GEO5016 - Machine Learning for the Built Environment

Assignment 1

Walter Kahn (5658543), Michele Giampaolo (5865506), Sharath Chandra Madanu - 5722101

TU Delft, Faculty of Architecture and The Built Environment

The Netherlands

1. Introduction

In this assignment we will use the two different supervised learning techniques in order to classify a dataset of pointclouds of urban objects. The given data is already segmented and classified so that it can be used for testing and validation. It is composed of 500 pointclouds with 5 classess: building, car, fence, pole, tree, where each of the classes has 100 pointclouds associated to it. We applied SVM and random forest to carry out the classification and analysed their outcome.

2. Methodology

2.1 Feature Engineering

In order to design a feature set for the classification we firstly decided to visualise the given pointclouds to intuitively detect similarities between pointclouds of the same object type (Figure 1). At first glance, we could see, for example, that some classifications seemed to have a larger number of points (buildings, trees), some had more linear elements (fences, poles) and some had a lower point density (cars, poles). From this initial observation we eventually came to 15 features that we thought could prove efficient in the classification by describing different geometrical properties of the object (Table 1). This feature set has a size of 15 that is relatively large, which can cause problems such as computational complexity and others deriving from the "curse of dimensionality". In order to decrease it and find the optimal set for this assignment, we mainly used brute-force approach to find the optimal set of features.

Number of points	nop	Number of points that make up the pointcloud object.
Linearity	lin	The degree to which points in a region tend to align along a straight line.
Sphericality	sph	the degree to which points in a region tend to be distributed evenly around a central point, resembling a shpere.
Verticality	vrt	How closely points of the pointcloud align with the vertical axis.
Planarity	pln	How closely the points of the pointcloud tend to lie on a flat surface.
Point normals x	pnx	The x value of the normal vector of the pointcloud based on its covariance matrix.
Point normals y	pny	The y value of the normal vector of the pointcloud based on its covariance matrix.
Point normals z	pnz	The z value of the normal vector of the pointcloud based on its covariance matrix.
Bounding box size	bbx	The volume of the shell bounding the pointcloud.
Average distance to centroid	adc	The average distance of every point from the pointcloud centroid.
Covariance xyz	cov	The covariance matrix of the x,y and z coordinates, describing the statistical relationship between the spatial distribution of the object's points .
Root mean square error	rmse	The average distance of every point from the pointcloud centroid.
Spread over x	sox	The difference between the maximum and minimum x values of the points in the pointcloud.
Spread over y	soy	The difference between the maximum and minimum y values of the points in the pointcloud.
Spread over z	soz	The difference between the maximum and minimum z values of the points in the pointcloud.

Table 1: The chosen features, their abbreviations used in the report and their description.

Figure 1: The classified given pointclouds, from left to right: building, car, fence, pole, tree (best viewed in Adobe Acrobat)

2.2 Model configurations and hyperparameters

When implementing Random Forest, we decided to test it with two different model configurations. These differed in the amount of features that each set had. The first one used a 2d feature set and the second one a 3d feature set. The Random forest hyperparameter of the test sample size was also tested with different values. Specifically with 9 different values: 50 (10%), 100 (20%), 150 (30%), 200 (40%), 250 (50%), 300 (60%), 350 (70%), 400 (80%), 450 (450%). The overall accuracy per test sample size as well as the mean per-class accuracies were calculated.
When implementing SVM, we tried to find the best hyperparameters by testing a chosen set of them over the same sample test sizes mentioned for the Random Forest. In the case of the Radial Basis Function kernel these were gamma and C, while for the Polynomial kernel these were degree and C. We tested the 9 possible combinations when giving each of them a value of 0.1, 1.0 or 10.0. This was done using a feature set that included all 15 features. We then picked the best performing combination of parameters to be our hyperparameters. Then, these were used to find the best performing

features with lower dimensional feature sets, as well as comparing the different dimensions' accuracies.

