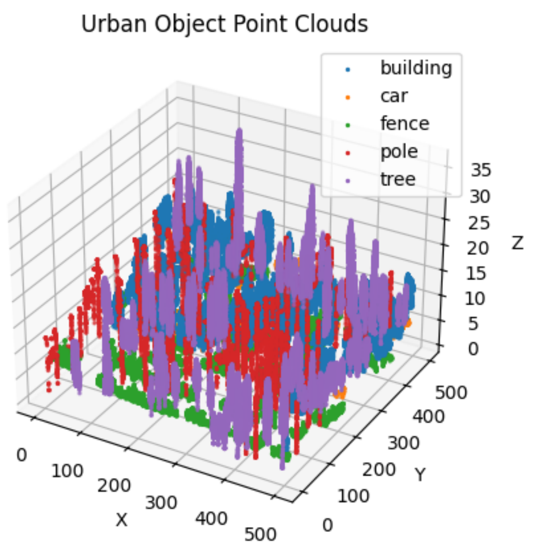
GEO5017 Assignment 2

Haohua Gan & Rafal Tarczynski
Student number: 6007503 & 4439899
Email: h.gan@student.tudelft.nl & r.m.tarczynski@student.tudelft.nl
GitHub: https://github.com/ATRM-Raphael/geo5017_A2.git

March 25, 2024



Contents

1. Introduction

Classification of urban objects can provide accurate, three-dimensional insight into the urban environment. Using 3D scans to make point cloud can be very time efficient and excel surveying processes. In this assignment, we aim to construct a set of features (at least 6) and use two supervised machine learning methods, Support Vector Machine (SVM) and Random Forest (RF), to classify point cloud (fig. 1) 5 categories of urban objects: buildings, cars, fences, poles and trees, with each category containing 100 labelled samples (table 1).

2. Methodology

2.1 Feature engineering

We use an "object-slicing" method to describe an urban object, which is to slice the object along XYZ axes into a certain number of slices as shown in figure 2. Then, for the object, two properties (point number and point density) are combined with slices in three dimensions to construct 6 types of features,

- 1.point numbers of slices_x 2.point densities of slices_x 3.point numbers of slices_y
- 4.point densities of slices_y 5.point numbers of slices_z 6.point densities of slices_z

The point density of a slice (D_{slice}) is calculated by dividing the point numbers of a slice (N_{slice}) by the volume of the object's axis-aligned bounding box (V_{object}), as shown in equation 1.

$$D_{slice} = \frac{N_{slice}}{V_{object}} \tag{1}$$

2.2 Model configuration

Grid search is applied to both SVM and RF (fig. 12, 13, 14), to search the optimal combination of hyper-parameters.

For the SVM model, both "poly" kernel and "rbf" kernel are tested. The code looks for the most optimal value of parameter C for "poly" and "rbf" kernels. C which controls the penalty of misclassification of samples. For "poly" kernel, the second parameter is *degree*. For "rbf" kernel, the second parameter is *gamma*, controlling the influence distance of a training sample.

For the RF model, the hyper-parameters we choose are *n_estimators* and *min_samples_leaf*. The *n_estimators* controls the number of trees, and the *min_samples_leaf* controls the minimum number of samples that a leaf node must contain. For both parameters, our code calculates values that yield the highest accuracy.

3. Experiments and evaluation

3.1 Feature selection

A roughly tuned SVM model (kernel='rbf', gamma=1e-10, C=1e+10) is used for testing. Figure 3, 4 and 5 show the test error results of three combinations:

1. Point numbers of slices_x, slices_y and slices_z

2. Point densities of slices_x, slices_y and slices_z
3. Both properties of slices_x, slices_y and slices_z.

According to the figures, it can be observed that using multiple properties better defines an object, reduces minimum test error and requires fewer slices. Initially, increasing the number of slices significantly reduces test errors, but further additions give diminishing returns and can lead to overfitting. The best combination in this case is the third combination with a slice number of every dimension set to 4. Using this combination, every urban object has 12 slices and 24 features.

