Assignment: GEO5017 A2: Urban Objects Classification from Airborne LiDAR Data | Group 17

1. Charlotte Jeline Kat: 5487528
2. Martijn Nijkamp: 4660838
3. Shreya Kejriwal: 5802709

Introduction:

The main aim of the assignment is to create feature vectors for the point cloud data based on observations and general characteristics of the element. The choice of feature vectors plays a crucial role in determining the accuracy of the model. This provides an opportunity to identify and code multiple features and try various combinations to find the most optimal set.

Methodology:

The first step was to visualize the point cloud data. The .xyz files were compiled and all the coordinates were extracted from the files to create arrays for the coordinates. We used matplot.lib to visualize individual objects and grasshopper + rhino to see all the objects together. Visualization helped in observing differences in the objects and how we could formulate the features.

Describe features and why they were chosen:

1. Height of the element:
The height of fences and cars are always quite low, compared to buildings and poles.

2. Area of the bounding box:
The area of the bounding box for cars, trees and buildings are separable. Each of these classes has their own recognizable area of bounding box. The area of bounding for fences and poles vary a lot within the class.

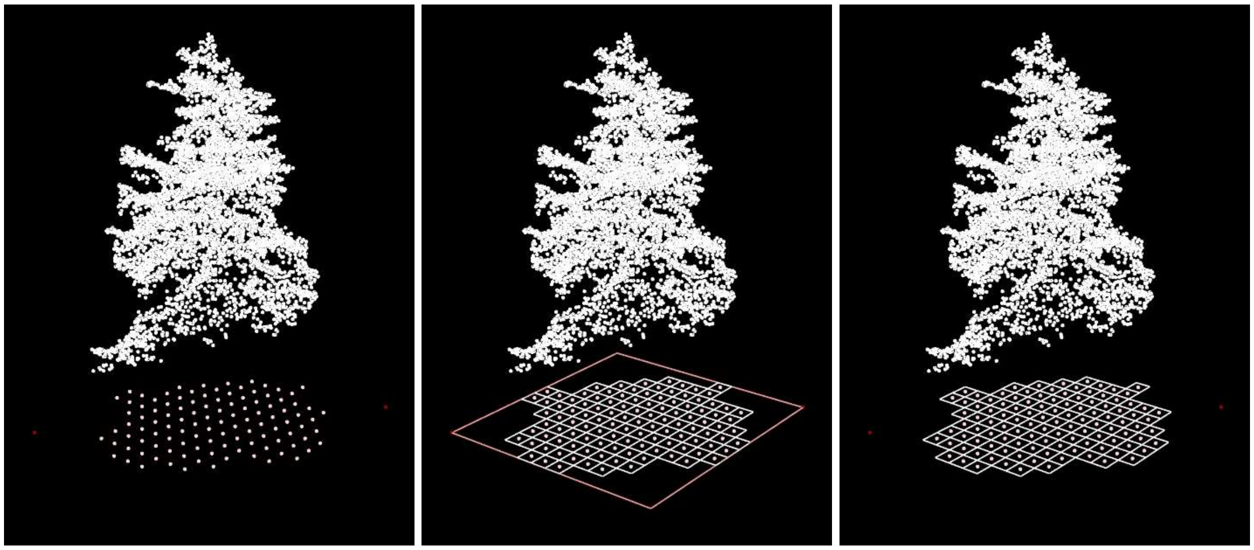
3. Point distribution / Density of the points:
The density of the points within the bounding box i.e., the number of points per m3. Buildings for example have a lot of empty space on the inside of the building, while the point density of trees is way higher.

5. Mean height:
The mean height is used to recognize the height where most of the points in the object's point cloud are located. This is used because trees for example have a lot of points in the parts with a lot of leaves. The mean height is cars is always quite low.