3. Experiments and evaluation

3.1 Random forest

3.1.1 Classification results

To analyse the classification using random forest, we used different parameters and configurations and compared them. Both 3d and 2d feature sets were considered, and for each the best combination was found as the test sample size increased. What we found was that the use of 3 features showed a significant increase in accuracy, around a 10% increase. We also saw that some features proved to be the most efficient in classifying the objects in their different classes (Figures 2 and 3). Particularly, *spread over z* was part of every optimal feature combination, both in 2d and 3d. *Root mean square error* was also consistently part of the most accurate feature sets in 2d (Table 2). While in 3d the other best features were *average distance to centroid* and *sphericality* (Table 3). The worst performing feature sets were, as expected, those where the same feature was repeated, specifically these were *point normal x, point normal y,* and *covariance xyz.*

test sample size	best feature set	best accuracy	worst feature set	worst accuracy
50	rmse, soz	0.900	pnx, pnx	0.160
100	rmse, soz	0.890	pny, pny	0.210
150	rmse, soz	0.880	pny, pny	0.260
200	nop, soz	0.865	pny, pny	0.265
250	rmse, soz	0.872	pny, pny	0.276
300	rmse, soz	0.863	cov, cov	0.303
350	rmse, soz	0.874	cov, cov	0.323
400	rmse, soz	0.835	pny, pny	0.313
450	rmse, soz	0.828	cov, cov	0.320

Table 2: Optimal and worst 2 dimensional feature choice and their accuracies at varying test sample sizes using RF

test sample size	best feature set	best accuracy	worst feature set	worst accuracy
50	nop, adc, soz	0.980	pnx, pnx, pnx	0.160
100	sph, sox, soz	0.970	pny, pny, pny	0.210
150	nop, adc, soz	0.980	pny, pny, pny	0.260
200	nop, adc, soz	0.960	pny, pny, pny	0.265
250	sph, adc, soz	0.956	pny, pny, pny	0.276
300	sph, adc, soz	0.950	cov, cov, cov	0.303
350	sph, adc, soz	0.937	cov, cov, cov	0.323
400	sph, adc, soz	0.938	pny, pny, pny	0.313
450	sph, adc, soz	0.904	cov, cov, cov	0.320

Table 3: Optimal and worst 3 dimensional feature choice and their accuracies at varying test sample sizes using RF

When analysing how the sample size affected accuracy, an overall decrease in accuracy can be seen in both cases with an increase in test sample size. This could make it tempting to consider the configuration with the smallest sample sizes to be the most accurate model, however this is likely due to overfitting, and would result in higher error if applied in reality. Therefore, the most optimal model is a split closer to 6:4 between training and test set. In our case would mean 300 objects for the training set and 200 object for the test set. The accuracy that corresponds to this split using random forest is of 96%.

