GEO5017 A2: Classification

Pratyush Kumar (5359252) ,Cynthia Cai (5625483)

1 Introduction

In this assignment, we implemented SVM and random forest with *scikit-learn* and evaluated its accuracy. We have the following motivations:

Key goals:
(1) Understand the mathematical principle of these two algorithms;
(2) Conduct feature engineering and explore its impact on the accuracy;
(3) Learn to implement and evaluate classification algorithms.

Side goals:
(4) Practise the processing of point cloud data;
(5) Explore the impact of train-test-ratio and classifier's parameters on the accuracy;
(6) Practise *scikit-learn, matplotlib, seaborn,* and *pandas*.

The methodology is:

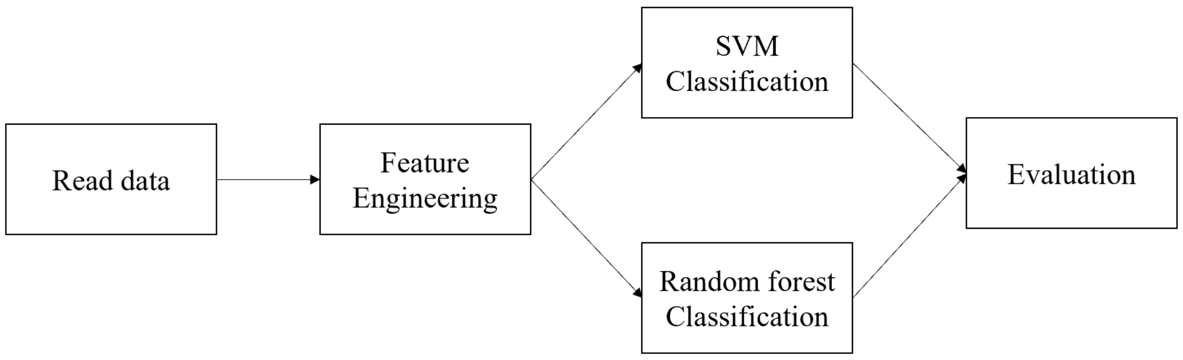


Figure 1. *Overview of the methodology*

The github repository can be found here.

2 Dataset

The dataset contains 500 point clouds of urban objects, which can be classified as building, car, fence, pole and tree.

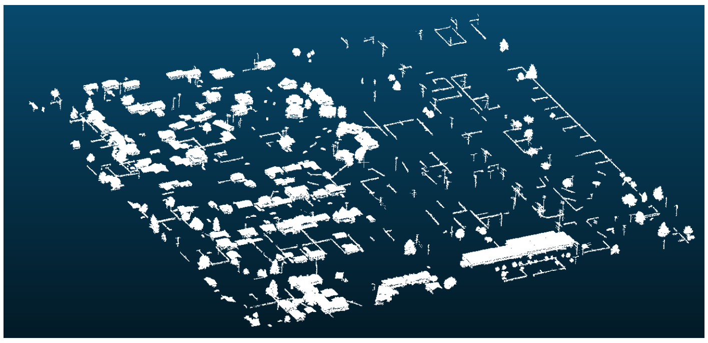
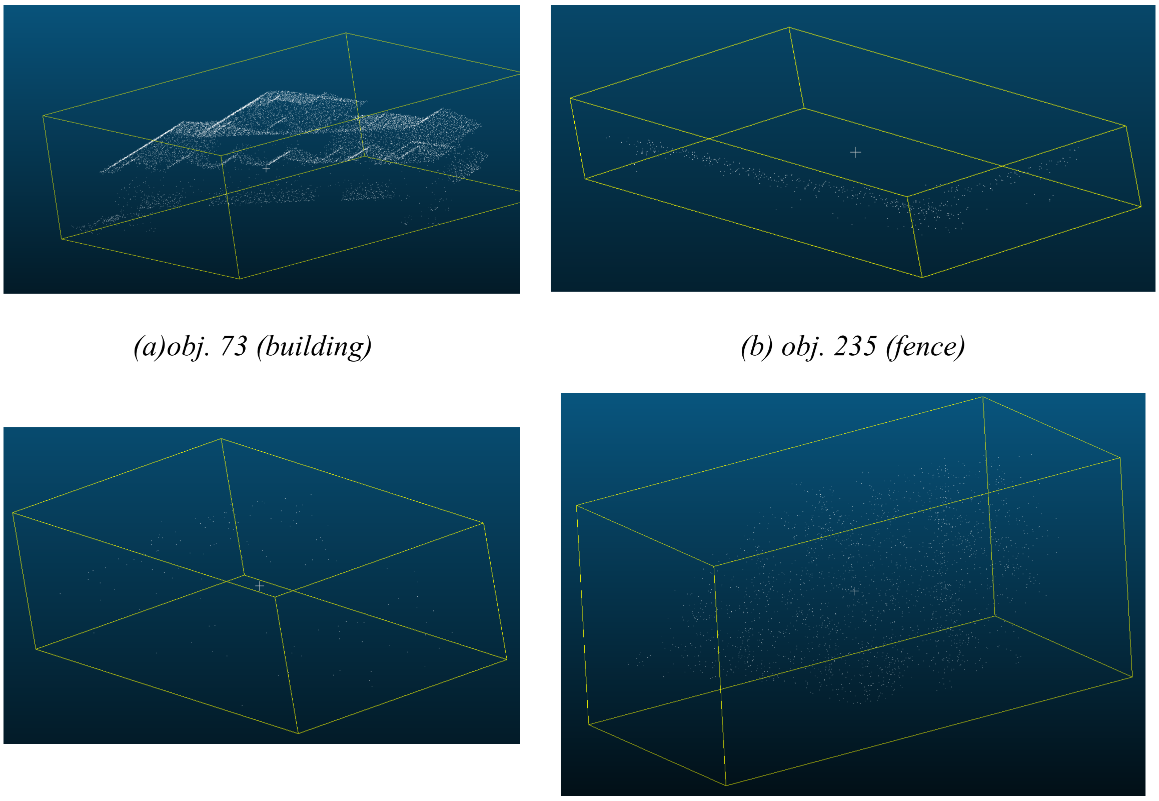