3.2 Hyperparameters tuning

The evaluation compares the performance of SVM with polynomial and RBF kernels and a Random Forest classifier in a point cloud classification task. We conducted classifications with 3 splits - the training and testing data split respectively 1) 0.5/0.5, 2) 0.6/04, 3) 0.7/0.3. Results and parameters that we present are set to 0.7/0.3, due to the highest overall accuracy and per-class accuracy 2.

The polynomial kernel SVM achieved an accuracy of 84.67%, with the best performance obtained at a high penalty parameter, C 42813323.99 , and a kernel degree of 1 (fig. 12), indicating a linear decision boundary. The RBF kernel SVM displayed a slightly higher accuracy of 86.67%, with C set to 100,000 and a small gamma value - 2.3357214690901214e-08 (fig. 13), suggesting a broad kernel function. The confusion matrix for the RBF SVM indicates a better distinction between classes compared to the polynomial kernel SVM.

The Random Forest classifier outperformed the SVM models with an accuracy of 90%, utilizing 182 estimators and a minimum of 3 samples per leaf (fig. 14), which points to a complexity level adept at capturing data variability without over-fitting.

Overall, the mean per-class accuracies align with the general accuracy trends, where Random Forest achieves the highest mean per-class accuracy at 89.23%. Cross-validation results corroborate the superior consistency and performance of the Random Forest model, which recorded a mean accuracy of 90.60% and the lowest standard deviation among the models evaluated.

3.3 Confusion matrix analysis

The SVM models demonstrated notable accuracy but faced challenges distinguishing between classes with closely related features, indicating areas for potential enhancement through refined feature engineering or model selection.

In contrast, the Random Forest model achieved a more consistent classification across various classes, reflecting its robust generalization capability. Despite its strong performance, slight confusions were observed among similar categories, underscoring the importance of nuanced feature distinction.

3.4 Learning curve analysis

99 different train-test-split ratios are used to generate learning curves, from 1% of samples used for training and 99% for testing to the reverse. Figure 6, 7 and 8 show the learning curves of two models with their best hyperparameters.

For both kernels of the SVM model, as the size of the training set increases, the apparent error increases from almost zero and the true error decreases, indicating that the model is more likely to overfit for small training sets and that adding more training samples could increase generalisation. It can also be seen that the true error curve becomes very variable towards the end of the curve. This is probably due to too few test samples, so even a small change in the test set could lead to very different accuracies. For both kernels, the two error curves start to converge when the size of the training set is towards 500, so there is hardly an opportunity to increase the accuracy of the model by adding more training samples.

For the RF model, the apparent error curve starts from a high error and drops rapidly, maintaining a fairly low error rate as the training sets expand, which may indicate that this RF model is too complex for small training sets and learns well when the training sets are large enough to fit the complexity of the model. The two error curves appear to converge when the training samples are close to 500, so similar to the SVM model, it is unlikely that the performance of the RF model will be improved by adding more training samples.

The learning curves for both models show that the two error curves cross at the end of the curve, which should not theoretically happen. The possible reason for this could be that the test samples are too few, which makes the models sensitive to noise and outliers, resulting in incorrect behaviour.

4. Conclusion

This assignment illustrates that increasing the training set size enhances the accuracy and per-class accuracy for SVM models employing both polynomial and RBF kernels. In contrast, the performance of the Random Forest classifier remains robust, showing little variation with changes in the training set size. Notably, the SVM with a polynomial kernel exhibited slower computational performance compared to its RBF counterpart and the Random Forest model, suggesting a trade-off between accuracy and computational efficiency.

Improvements: To optimize our models further, a comprehensive exploration of the hyper-parameter space is recommended, beyond the initially identified best-performing parameters (see fig. 12, 13, 14). Additionally, addressing the observed fluctuations in the learning curves is crucial. A detailed investigation into their causes—be it noisy data, insufficient training size, or model-specific issues—will provide insights that can guide targeted improvements. Enhancing data preprocessing and considering more sophisticated model tuning strategies could yield better predictive performance and model robustness.

