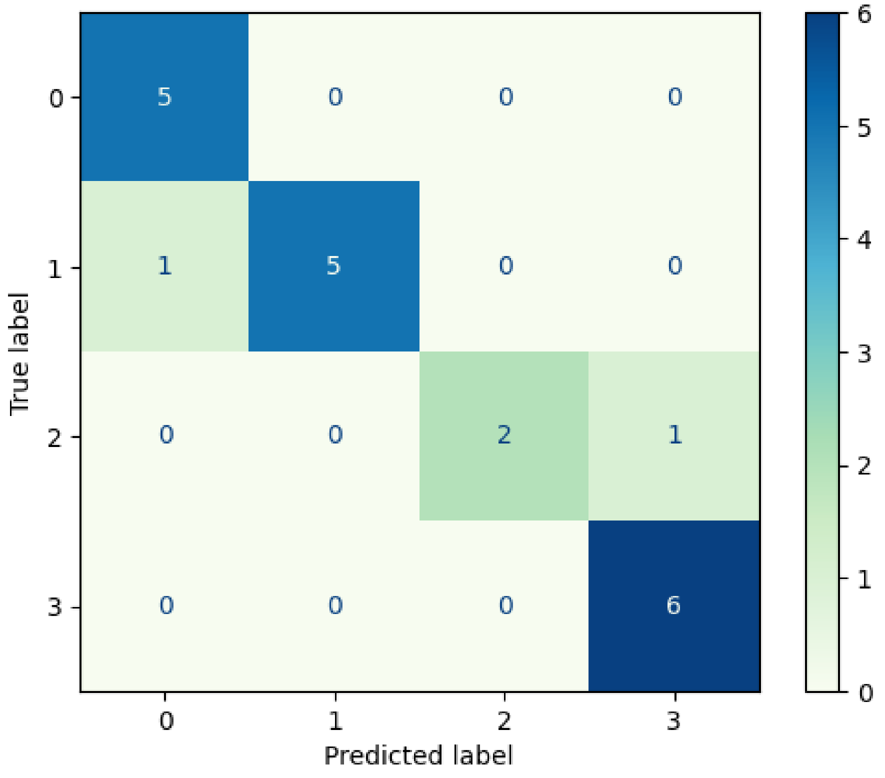
Data about phone price

After a quick look into the data set, we can start applying K-nearest neighbors classification algorithm. The algorithm is quite simple in the idea, using the distance between each point to classify a point to the same group of the nearest data point. And we got this confusion matrix for the mushroom data with a small testing data set of 20 data points separated from training data:



Mushroom dataset's testing result

As we can see that the predict result only get to 50% accuracy, since we haven't normalized the data yet. On the other hand, the phone data gave a 90% accuracy prediction.



Phone dataset's confusion matrix

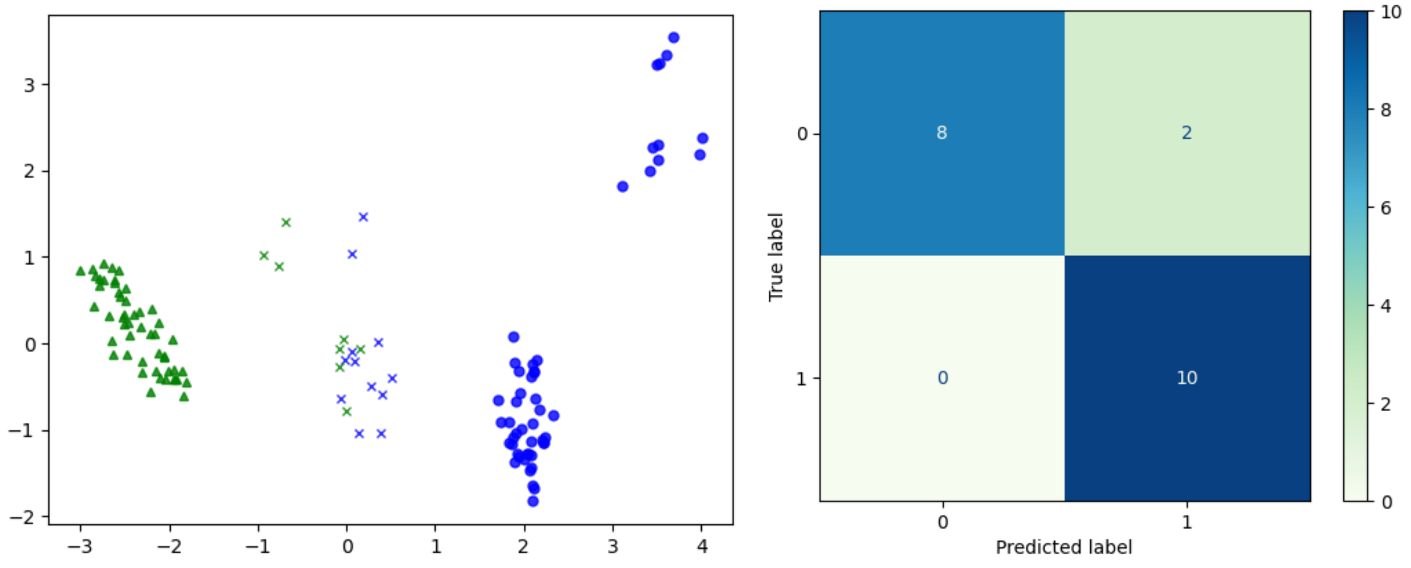
We might try to vary k from 5 up to 20 to see if there is any different. With the mushroom data, there is totally no different with all the k value in the given range. On the other hand, the prediction with phone data varies slightly with k value. At k=10, the accuracy is down a little bit with 1 more misclassify, but when k=15 and above, the accuracy is back to normal at 90%.