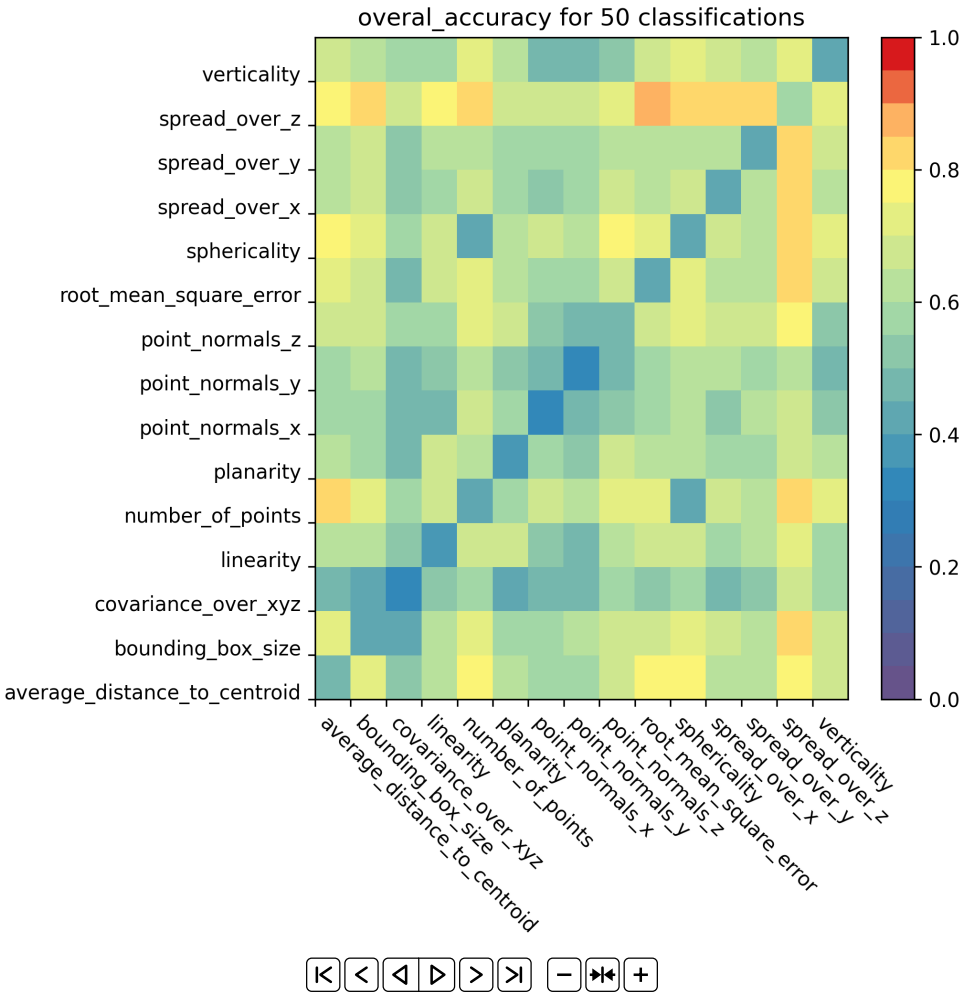

Figure 2: 2d classifier raster (best viewed in Adobe Acrobat) (use controls to pause and speed up)

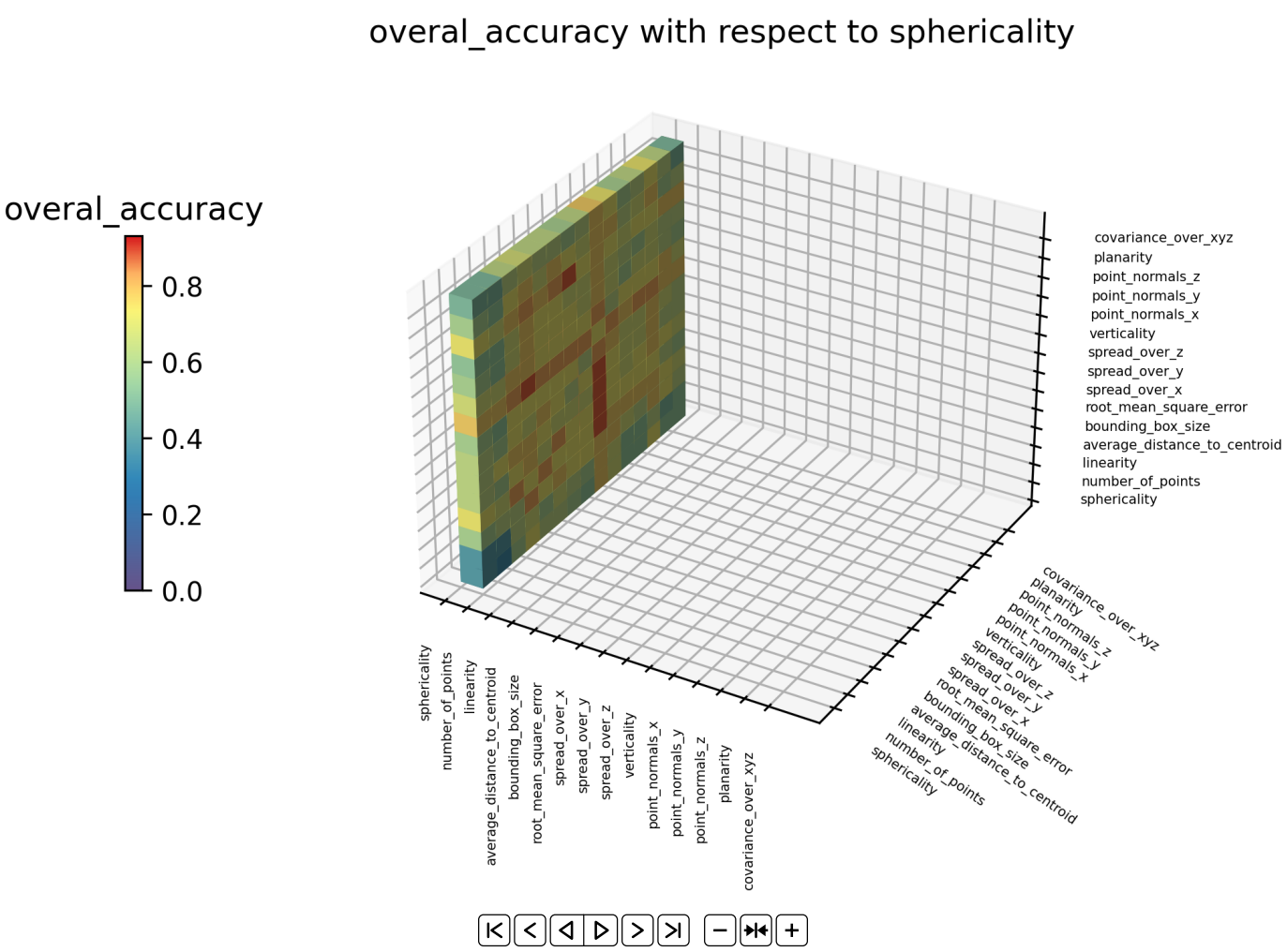

Figure 3: 3d classifier voxels (best viewed in Adobe Acrobat) (use controls to pause and speed up)