Figure 2.1 *The overview of the point clouds*

Each class has 100 objects and each object contains multiple 3D points with (x,y,z) coordinates.


(a)obj. 73 (building)

(b) obj. 235 (fence)

(c) obj. 137 (car)

(d) obj. 456 (tree)

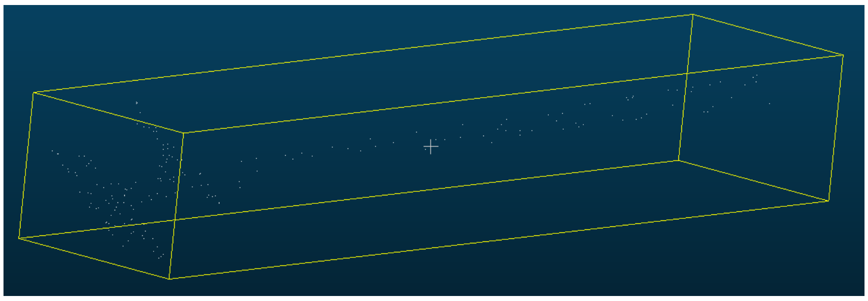

(e) obj. 377 (pole)

Figure 2.2 *Examples of objects in each class*

3 Feature engineering

We designed six normalised features and analysed them with a correlation matrix and scatter plots. In a later experiment in chapter 4, we will explore more on feature combinations.

3.1 Feature design and normalisation

From our observation on the point clouds, we created six features with reasons.

No.	Feature	Reason for creation
1	Range of height	Range of x,y,z was taken into consideration since different objects have different shapes, and apart from the variance, the range also gives us a fair idea of the spread of the points in the object
2	Median of height	Since objects like trees have more density on the upper portion of the object, we thought it would be nice to take the median Z value into account, thus for trees the median Z would be closer to the canopy than trees, this would be lower for cars and fences.
3	volume of convex hull	The convex hull of the object will be similar for similar objects, like trees will be in a typical range, buildings would have higher volume and cars would have a smaller volume.
4	ratio of length and height	It reflects the shape of the object. The five classes have significant different shapes.
5	2D footprint density	Following the volume, we also take the footprint density into account, it is a measure of the spread of the points on the x-y plane
6	3D volume density	It varies depending on the size of the object and number of points it has, given that the LiDAR points are collected uniformly, the density will also prove to be an important metric for classification.

Table 3.1: *Features to be used*

We **normalised** all features using min and max scaling. The formula is:
current_value = (current_value - min_value) / (max_value - min_value)

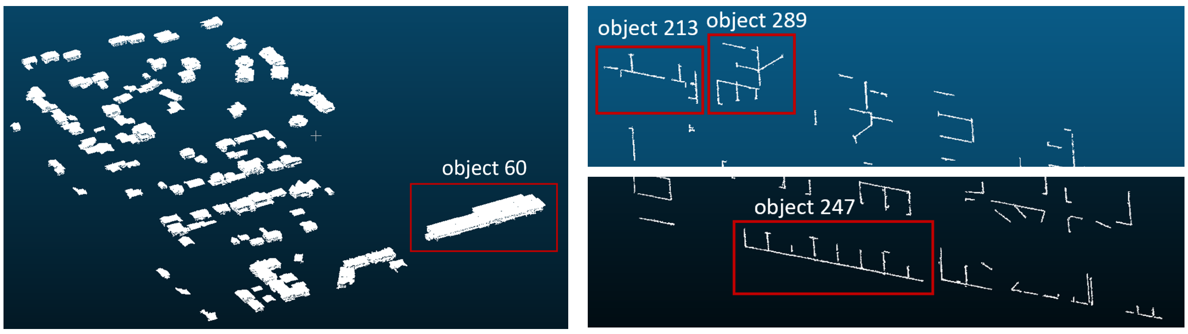
current_value is rolling with each row, and min_value and max_value are the minimum and maximum value of the column. The formula is easily done by using *pandas*' ability to manipulate an entire dataframe at once. Then we obtained this feature table:

	mean_Z	range_Z	volume	ratio	footprintDensity	volumeDensity	label
0	0.716721	0.199121	0.021946	0.027853	0.382691	0.126469	building
1	0.229411	0.136703	0.028988	0.019509	0.329856	0.109521	building
2	0.486472	0.307692	0.264705	0.025458	0.336488	0.049716	building

3	0.526394	0.289231	0.281438	0.019058	0.297612	0.054039	building
4	0.249514	0.207473	0.233507	0.012187	0.237640	0.078642	building

Table 3.2 *Examples of the normalised feature (the first 5 objects)*

3.2 Outlier processing



(a) Large building *(2) Large fence*

Figure 3.1 *Point clouds of large objects*

Particularly, we notice that the existence of extremely large buildings and fences can lead to weird scaling in data, therefore we processed them specifically. In order to find the outliers, we printed the top ten objects with the largest volume in Pandas DataFrame and drew a histogram:

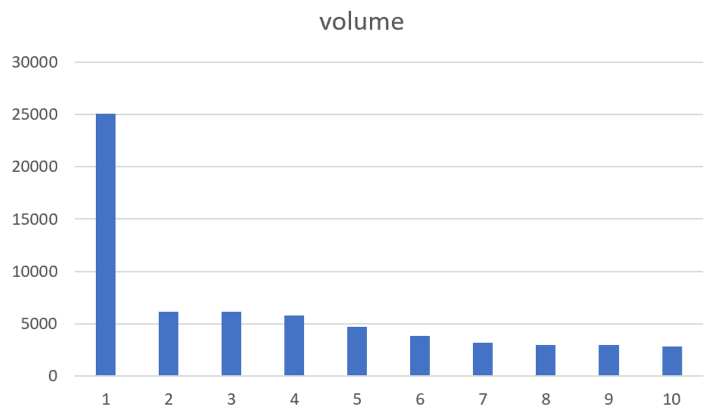
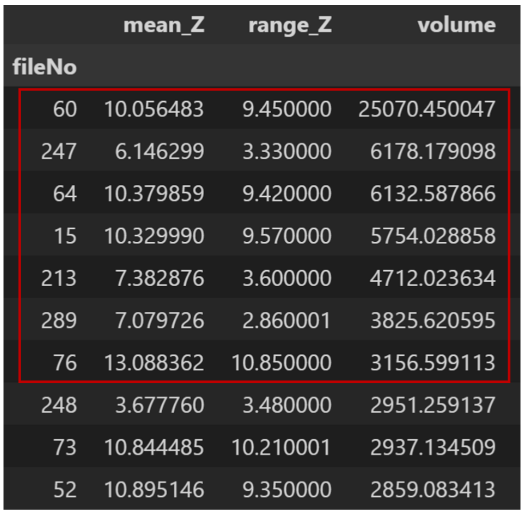


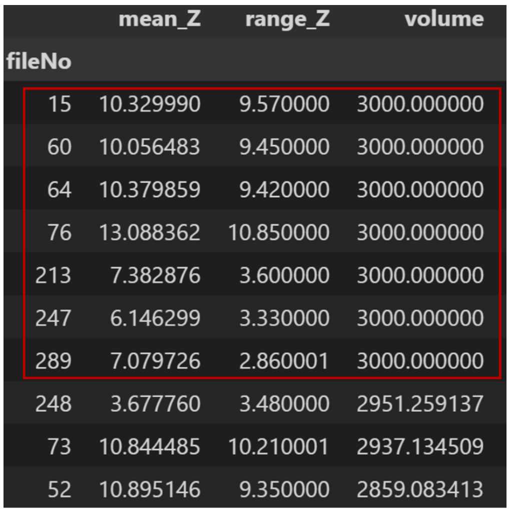
Figure 3.2 *The volume distribution of top 10 objects*

Obj. 60 is apparently a very special case. We first assigned obj. 60 with the volume of obj. 247, but the overall distribution of volume was still bad . Our modification process is shown in Figure 3.3. We eventually processed this: for those whose volume is over 3000, assign it 3000.



