Title?

Abstract

Urban building detection from remote sensing data poses particular challenges, due to a combination of repetitive and unique geometry and tight building density. This research paper compares three machine learning approaches in the detection of common building features from aerial laser scanning. The study examines the extraction of doors, roofs, and windows. This study uses the DublinCity Data to build predictive models for

M, write about the data set that you trained on and that you validated on.

door versus roof classification and door versus window classification. The models performed with an overall accuracy rate of 92%. Such results allow for greater flexibility in generalizing the model to other urban cities.

* Point cloud data not well established (serves as a guide) Data: discontinuous, non uniform.

Introduction

It is often hard to study building structure, especially in urban and denser cities. Building design and architecture tend to be more complex, and taller buildings and skyscrapers make it harder to study the exterior properties of a building. LiDAR can be easily used to gather information on a building’s design, structure, or surface.

The accurate classification and location of building components such as doors, roofs, and windows will play a role in shaping the future of building inspection in major cities across the world. Building inspection procedures tend to be of high risk since it is done manually [4]. For instance, New York City skyscrapers are inspected by qualified engineers, all while latched onto a rope. Better classification of window, door, and roof location aids in speeding up inspection processes and minimizes risk.

The collection of urban data opens the opportunity of rendering the features in historical buildings. (more detail will be added when I do additional reading)

-In intro note that this topic hasn’t been widely researched (geometry)

Three-dimensional data is being collected at a rapid pace and may be stored in the form of point clouds. However, there has been little investigation about how machine learning techniques may be used to provide better outcomes for feature classification.

The purpose of this project is to gain insight into applying machine learning algorithms to discontinuous, three-dimensional, sparse datasets. Three commonly used machine learning models were used to classify three different objects, roof vs door, and window vs door.

Background

**Scope & Methodology**

M, explain the scope of your work and the parameters you varied. Then explain the methodologies. At the very end, describe the data

**Introduce Models**

This project considered the detection of three common urban objects (roof, door, and window). Three supervised machine learning algorithms (logistic regression [8], SVM [9], and random forests [10]) were used as base models for urban object classification. A normalization technique was applied to each of the models and tested for its effectiveness. Regularization was also applied to all the models and was further tuned to find an optimal model. Since SVM methods also require the use of a kernel function for data transformation, the behavior of four different kernels was investigated.

The results of the logistic regression models were interpreted by their coefficient weights. SVM classifiers are not easily interpretable, so a Random Forest classifier was created to determine the feature importance of each attribute. An SVM classifier runs similarly to a Random Forest classifier and should therefore provide similar accuracies and feature weights. Each model was evaluated using the classifier’s accuracy score and runtime performance.

This section introduces the dataset and the models that were used, followed by a discussion of its evaluation methods.

Dublin City Data:

1. Factual
   1. **Year, high density, area of dataset, total pts**
   2. **What I used of that (I used a tile + number of points)**

An aerial LiDAR scan of Dublin City was conducted by the Urban Modeling Group at University College Dublin in 2015, covering an area of 5.6km2 (including partially covered areas), with a total of 1.4 billion points [3]. Due to the size and high density of the dataset, the densest point cloud tiles were selected to be split into thirteen tiles. The selected dataset has 260 million labeled points. One tile (T\_315500\_234500\_NW) consists of about 50 million points, and points labeled as buildings totaled about 25 million points. Each of the points was pre-labeled into 13 different classes.

At the dataset’s first level, each point was assigned to one of four classes: Ground, Building, Vegetation, or Undefined. This project works extensively with building labeled points; thus, the rest of the dataset may be filtered out. Within the building class, some points, if applicable, were further classified as belonging to a part of a ‘door’, ‘window’, or ‘roof’ structure. Figure 1 gives a breakdown of the point classification.