3.1.2 Confusion matrices

These are the confusion matrices for the best 2d and 3d classification results for the 6:4 split (Figure 5). From these we can see the accuracy of the two models and in which classes they perform better and worse. Overall we can again see an increase in accuracy from the 2d feature set to the 3d feature set. The classes that performed better in the 2d case were "pole" and "tree", while it under performed with the "fence" and "building" classes. In the 3d case, the most accurate classes were "pole" and "car", while the worst were "tree" and "building". Specifically, buildings were often confused with fences and trees, fences with cars and trees with poles and cars.

3.1.3 Learning curves

The learning curves of the Random forest show clearly the difference between using two and three features per set in their accuracy. It also shows the previously mentioned decrease in accuracy as the test sample size increases, which is probably due to overfitting.

0.8566666666666667
spread_over_z
sphericality
0.6

	building	car	fence	pole	tree	total
building	057	000	009	000	004	070
car	000	053	010	000	000	063
fence	003	009	035	000	001	048
pole	000	000	000	061	003	064
tree	001	000	000	003	051	055
total	061	062	054	064	059	300

(a) 2d RF classification

0.93
sphericality
average_distance_to_centroid
spread_over_z
0.6

	building	car	fence	pole	tree	total
building	061	000	002	000	004	067
car	000	055	002	000	000	057
fence	000	003	050	000	000	053
pole	000	000	000	059	001	060
tree	000	004	000	005	054	063
total	061	062	054	064	059	300

(b) 3d RF classification

Figure 5: Best resulting confusion Random Forest matrices

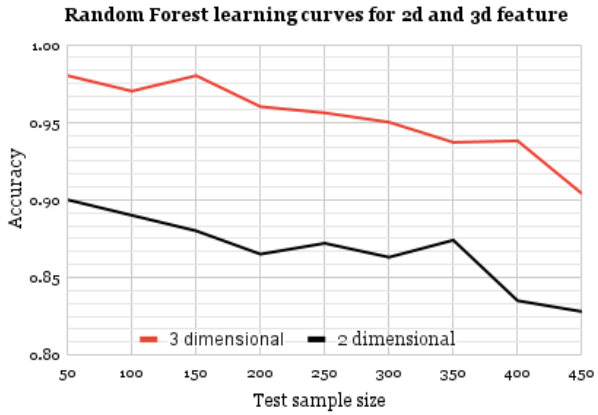

Figure 4: Random Forest learning curves

3.2 Support Vector Machine

3.2.1 Classification results

The data is classified using two SVM kernels: Polynomial and Radial Basis Function (RBF). The two hyperparameters in RBF are C (the regularization parameter) and Gamma (which regulates the Gaussian distribution's width). The hyperparameters chosen for both kernels are mentioned in the Table below.

SVM Kernel	C	gamma	degree
RBF	0.1, 1.0, 10.0	0.1, 1.0, 10.0	-
Polynomial	0.1, 1.0, 10.0	-	1, 2, 3

Table 4: Hyperparameters choice.

Learning curves for both kernels are depicted Figure 6 and Figure 7. All fifteen features developed were used to fit the data to create these learning curves.