5. Task allocation

- **Haohua**: responsible for the codes for feature engineering, feature curve and learning curve, and writing the corresponding parts of the report as well as the introduction.
- **Rafal**: responsible for the codes for hyperparameters tuning, confusion matrix, model training, cross-validation and integration of all codes (main.py), and writing the corresponding parts of the report.

Appendix

Table 1: *500 point cloud files*

Semantic label	File number
building	000 - 099
car	100 - 199
fence	200 - 299
pole	300 - 399
tree	400 - 499

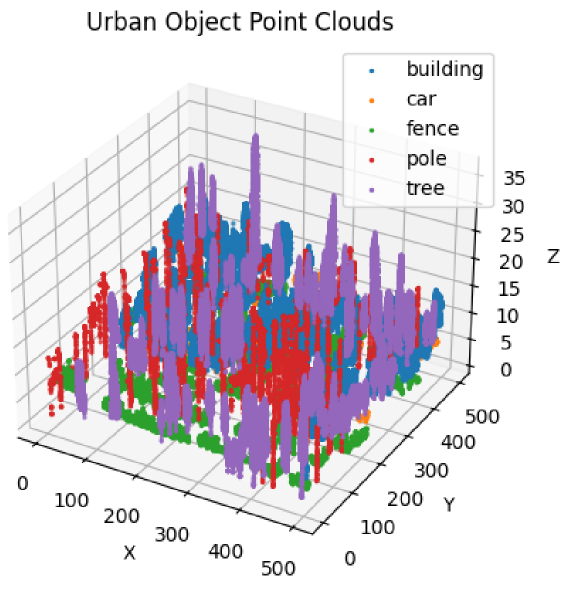


Figure 1: *Point Cloud Visualisation*

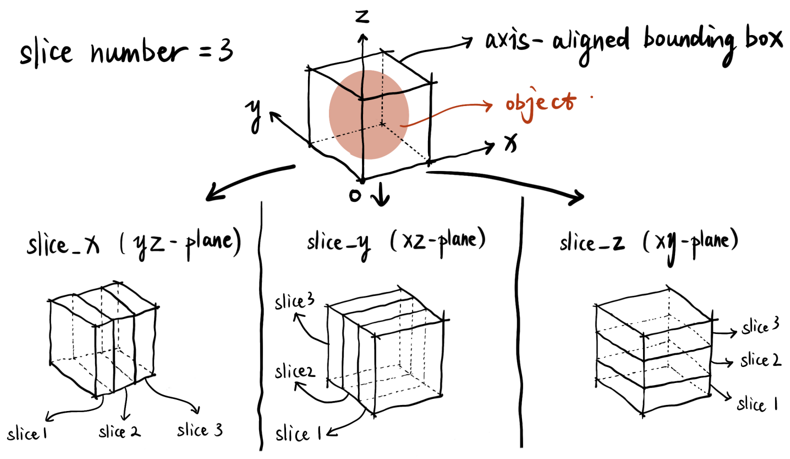


Figure 2: *Object slicing along XYZ axes*

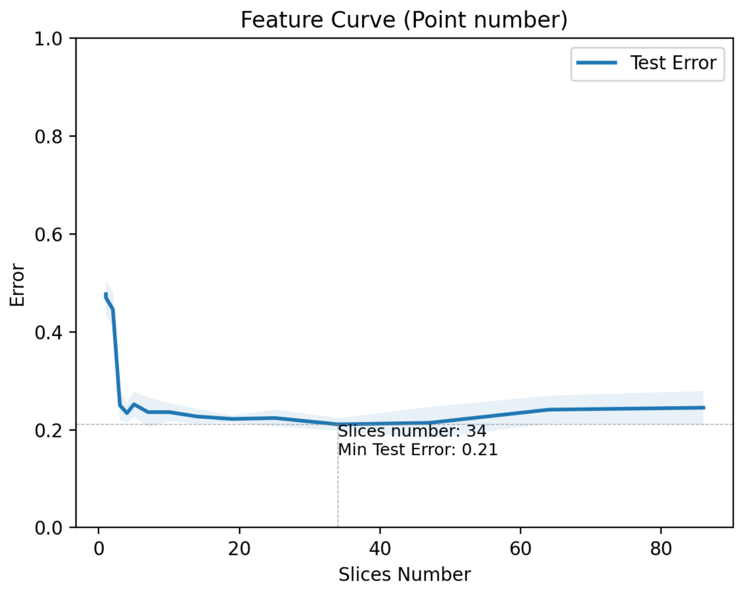


Figure 3: *Test error for different slice number, using only point number for feature construction*