fileNo	mean_Z	range_Z	volume
60	10.056483	9.450000	25070.450047
247	6.146299	3.330000	6178.179098
64	10.379859	9.420000	6132.587866
15	10.329990	9.570000	5754.028858
213	7.382876	3.600000	4712.023634
289	7.079726	2.860001	3825.620595
76	13.088362	10.850000	3156.599113
248	3.677760	3.480000	2951.259137
73	10.844485	10.210001	2937.134509
52	10.895146	9.350000	2859.083413

(a) Before modification

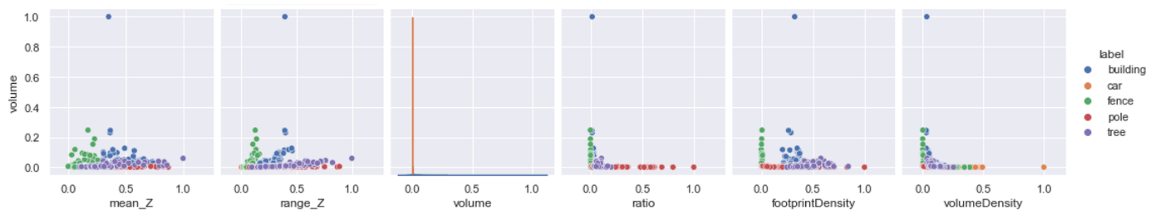


fileNo	mean_Z	range_Z	volume
15	10.329990	9.570000	3000.000000
60	10.056483	9.450000	3000.000000
64	10.379859	9.420000	3000.000000
76	13.088362	10.850000	3000.000000
213	7.382876	3.600000	3000.000000
247	6.146299	3.330000	3000.000000
289	7.079726	2.860001	3000.000000
248	3.677760	3.480000	2951.259137
73	10.844485	10.210001	2937.134509
52	10.895146	9.350000	2859.083413

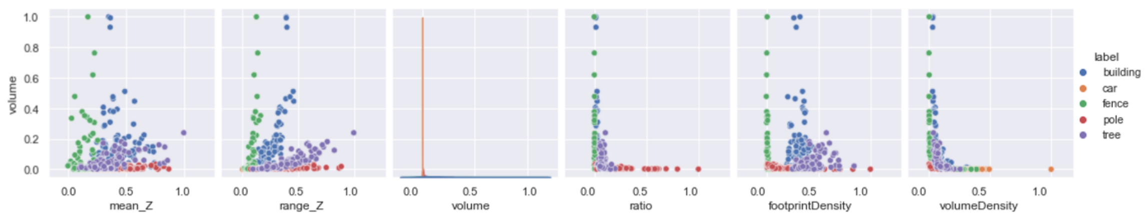
(b) After modification

Figure 3.3 *Top 10 objects of largest volume*

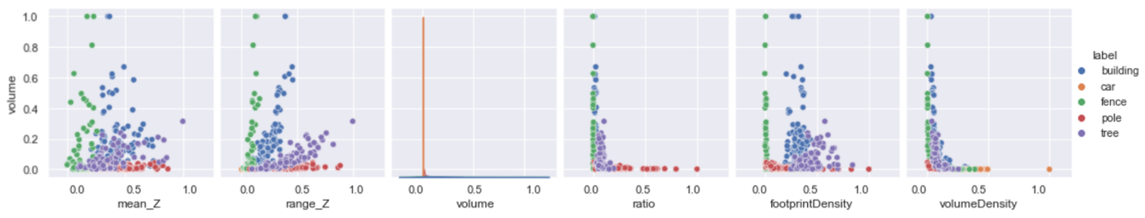
From Figure 3.4, we can see that the distribution was significantly improved.



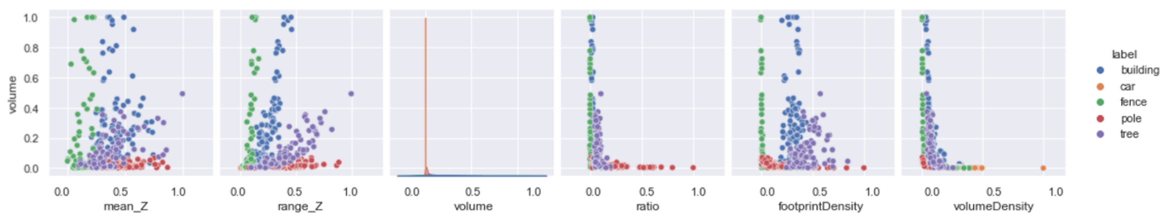
(a) No modification



(b) Modification of obj.60 (with obj. 247's value)