2. Normalize Data

Since K-NN measure the distance between each data points, it's recommended to normalize the data before calculation. And we will do it in this section then recalculate K-NN performance after that.

To normalize dataset, we simply divide each column by its standard deviation and subtract by mean to center it for later PCA.

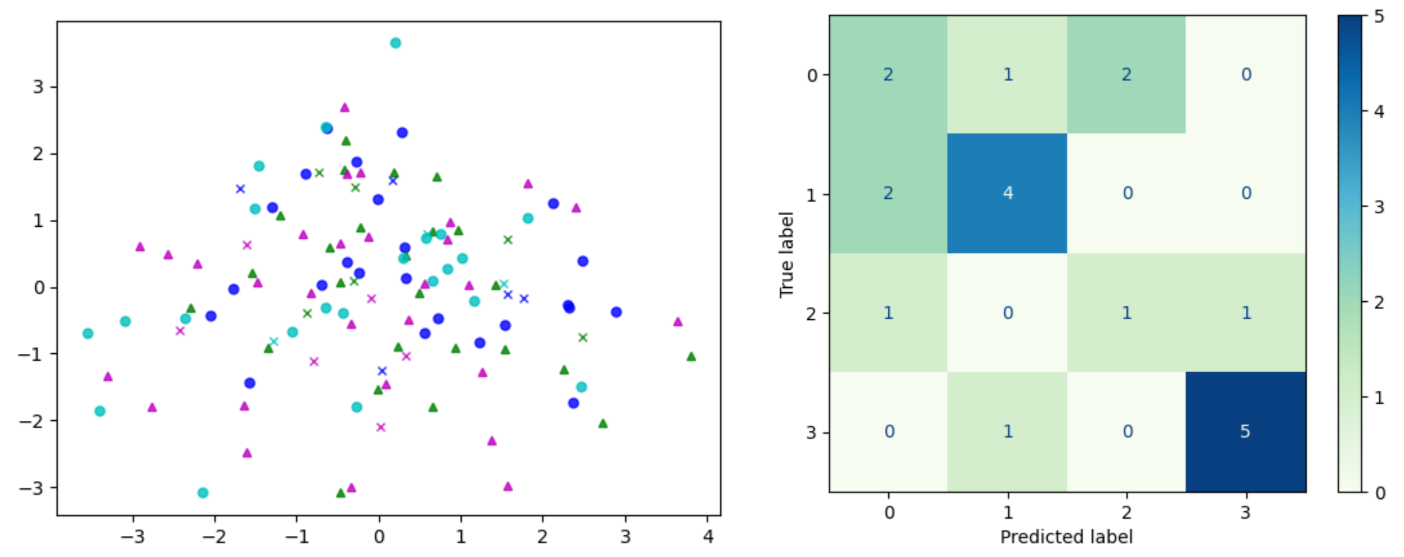
After that, we can perform K-NN on the dataset again and see the result.



Plotted result and confusion matrix of mushroom data

As we can see from the plotted result, x is the test data, there are some points that are closer to blue dot but classified into green category. This is because data lost in PCA. We also got a huge performance improvement, from 50% to 90% accuracy with test data.

In contrary, the phone data experienced a reduction in accuracy, from 90% to 60%. This is because the normalized data is scattered and became very hard to classify.