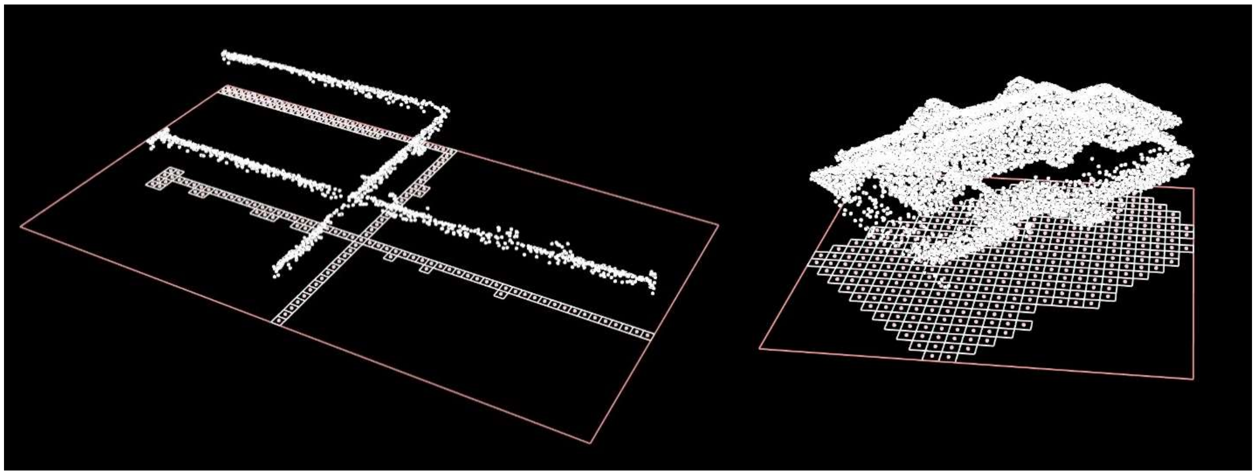
6. Planarity of the object:
Takes the two highest points of the object's point cloud and compares the Eigenvalues of both points to describe the planarity of the object. If the ratio is close to 0 it means that all eigenvalues are equal and thus the surface is planar. This helps in differentiating trees and other non-planar objects from planar objects.

4. Projected area to bounding box area:

This feature uses the projected area of the point cloud. It takes all the points in the point cloud and rounds their x and y components to their nearest integer. This operation essentially clusters all points into 1x1m squares. Counting the numbers of squares results in an estimation of a projected area of the point cloud. This projected area is then divided by the area of the bounding box of the squares, which results in a ratio of how much area of the bounding box is actually occupied by the object.



The ratios for the categories cars and trees don't very much with each other, and almost always cover 95-100% of the bounding box area. It is very effective however to separate fences, which can have a large bounding box, from houses, since houses will occupy a large area of their bounding box, where fences will not.



One improvement that can be made to this feature is first rotating the point cloud to world XY. Otherwise, the bounding box of the point cloud can be bigger than it is in reality.

Model configuration and hyperparameters for SVM and RF:
For this experiment the input labels are separated from output labels. The data is split into test and training data with a 40-60 ratio. The Scikit learn library is used to create an SVM classifier and Random Forest classifier to train the model using X_train and y_train data. The model is fitted to the data and used to predict the value of the classifier denoted as y_hat.

Support vector machine (SVM):
A SVM model finds the hyperplane in high-dimensional space to separates classes. A SVM model can be used for both linear and non-linear classification. The model maximizes the margin between support vectors and decision boundaries to separate classes.

SVM Hyperparameters:
```
svm.SVC(C=1, kernel='rbf', gamma='scale', random_state=42)
```

- C: regularization parameter
- Kernel types: linear, polynomial, Radial Basis Function (RBF)
- Gamma: kernel coefficient for RBF
- Random state:

Random Forest (RF):
- Model configuration
  - Uses the majority vote (classification) or average (regression) of the individual predictions from multiple decision trees
  - Each individual decision tree uses a random subset of the data and features

RF Hyperparameters:
```
RandomForestClassifier(criterion='gini', max_depth=4, n_estimators=100, random_state=42,
oob_score=False)
```

- Criterion
- Maximum depth
- Number of trees
- Maximum features
- Feature importance
- Random_state
- Stratify: If the classes are imbalanced and unevenly distributed then stratify can help by taking equal number of items from each class for training purposes. In this case the classes are equal.