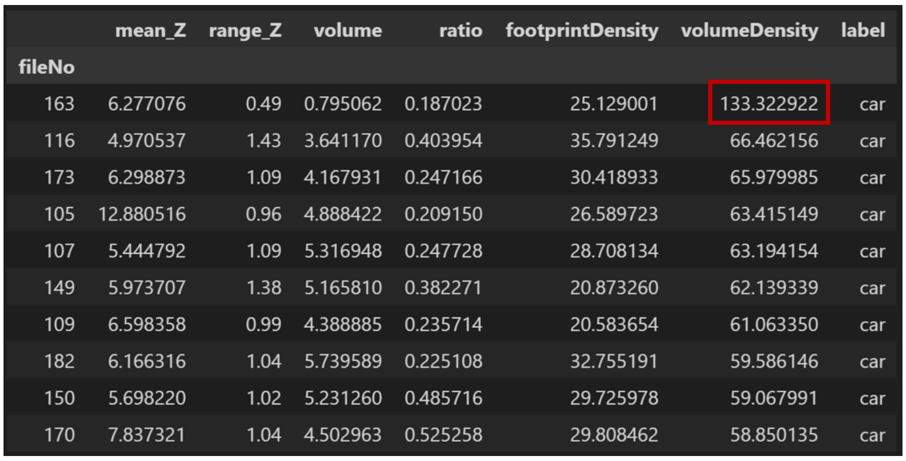
(c) Modification of obj.60 247 64 15 (with obj. 213's value)



(d) Modification of those above 3000 (with 3000)

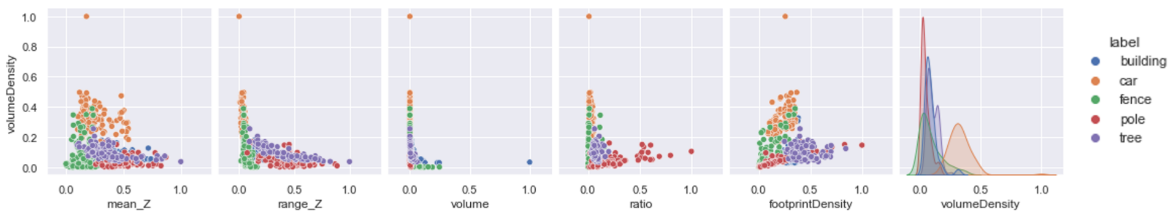
Figure 3.4 *The volume related distribution with different modification*

Similarly, we found an outlier in volumeDensity, and we assigned 67 to obj. 163’s volumeDensity.

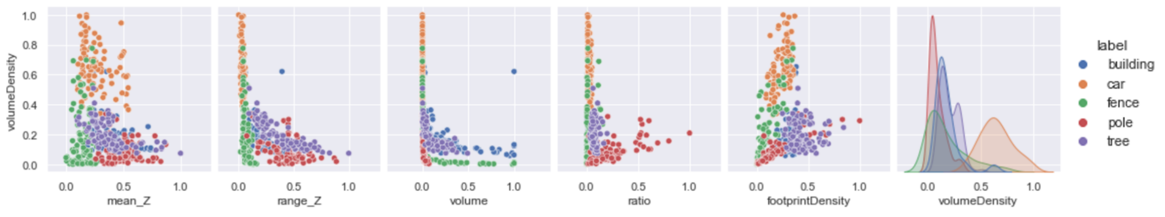


	mean_Z	range_Z	volume	ratio	footprintDensity	volumeDensity	label
fileNo							
163	6.277076	0.49	0.795062	0.187023	25.129001	133.322922	car
116	4.970537	1.43	3.641170	0.403954	35.791249	66.462156	car
173	6.298873	1.09	4.167931	0.247166	30.418933	65.979985	car
105	12.880516	0.96	4.888422	0.209150	26.589723	63.415149	car
107	5.444792	1.09	5.316948	0.247728	28.708134	63.194154	car
149	5.973707	1.38	5.165810	0.382271	20.873260	62.139339	car
109	6.598358	0.99	4.388885	0.235714	20.583654	61.063350	car
182	6.166316	1.04	5.739589	0.225108	32.755191	59.586146	car
150	5.698220	1.02	5.231260	0.485716	29.725978	59.067991	car
170	7.837321	1.04	4.502963	0.525258	29.808462	58.850135	car

Figure 3.5 *Top 10 objects of biggest volumeDensity*



(a) No modification



(2)Modification of obj. 163 (with 67)

Figure 3.6 *Modification of volumeDensity*

3.3 Feature analysis

We performed two analyses. One is the **scatter plot**. We quickly plotted against one another in 2D via *seaborn* and it can be seen that some of the features are already well distinguishable. The 3D scatter plots will be shown in chapter 4.