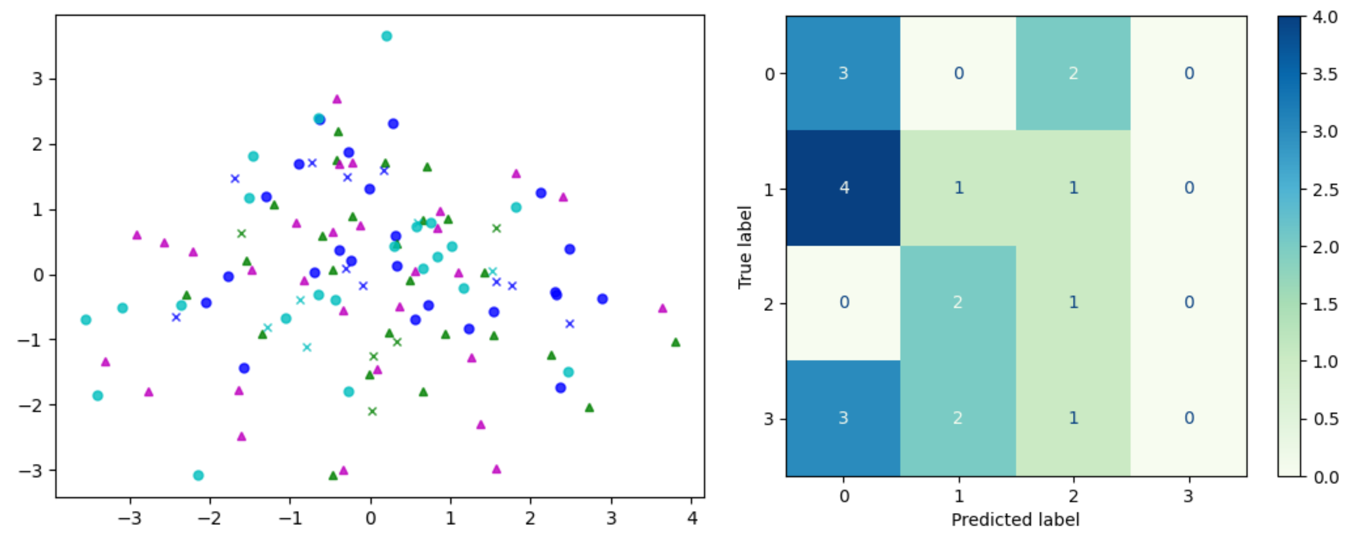


Plotted result and confusion matrix of phone data

3. Apply PCA

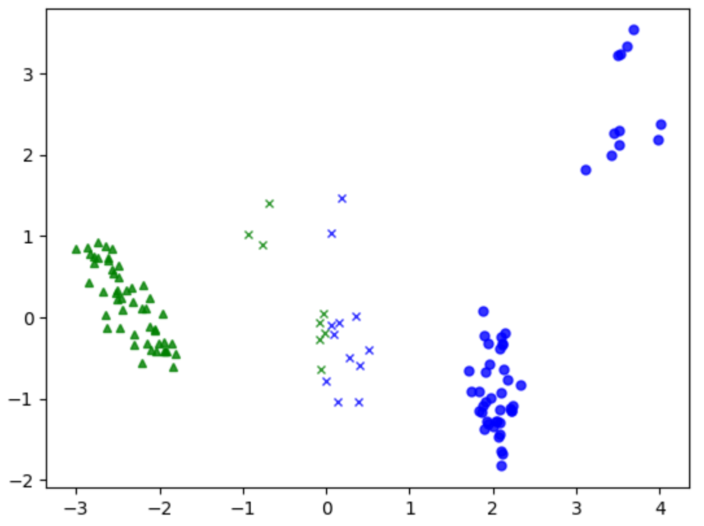
Now we try to apply K-NN on a 2D dataset transformed by PCA and see the result.