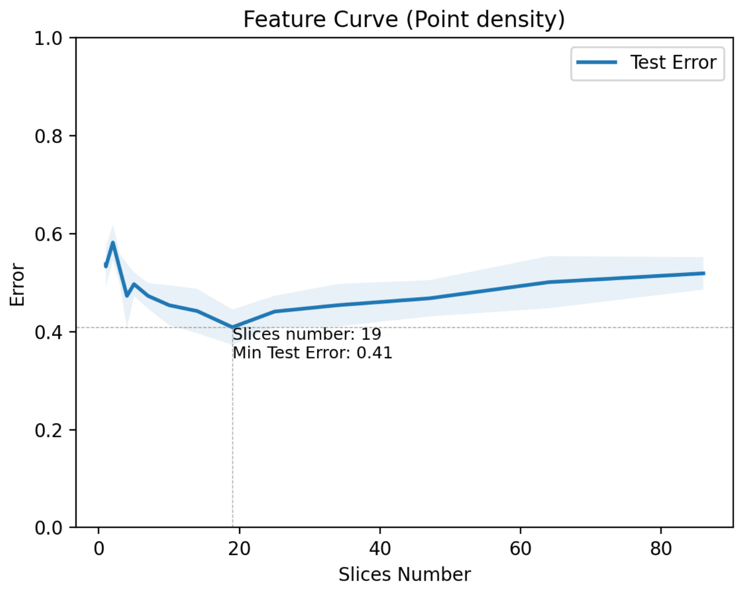


Figure 4: *Test error for different slice number, using only point density for feature construction*

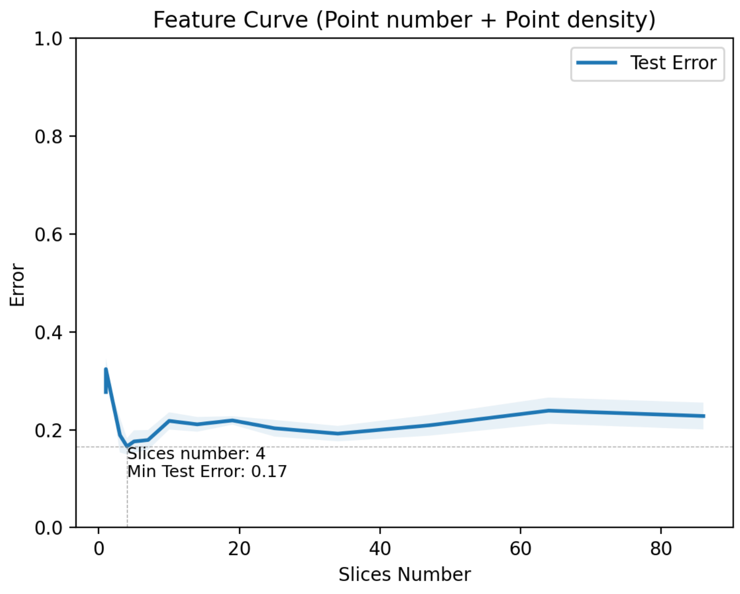


Figure 5: *Test error for different slice number, using both point number and point density for feature construction*