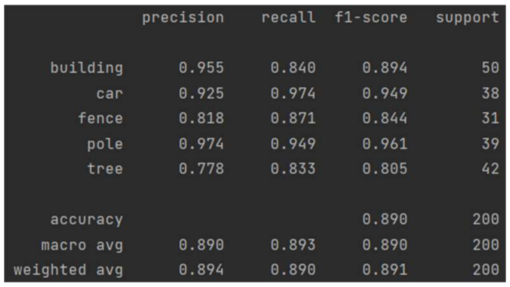
Feature Name	Feature importance in SVM	Feature importance in RF
Height of the element	0.252	0.147
Area of bounding box	0.180	0.187
Density of bounding box	0.066	0.197
Projected area to bounding box area	0.095	0.151
Mean of heights	0.148	0.102
Planarity	0.338	0.211

Experiments and evaluation:

Describe your experiments and classification results. Analyze the effect of hyperparameters and model configuration:

<u>*High precision low recall:*</u>

In SVM model, at one stage the model was extremely careful in assigning incorrect labels for 'tree' and 'fence' as seen in the precision count, but the recall was comparatively low meaning due to the careful nature of the model it was missing a lot of correct labels.

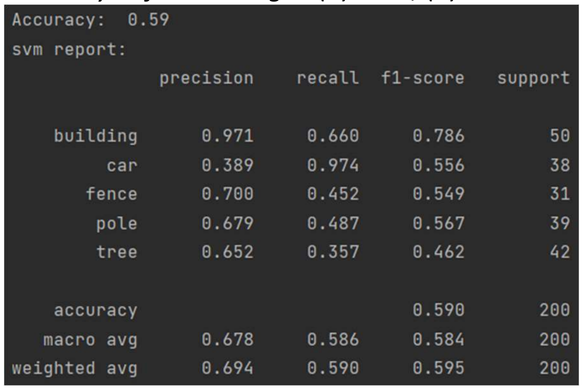


```
              precision    recall  f1-score   support

    building      0.955     0.840     0.894        50
         car      0.925     0.974     0.949        38
       fence      0.818     0.871     0.844        31
        pole      0.974     0.949     0.961        39
        tree      0.778     0.833     0.805        42

    accuracy                          0.890       200
   macro avg      0.890     0.893     0.890       200
weighted avg      0.894     0.890     0.891       200
```

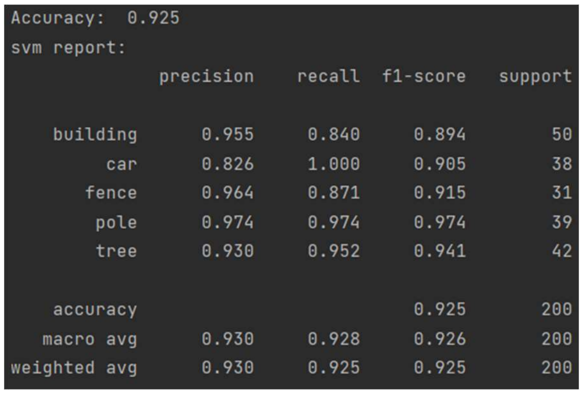
<u>*Scaling the testing and training features:*</u>

Many types of parameters and features were used for the classification model. They had various ranges, units, etc. This resulted in a suboptimal model so all the training and testing labels were scaled before being used in the model. This was especially affecting the SVM model.

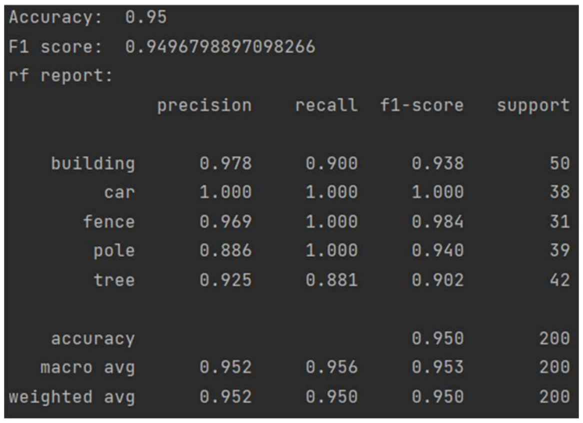
Accuracy before scaling in (1) SVM, (2) Random Forest