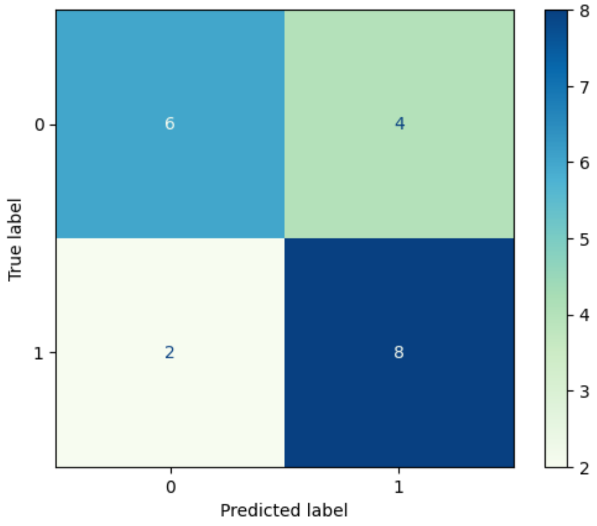
For the phone data, there is a steep fall in accuracy since it's now only 25%.



Plotted result and confusion matrix of phone data after PCA

The mushroom dataset also experienced an accuracy drop, from 90% to 70%. This fall in accuracy is not as big as phone data because this data is linearly separable, not like the phone data. We can also notice that the plotted data doesn't contain any outliners.





Plotted result and confusion matrix of mushroom data after PCA

4. Improvement

Since we have 100 data points in each data set, we can apply a 5-Fold cross validation with each test data set contains 20 points. We can train our model 4 times to get a better performance.

II. Support Vector Machine

1. Analyze the data

For the second part, we chose the same datasets as the previous part but more objects. For example, there will be over 5000 mushroom individuals and 2000 phones.

In order to define whether the dataset is linear separable or non-linear separable. We use some linear regression model to train and calculate the accuracy of the model with the given data, if the accuracy is approximately to 1 so the data is linear separable or else if it is smaller than 1 then the data is non-linear separable

Regarding the mushroom dataset, we use two models to classify the data which are Linear Regression and Linear Support Vector Machine. After training the data we have the accuracy of 0.81 and 0.71 respectively to Linear regression model and Linear Support vector machine model, which mean that the data cannot be separated linearly. In conclusion, the mushroom data is non-linear separable.

Regarding the phone dataset, we also use Linear Regression model as the mushroom dataset. After the training, the accuracy that we get is 0.97, which is very high and close to 1. In a nutshell, the data is linear separable.

2.Set up the SVM

To set up the suitable SVM parameters for the dataset, we will create a dictionary name "parameters" including the key such as "kernel", "C", "degree", "coef0", and "gamma". Next, for each key, we assign a list or a tuple of values to the keys. For example, the "kernel" assigned a tuple of kernel types ("linear", "poly", "rbf", "sigmoid") and the key "C" assigned a list ('1', '52', '10') and so on.

*<u>**Mushroom dataset**</u>.

By running the tool, we can find that the suitable parameters for the selected dataset are {'C': 52, 'coef0': 0.5, 'degree': 8, 'gamma': 'auto', 'kernel': 'poly'}.

***Phone dataset**

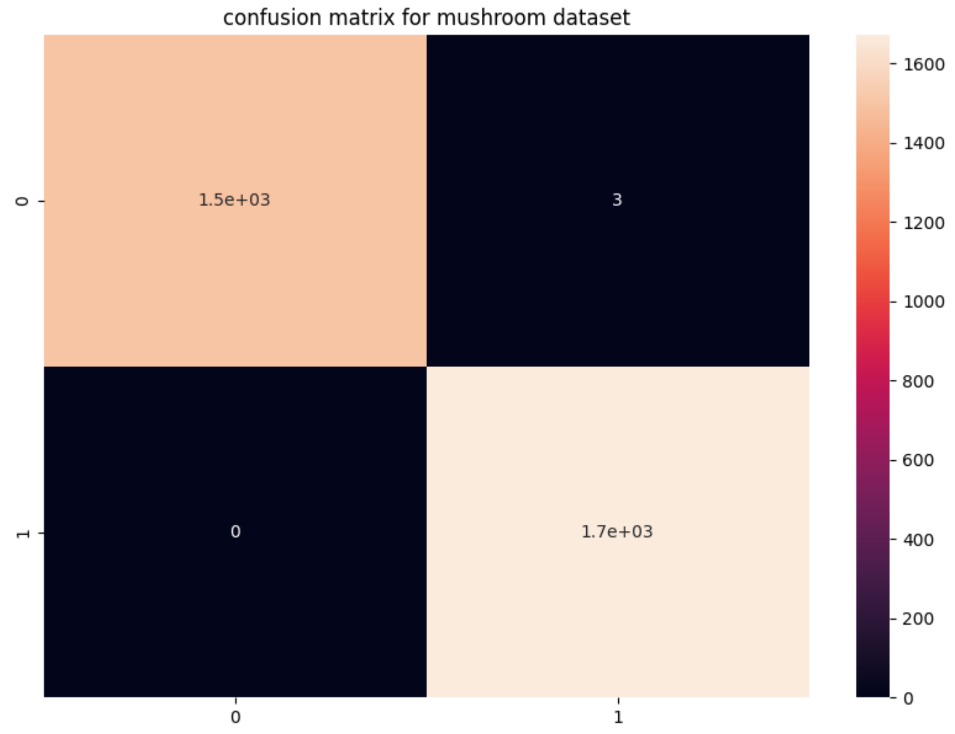
Using the same tool, we can find the suitable parameters for the dataset are {'C': 52, 'coef0': 0.001, 'degree': 3, 'gamma': 'auto', 'kernel': 'linear'}.