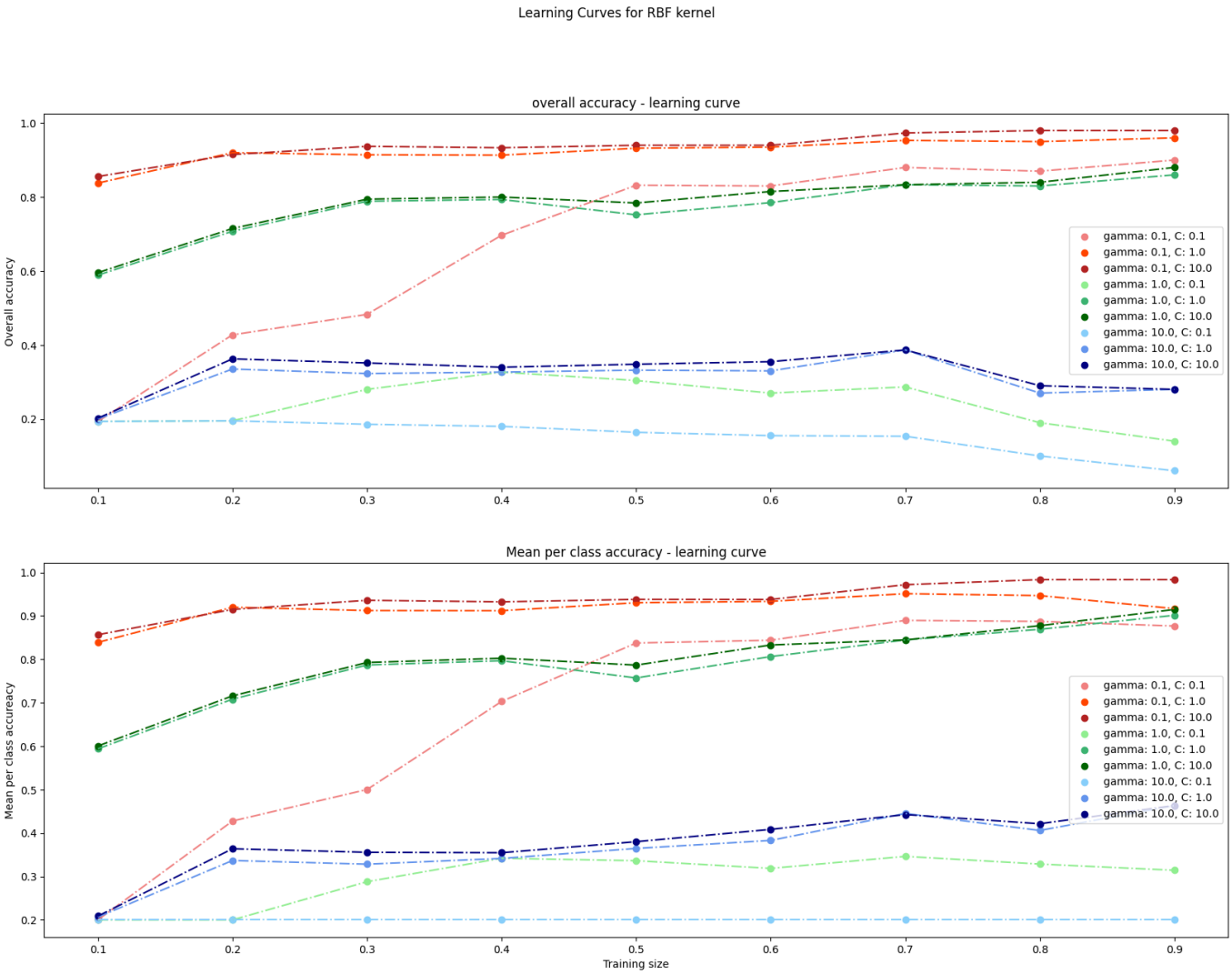

Figure 6: Learning curve of RBF SVM.