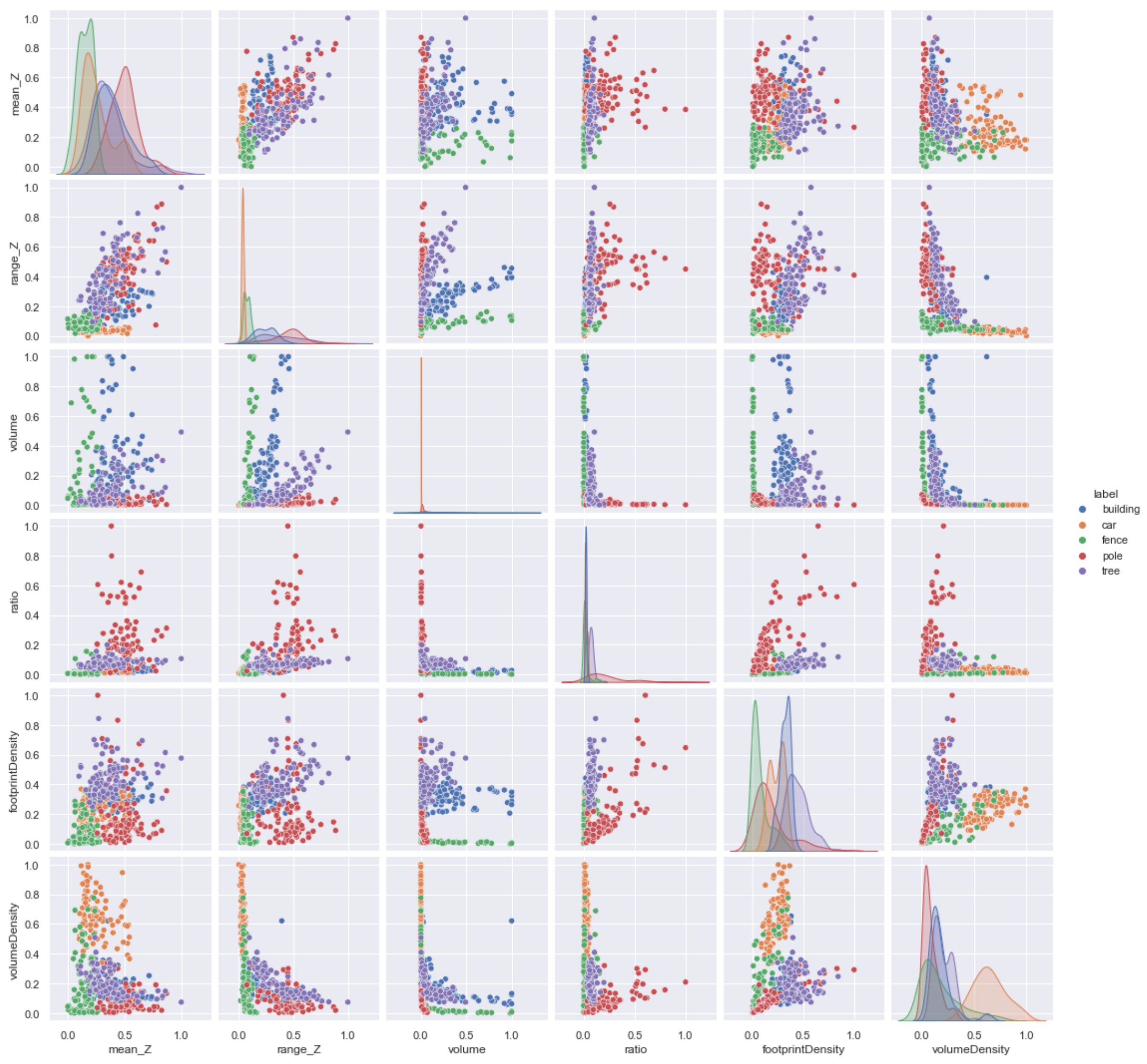


Figure 3.7 *Scatter plot in 2D feature space*

The other is **correlation matrix**. As we know, irrelevance/independence is usually encouraged among the features in feature design. This correlation matrix can give us hints to kick out highly correlated features. For example, mean_Z and range_Z don't need to be used in one test.

	mean_Z	range_Z	volume	ratio	footprintDensity	volumeDensity
mean_Z	1	0.68	0.04	0.36	0.25	-0.32
range_Z	0.68	1	0.12	0.49	0.31	-0.58
volume	0.04	0.12	1	-0.12	0.01	-0.21
ratio	0.36	0.49	-0.12	1	0.28	-0.20
footprintDensity	0.25	0.31	0.01	0.28	1	0.17

volumeDensity	-0.32	-0.58	-0.21	-0.20	0.17	1

Table 3.3 *Correlation matrix of the normalised features*

4 Classification and evaluation

The experiment is conducted in *Jupyter Notebook*, and the imported libraries included *numpy, pandas, scikit-learn, scipy, glob, seaborn and matplotlib.pyplot*. We mainly used *Pandas Dataframe* to process and store data and *scikit-learn* for classification and evaluation.

We set the parameters as shown in Table 4.1.