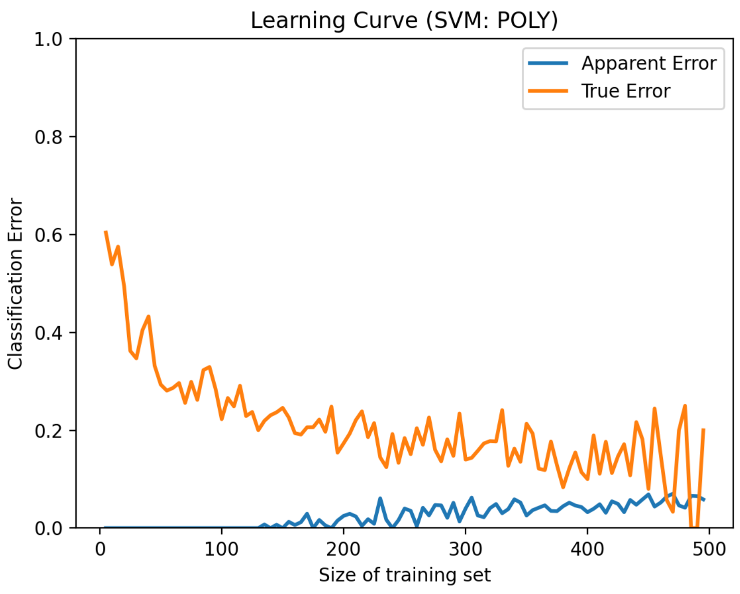


Figure 6: *The learning curve of the best SVM poly model (kernel='poly', degree= 1, C= 4.28e+7)*

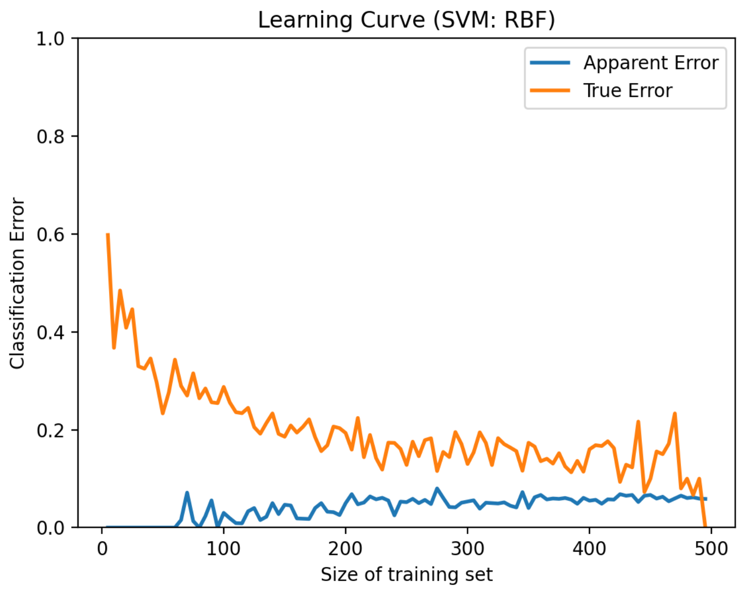


Figure 7: *The learning curve of the best SVM rbf model (kernel='rbf', gamma=2.34e-8, C=1e+5)*