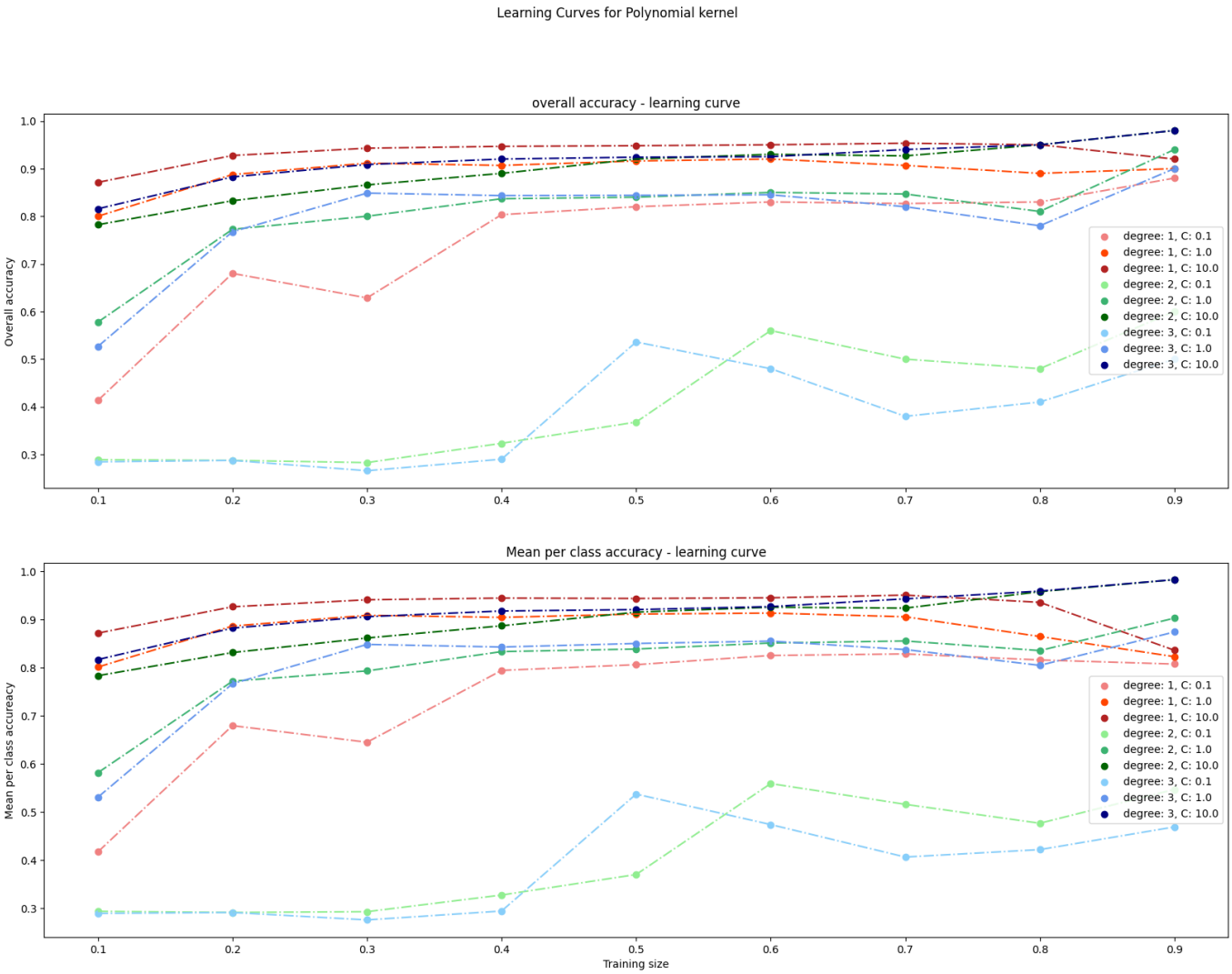

Figure 7: Learning curve of Polynomial SVM.

Some observations from the learning curves:

- From the above images, it is clear that RBF models with a high value of gamma (10) consistently underperformed. This is because, with higher gamma values, the classifier model complicates the decision boundary and tries to overfit the training data.
- Overall accuracy is increasing for a fixed gamma with an increasing C value. This means that more regularisation better the performance for a fixed gamma.
- Something very similar is observed with the SVM polynomial; with the higher degrees the model failed to achieve good accuracy on test data. This could be attributed to the fact that the decision boundary is complicated at higher degrees.
- When the models are regularized with higher C values for a fixed degree, they achieve better accuracies.
- Better performing models from both the kernels did not benefit from having more training data after a point. This saturation of accuracy happened around when 40% of the whole data was used for training.

We choose the best perming models from both kernels for further analysis. The hyperparameters chosen are in the below table.

SVM Kernel	**hyperparameters**
RBF	gamma = 0.1, C = 10.0
Poly	degree = 1, C = 10.0

Table 5: hyperparameters for best models.

We came up with 15 features which could be used for model training. But it does not necessarily means that all the features developed contribute equally to the model's performance. So it is equally important to understand how many features are minimally required and what those parameters are.
For that first, the data is split in a 60-40 ratio for training and testing purposes. The hyperparameters in the above table are used for training the models.