Module	Function	parameters
sklearn.model_selection	*train_test_split()*	test_size=0.3, random_state=45
	learning_curve()	/
sklearn.metrics	*accuracy_score()*	/
	confusion_matrix()	/
sklearn.svm	*SVC()*	decision_function_shape='ovo', kernel='linear'
sklearn.ensemble	*RandomForestClassifier()*	n_estimators = 50, max_depth=2, random_state=0

Table 4.1 *Parameter settings*

4.1 Accuracy

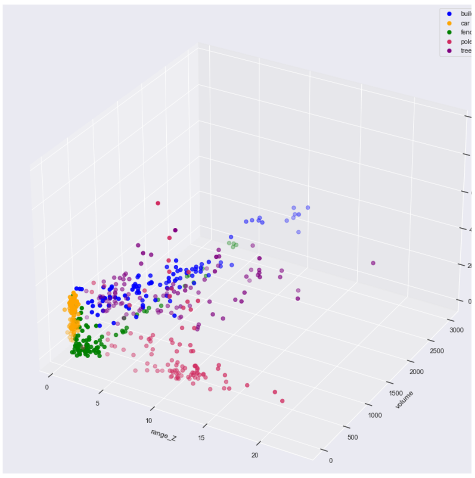
In this particular case, since the number of objects in each class is equal, overall accuracy is equal to mean per-class accuracy. We tried multiple feature combinations and obtained the accuracy shown in Table 4.2.

feature combination	SVM accuracy	RF accuracy	Average accuracy
[h, v, r]	0.65	0.79	0.72
[h, v, fDen]	0.84	0.79	0.815
[h, v, vDen]	0.87	0.71	0.79
[h, r, fDen]	0.8	0.88	0.84
[h, r, vDen]	0.79	0.91	0.85
[mean_h ,v, r]	0.53	0.81	0.67
[mean_h, v, fDen]	0.71	0.77	0.74

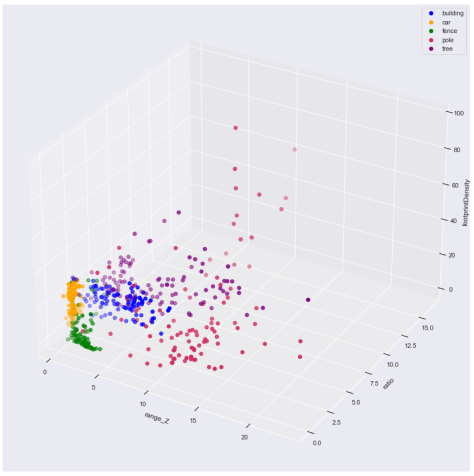
[mean_h, v, vDen]	0.73	0.70	0.715
[mean_h, r, vDen]	0.62	0.86	0.74
[v, r, fDen]	0.76	0.91	0.835
all	0.94	0.92	0.93

Table 4.2 *Accuracy of different feature combination*

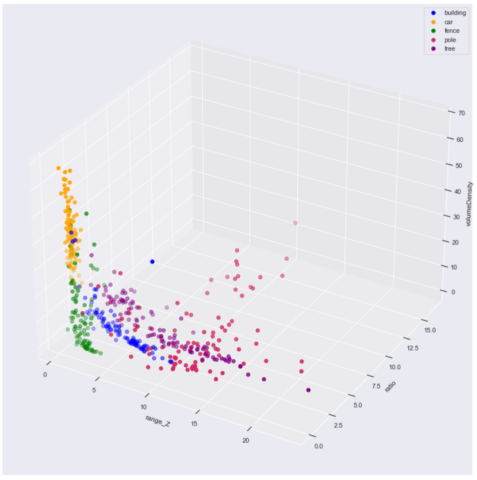
(pink: good results for SVM, green: good results for RF, blue: good results for both)



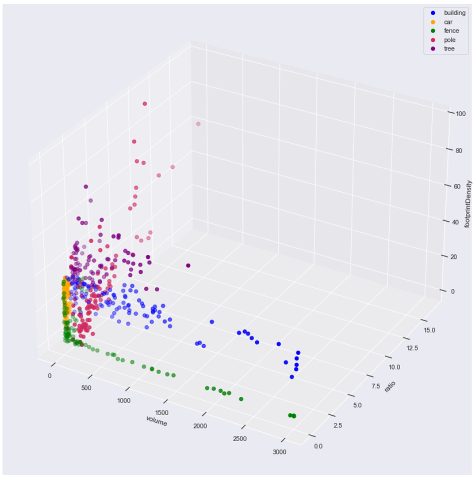
(a) [h, v, fDen]



(b) [h, r, fDen]



(c) [h, r, vDen]



(d) [v, r, fDen]

Figure 4.1 *Scatter plot in 3D feature space*

From Figure .9, Figure 4.1 and Table 4.1, some figures in our A1 report and Lisa-Marie Mueller and Sebastian Stripp's A1 report, our conclusions are:

(1) **[height, ratio, volumeDensity] and [height, ratio, footprintDensity]** are generally the best combinations, with a classification accuracy of around 80%.