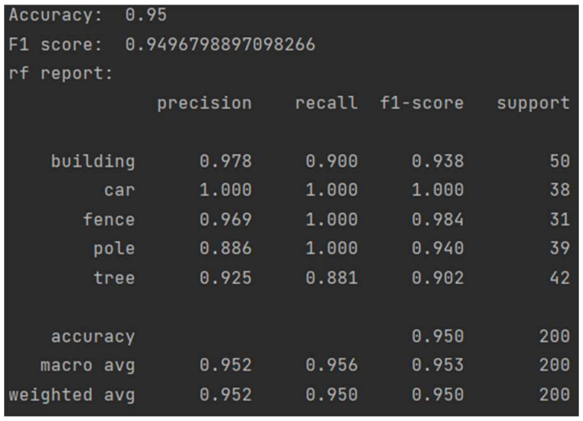
```
Accuracy:  0.59
svm report:
              precision    recall  f1-score   support

    building      0.971     0.660     0.786        50
         car      0.389     0.974     0.556        38
       fence      0.700     0.452     0.549        31
        pole      0.679     0.487     0.567        39
        tree      0.652     0.357     0.462        42

    accuracy                          0.590       200
   macro avg      0.678     0.586     0.584       200
weighted avg      0.694     0.590     0.595       200
```



```
Accuracy:  0.925
svm report:
              precision    recall  f1-score   support

    building      0.955     0.840     0.894        50
         car      0.826     1.000     0.905        38
       fence      0.964     0.871     0.915        31
        pole      0.974     0.974     0.974        39
        tree      0.930     0.952     0.941        42

    accuracy                          0.925       200
   macro avg      0.930     0.928     0.926       200
weighted avg      0.930     0.925     0.925       200
```



```
Accuracy:  0.95
F1 score:  0.9496798897098266
rf report:
              precision    recall  f1-score   support

    building      0.978     0.900     0.938        50
         car      1.000     1.000     1.000        38
       fence      0.969     1.000     0.984        31
        pole      0.886     1.000     0.940        39
        tree      0.925     0.881     0.902        42

    accuracy                          0.950       200
   macro avg      0.952     0.956     0.953       200
weighted avg      0.952     0.950     0.950       200
```



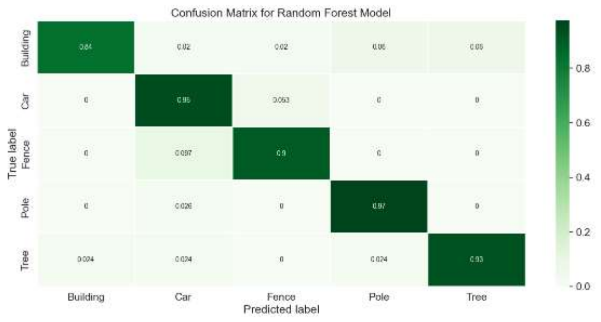
```
Accuracy:  0.95
F1 score:  0.9496798897098266
rf report:
              precision    recall  f1-score   support

    building      0.978     0.900     0.938        50
         car      1.000     1.000     1.000        38
       fence      0.969     1.000     0.984        31
        pole      0.886     1.000     0.940        39
        tree      0.925     0.881     0.902        42

    accuracy                          0.950       200
   macro avg      0.952     0.956     0.953       200
weighted avg      0.952     0.950     0.950       200
```

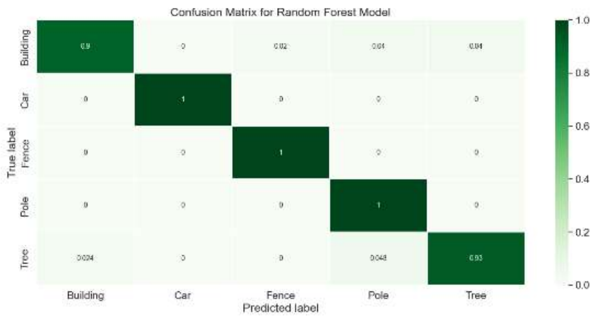
Accuracy after scaling in (1) SVM, (2) Random Forest

<u>Tuning the hyperparameters:</u>

<u>*SVM kernel type:*</u> Changing kernel from 'rbf' to 'poly' with degree 3 or higher gave high inaccuracy in identifying buildings and misclassified several objects as cars. The 'rbf 'kernel was a better match in this case because it projects the non-linearly separable data into higher dimensional space so that it can be separable using a hyperplane.