3.SVM's performance.

By using the above parameters that we found in the previous part, we start to train the SVM model with the selected datasets.

*Mushroom dataset.

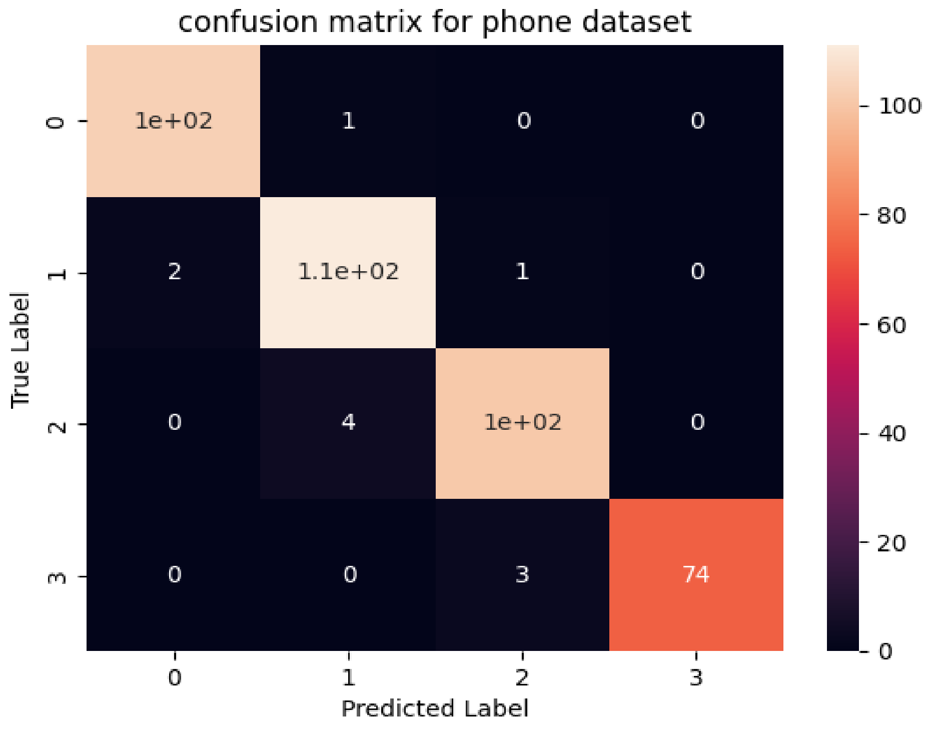
Firstly, we split the dataset into training and testing set using "train_test_split" . After that we create an SVM model with the parameters that suitable for the dataset above. The next step is to train the model with the training set that has just been created. We can plot the heatmap of the confusion matrix for the prediction of the model with the testing set below:



With the confusion matrix above, we can calculate the accuracy of the model and this have a result of 99.9% which means that the dataset is well split and the model did it perfectly.

*Phone dataset.

With the same steps as the mushroom dataset, we can train the SVM model with the selected dataset and get the confusion matrix as below:



The accuracy that we can compute with the given confusion matrix is 97.25%, which is also very high and gives we the conclusion that the model fit very well on the dataset.

4. Multi-class datasets

There are 2 most common ways for SVM to handle the classification task for multi-class dataset which are “One-vs-Rest” and “One-vs-One”:

With the “One-vs-Rest” ways, with n classes in the dataset, we will train n SVMs. For each classifier, we will treat the current class as positive and the other are negative. During the prediction, we will choose the highest score of the output as the final prediction.

With the “One-vs-One” way, we will create a binary classifier for each pair of classes, which means that we will have (n*(n-1))/2 classifiers. For each classifier, we only train on the data of two classes and ignore the rest. During the prediction, we employ majority voting along with the distance from the margin as a confidence criterion.