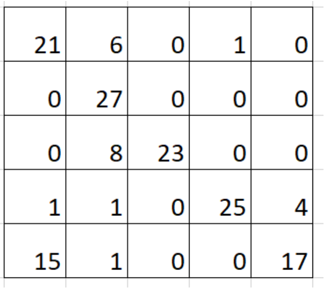
(2) Feature selection affects the accuracy greatly. Bad features will definitely cause bad classification results.

(3) Different feature combinations' performance may vary in different algorithms. For example, one combination may perform well in random forest, but perform badly in SVM, i.e. [mean_height, volume, ratio].
(4) In this case, supervised learning (classification) performs much better than unsupervised learning (clustering). The former one can reach an accuracy of around 80%, while the latter one can only reach around 50%.

4.2 Confusion matrix

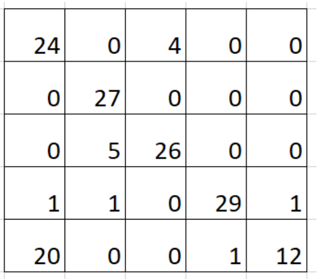
We obtained the confusion matrix with the feature combination of [height, ratio, volumeDensity].
From Figure 3.7, our conclusions are:

(1) The two algorithms generally perform well.
(2) They make mistakes in similar cases, but slightly different.



21	6	0	1	0
0	27	0	0	0
0	8	23	0	0
1	1	0	25	4
15	1	0	0	17

(a) SVM



24	0	4	0	0
0	27	0	0	0
0	5	26	0	0
1	1	0	29	1
20	0	0	1	12

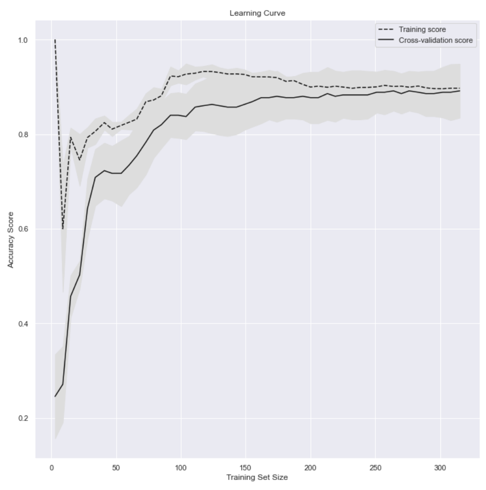
(b) Random forest

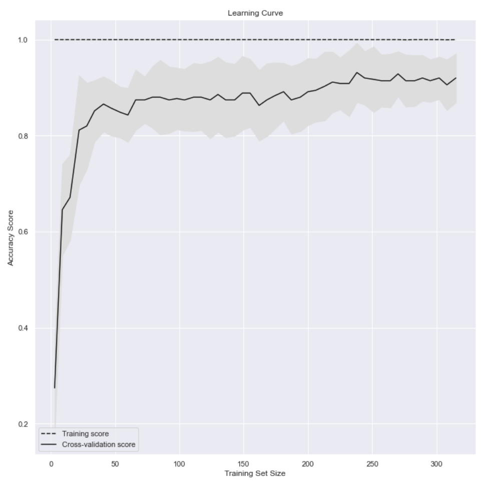
Figure 4.2 *Confusion matrix*

4.3 Learning curve

We obtained the learning curve with the feature combination of [height, ratio, volumeDensity].
From Figure 4.3, our conclusions are:

(1) Training score in both graphs has similar patterns. The accuracy score increases significantly with more training samples in the beginning, but once arriving at a certain value (near 50), the accuracy score becomes steady.
(2) Cross-validation score of SVM has a similar pattern compared to its training score, but somehow that of RF is wrong (always equal to 1).





(a) SVM *(2) Random forest*

Figure 4.3 *Learning curve*

5 Discussion

Our findings are already well presented in Chapter 4.

Potential improvement:

1. Design better features, i.e. roughness (suggested by Max van Schendel), planarity (suggested by Lisa-Marie Mueller and Sebastian Stripp)
2. A more scientific outlier detection and processing. This time we did it manually, but it can be more efficient to design an algorithm for that.
3. Add weight to key features, like height, and the weight should certainly be decided by experiment.
4. Integrate some Intelligent algorithms to determine parameters, i.e. the ant colony algorithm.

Work distribution

Name	Work in code	Work in report
Pratyush	Load dataset Feature Engineering (shared) Classification	Full cooperation
Cynthia	Feature Engineering (shared) Evaluation	

Table 5.1: *Work distribution in the assignment*