<u>*Depth of model in random forest:*</u> Different depths have different accuracy rates. The higher the depth, there will be more splits and more information captured. We worked with a range of depths beginning from '2' to '6'. The most effective measure was '5' which was ultimately chosen. (Left: depth=3, Right: depth=5)

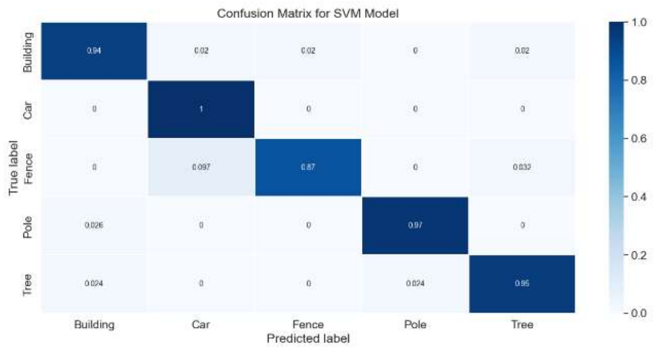


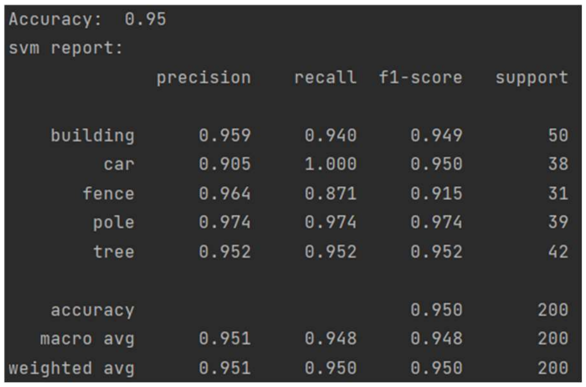


<u>*Max features in random forest:*</u> It refers to the number of features that the tree is allowed to try in individual tree. We changed the feature from 'None' to 'sqrt' which improved the classification of a few classes.

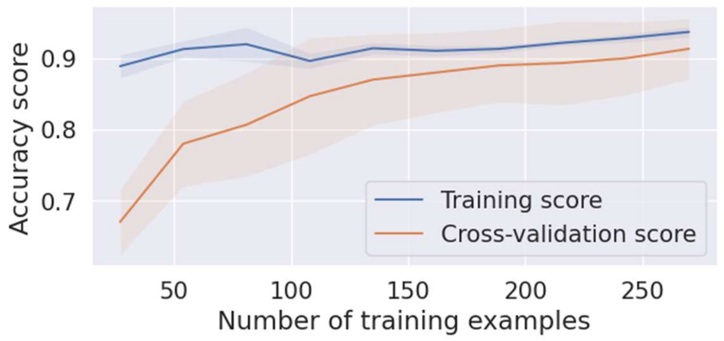
Conclusion:

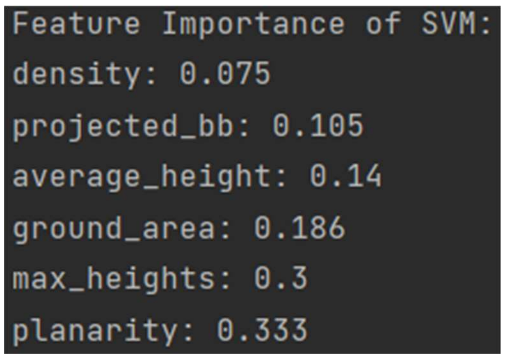
SVM model:





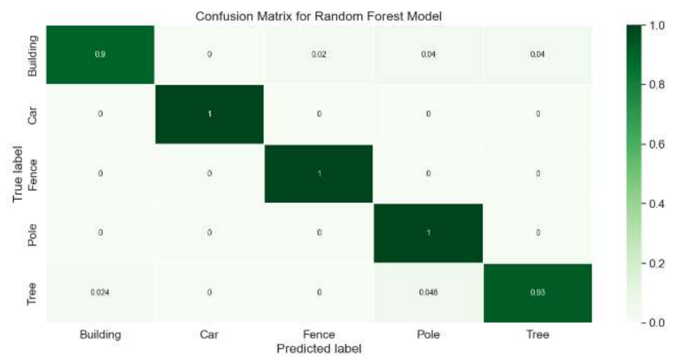
```
Accuracy:  0.95
svm report:
              precision    recall  f1-score   support

    building       0.959     0.940     0.949        50
         car       0.905     1.000     0.950        38
       fence       0.964     0.871     0.915        31
        pole       0.974     0.974     0.974        39
        tree       0.952     0.952     0.952        42

    accuracy                           0.950       200
   macro avg       0.951     0.948     0.948       200
weighted avg       0.951     0.950     0.950       200
```

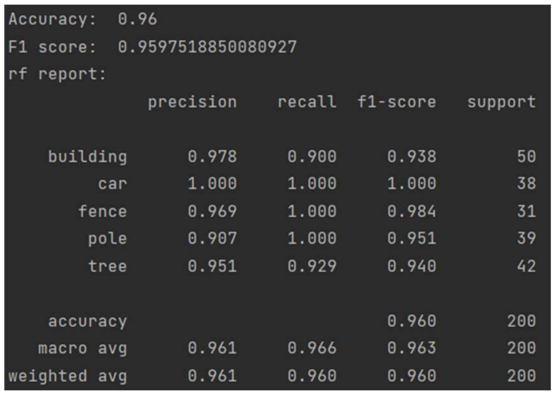




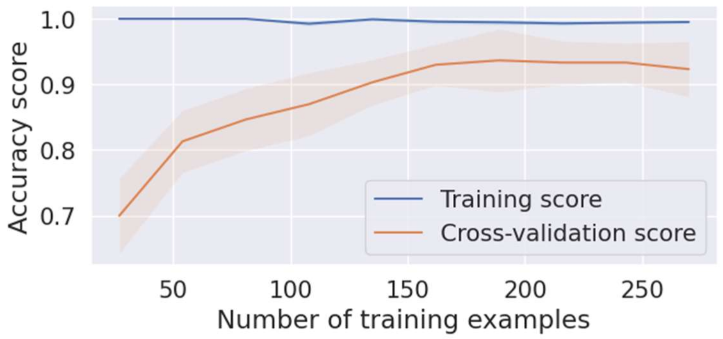
```
Feature Importance of SVM:
density: 0.075
projected_bb: 0.105
average_height: 0.14
ground_area: 0.186
max_heights: 0.3
planarity: 0.333
```

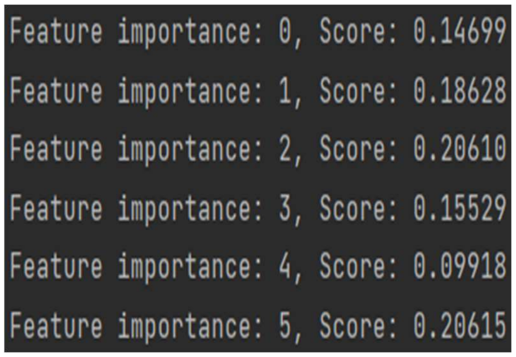
Random Forest model:





```
Accuracy:  0.96
F1 score:  0.9597518850080927
rf report:
              precision    recall  f1-score   support

    building      0.978     0.900     0.938        50
         car      1.000     1.000     1.000        38
       fence      0.969     1.000     0.984        31
        pole      0.907     1.000     0.951        39
        tree      0.951     0.929     0.940        42

    accuracy                          0.960       200
   macro avg      0.961     0.966     0.963       200
weighted avg      0.961     0.960     0.960       200
```





```
Feature importance: 0, Score: 0.14699
Feature importance: 1, Score: 0.18628
Feature importance: 2, Score: 0.20610
Feature importance: 3, Score: 0.15529
Feature importance: 4, Score: 0.09918
Feature importance: 5, Score: 0.20615
```

From the above shown results we can conclude that the random forest model is the best performing model for this situation. One observation we can make is that the two features: density of the points and projected bounding box ration, are not important at all for the SVM, while these features are important features for the random forest.