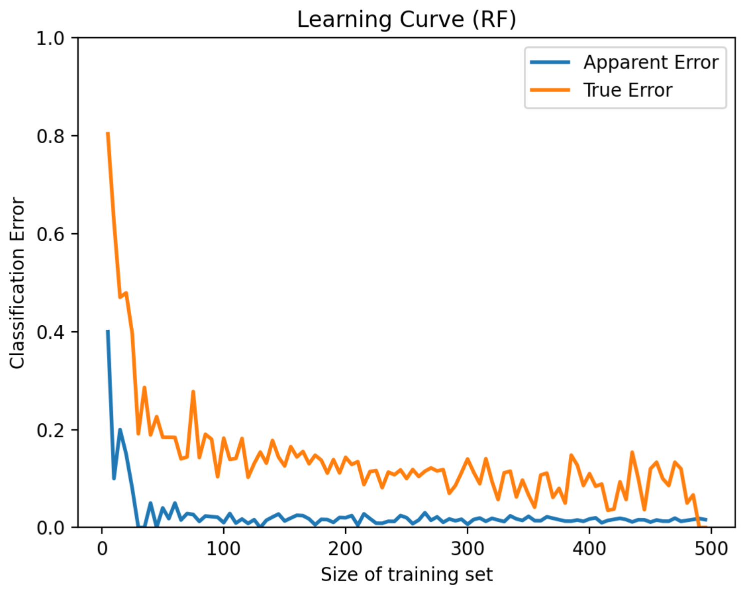
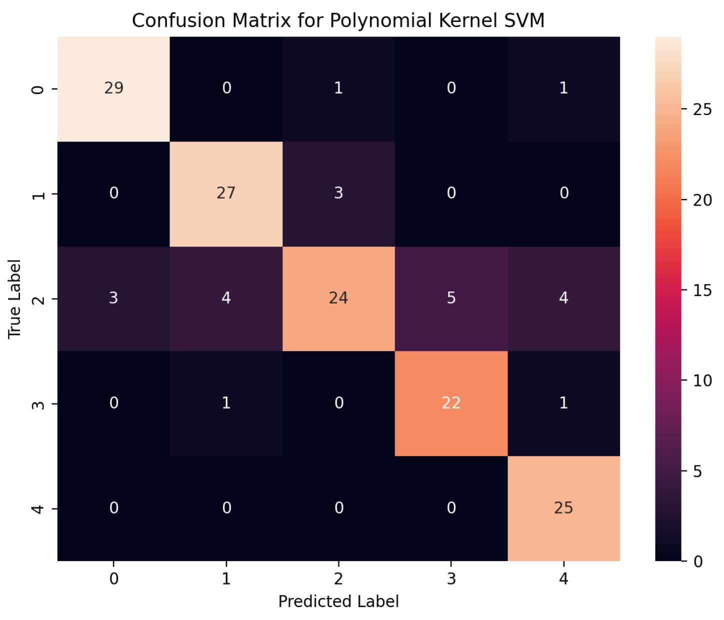
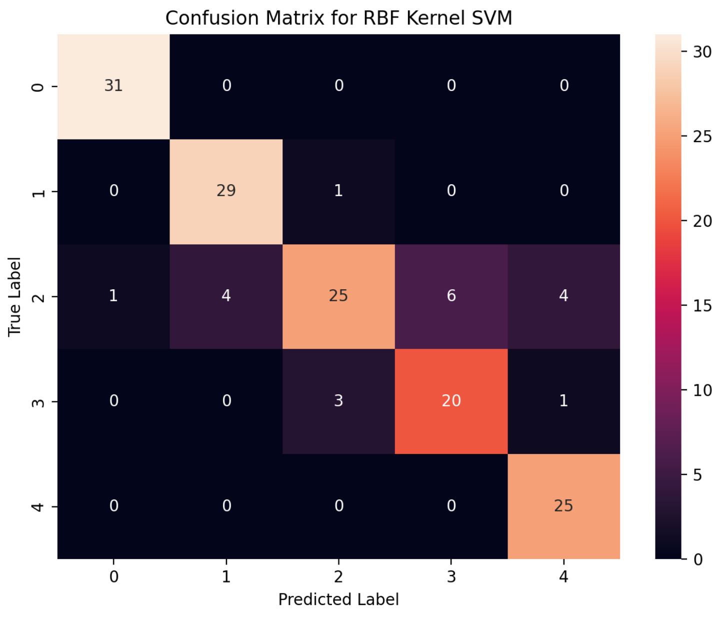


Figure 8: *The learning curve of the best RF model (n_estimators=182, min_samples_leaf=3)*