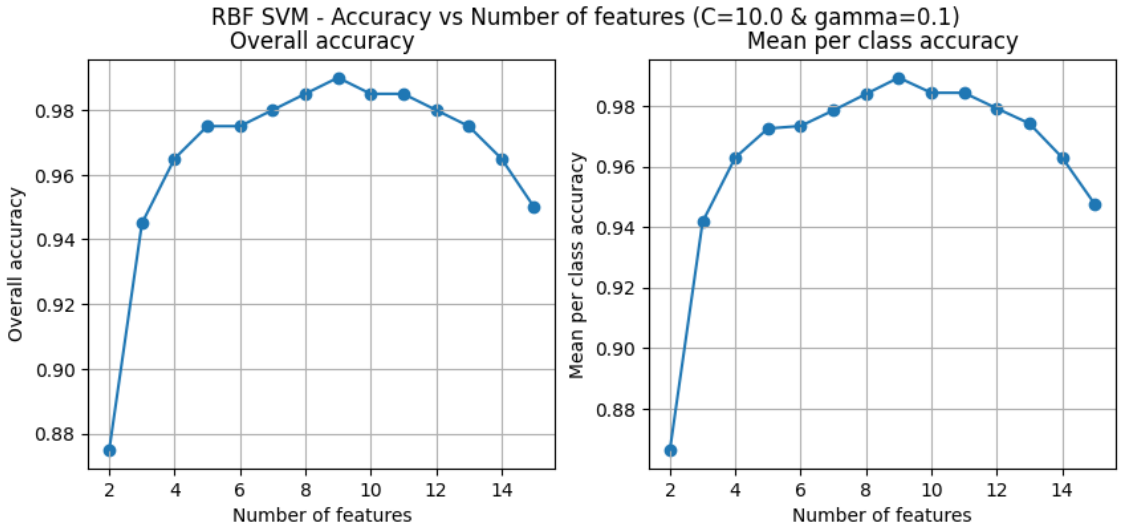

Figure 8: RBF SVM accuracy with increasing features size.

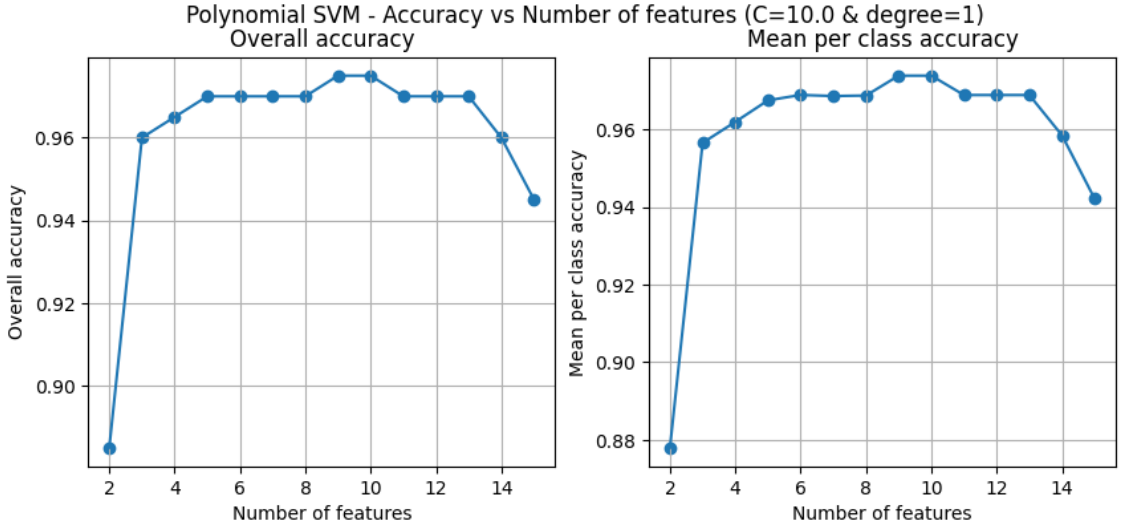

Figure 9: Polynomial SVM accuracy with increasing features size.

Some interesting conclusions from the image are:

- Adding features can only help a model to a certain extent, and after that, these extra features are hurting the model's performance.
- For RBF SVM best number of features is nine (Figure 8). (See the below table for features' names)
- For Polynomial SVM best number of features is ten (Figure 9). (See the below table for features' names)

features size	best features	accuracy
2	sph, soz	0.875
3	sph, adc, soz	0.945
4	sph, adc, soz, pny	0.965
5	sph, adc, rmse, soz, pnz	0.975
6	sph, nop, adc, soz, vrt, pnz	0.975
7	sph, lin, adc, rmse, soy, soz, pnz	0.980
8	sph, nop, adc, soy, soz, pnz, pln, cov	0.985
9	sph, lin, adc, rmse, soy, soz, pnz, pln, cov	0.990
10	sph, nop, lin, adc, bbx, soy, soz, pnz, pln, cov	0.985
11	sph, nop, lin, adc, bbx, rmse, soy, soz, pnz, pln, cov	0.985
12	sph, nop, lin, adc, bbx, rmse, soy, soz, ver, pnz, pln, cov	0.980
13	sph, nop, lin, adc, bbx, rmse, sox, soy, soz, ver, pnz, pln, cov	0.975
14	sph, nop, lin, adc, bbx, rmse, sox, soy, soz, ver, pnx, pnz, pln, cov	0.965
15	sph, nop, lin, adc, bbx, rmse, sox, soy, soz, ver, pnx, pny, pnz, pln, cov	0.950