	0	1	2	3	4
0	29	0	1	0	1
1	0	27	3	0	0
2	3	4	24	5	4
3	0	1	0	22	1
4	0	0	0	0	25

Figure 9: *Confusion Matrix for Polynomial Kernel SVM*



	0	1	2	3	4
0	31	0	0	0	0
1	0	29	1	0	0
2	1	4	25	6	4
3	0	0	3	20	1
4	0	0	0	0	25

Figure 10: *Confusion Matrix for RBF Kernel SVM*

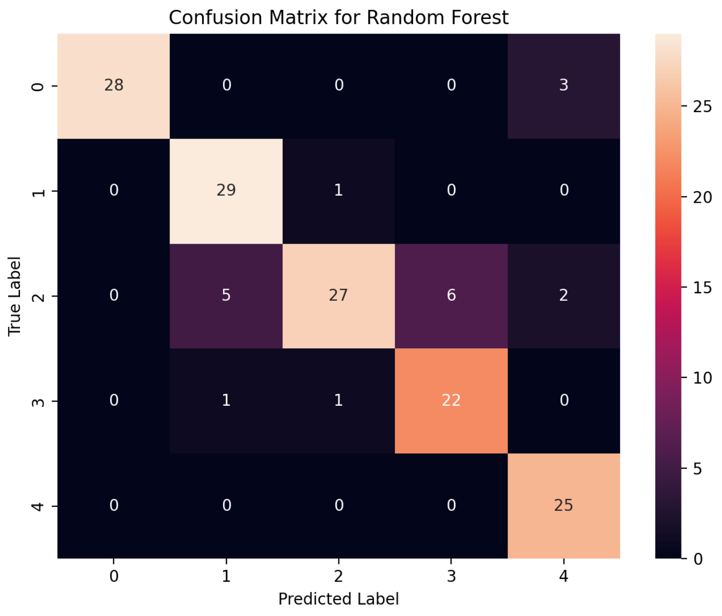


Figure 11: *Confusion Matrix for Random Forest*

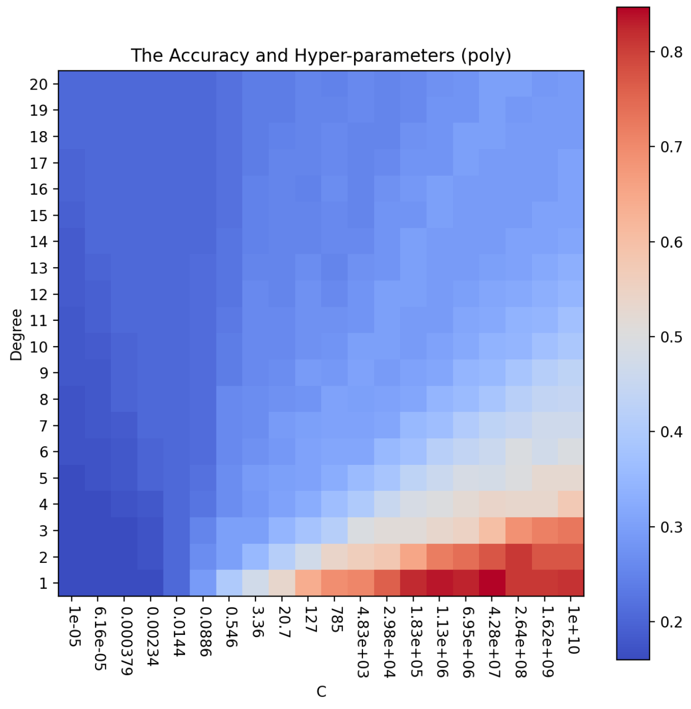


Figure 12: *Hyper-parameter tuning of SVM with polynomial kernel*

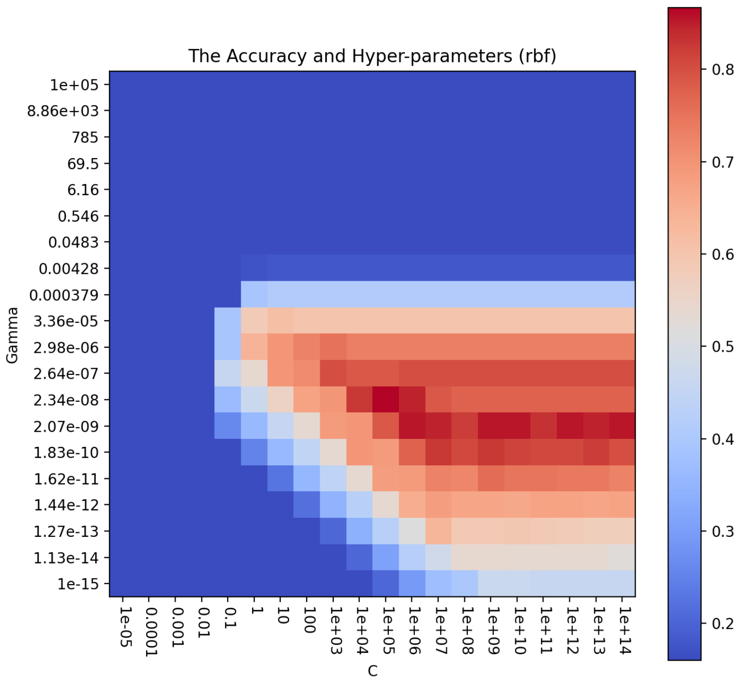


Figure 13: *Hyper-parameter tuning of SVM with rbf kernel*

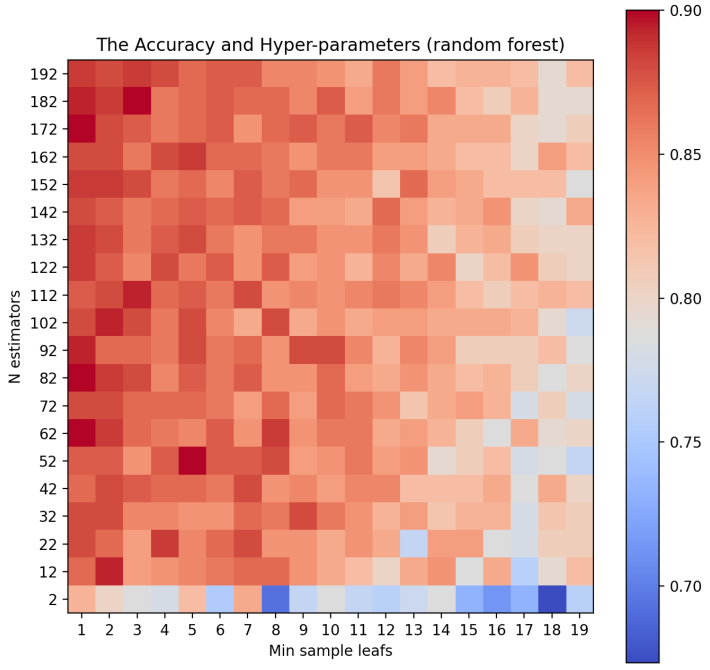


Figure 14: *Hyper-parameter tuning of random forest*

(Training/Test)	**Experiment 1 (0.5/0.5)**	**Experiment 2 (0.6/0.4)**	**Experiment 3 (0.7/0.3)**
SVM POLY Best Accuracy	0.816	0.83	0.8467
SVM POLY Best C Value	6951928	6951928	42813324
SVM POLY Best Degree	1	1	1
SVM RBF Best Accuracy	0.828	0.85	0.8667
SVM RBF Best C Value	100000.0	100000000.0	100000.0
SVM RBF Best Gamma	2.34×10^{-8}	1.83×10^{-10}	2.34×10^{-8}
Random Forest Best Accuracy	0.904	0.92	0.9
Random Forest Best Estimators	122	32	182
Random Forest Best Min Samples Leaf	1	1	3
POLY Mean Per-Class Accuracy	83.00	84.72	87.04
RBF Mean Per-Class Accuracy	84.35	86.14	88.50
Random Forest Mean Per-Class Accuracy	89.92	88.42	89.23
POLY CV Mean Accuracy (Std)	85.00 (2.61)	85.00 (2.61)	84.60 (3.01)
RBF CV Mean Accuracy (Std)	85.80 (1.60)	82.20 (5.42)	85.80 (1.60)
Random Forest CV Mean Accuracy (Std)	91.60 (1.62)	91.00 (2.00)	90.60 (1.20)

Table 2: *Comparison of SVM POLY, SVM RBF, and Random Forest Parameters and Results*