Table 6: RBF SVM - features for best fitting classifier (gamma = 0.1, C = 10.0, training to test = 60-40).

features size	best features	accuracy
2	sph, soz	0.885
3	sph, adc, soz	0.960
4	sph, adc, soz, pnz	0.965
5	sph, lin, rmse, soz, pln	0.970
6	sph, nop, adc, sox, soz, vrt	0.970
7	sph, nop, lin, adc, sox, soz, pnz	0.970
8	sph, nop, lin, adc, bbx, sox, soz, pnz	0.970
9	sph, nop, lin, adc, rmse, soz, pnz, pln, cov	0.975
10	sph, nop, lin, adc, rmse, sox, soz, onx, pnz, cov	0.975
11	sph, nop, lin, adc, bbx, rmse, sox, soz, vrt, pny, pnz	0.970
12	sph, nop, lin, adc, bbx, rmse, sox, soz, vrt, pnx, pny, pnz	0.970
13	sph, nop, lin, adc, bbx, rmse, soy, soz, vrt, pnx, pny, pnz, cov	0.970
14	sph, nop, lin, adc, bbx, rmse, sox, soy, soz, vrt, pnx, pny, pnz, cov	0.960
15	sph, nop, lin, adc, bbx, rmse, sox, soy, soz, vrt, pnx, pny, pnz, pln, cov	0.945

Table 7: Polynomial SVM - features for best fitting classifier (degree = 1, C = 10.0, training to test = 60-40).

	precision	recall	f1-score	support
building	1.00	0.97	0.99	39
car	0.98	1.00	0.99	48
fence	0.97	0.97	0.97	36
pole	1.00	1.00	1.00	40
tree	1.00	1.00	1.00	37
accuracy			0.99	200
macro avg	0.99	0.99	0.99	200
weighted avg	0.99	0.99	0.99	200

(a) Classification report of RBF SVM classifier.

```
[[38  0  1  0  0]
 [ 0 48  0  0  0]
 [ 0  1 35  0  0]
 [ 0  0  0 40  0]
 [ 0  0  0  0 37]]
```

(b) RBF confusion matrix

Figure 10: Performance of RBF SVM. (gamma = 0.1, C = 10.0, test-train = 60-40)

The classification reports of both the RBF and Polynomial SVMs show that both the best models had difficulty identifying cars correctly.

4. Conclusion

The assignment allowed us to better understand the implementation of the two supervised learning techniques. Specifically, it was made clear how important a good feature set selection is essential in order to have an accurate model, as we saw that incorrectly choosing them could have resulted in accuracy around 20% in this case. The effect of the test sample size

	precision	recall	f1-score	support
building	1.00	1.00	1.00	39
car	0.92	1.00	0.96	48
fence	1.00	0.92	0.96	36
pole	0.98	1.00	0.99	40
tree	1.00	0.95	0.97	37
accuracy			0.97	200
macro avg	0.98	0.97	0.98	200
weighted avg	0.98	0.97	0.97	200

(a) Classification report of Polynomial SVM classifier.

```
[[39  0  0  0  0]
 [ 0 48  0  0  0]
 [ 0  3 33  0  0]
 [ 0  0  0 40  0]
 [ 0  1  0  1 35]]
```

(b) Polynimial SVM confusion matrix

Figure 11: Performance of Poly SVM. (degree = 1, C = 10.0, test-train = 60-40)

was also made apparent in the accuracy of the model, and how overfitting can result in seemingly more accurate models. Overall, a good knowledge and visualisation of the initial dataset will bring to optimal hyperparameters and configuration and result in a more successful model.
Better results could be obtained by considering different features that we have not thought of, which might be able to more efficiently distinguish between the urban object classes. Another aspect that could be improved is the way in which we found the optimal feature sets. Although the limited size of our data set (500 pointclouds) allowed us to brute-force the solution, this might not be the case in more complex real-world applications, where brute-forcing would mean more computational complexity especially with higher dimensions of feature sets. Therefore some other sub-optimal, but much less complex, searching algorithms would need to be implemented.

5. Who did what

Report: Michele;
Features design: Walter;
Visualisations: Michele Walter;
RandomForrestClassifier: Walter;
SupportVectorMachine: Sharath;