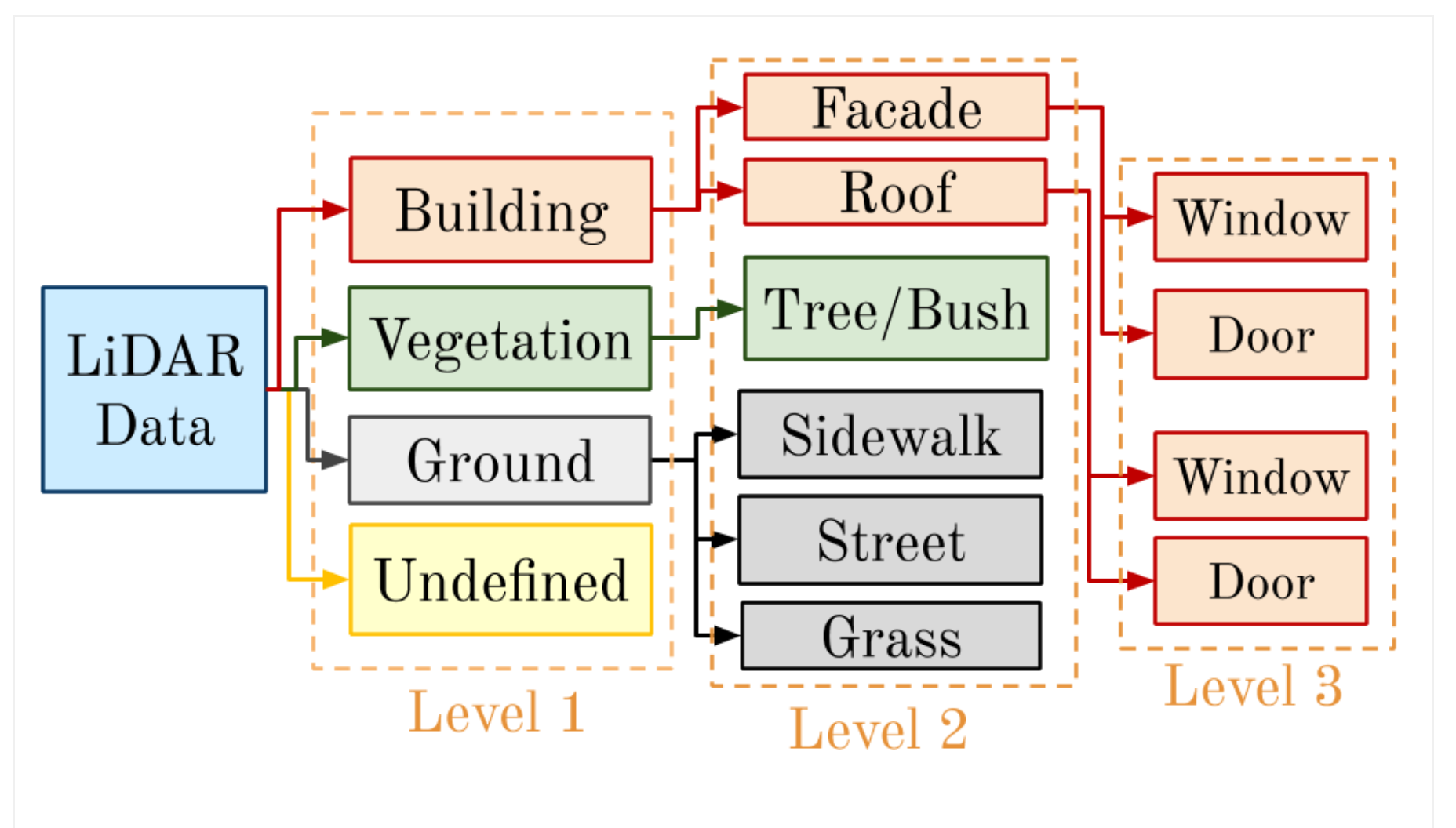


Figure 1: LiDAR Data Breakdown by level of classification [3]

As shown in Figure 1, buildings were further classified by being a ‘Facade’ or a ‘Roof’ in Level 2. Within these two subclasses, points that are classified as a ‘Facade’ may be categorized as ‘Window’ or ‘Door’. Similar to facades, roofs may also have windows or doors. Such points are also labeled in the dataset. In this project, windows and doors located on roofs will be given the ‘roof’ classification.

* 1. **Tile was labeled**

Given the relatively large size of the DublinCity Dataset, one selected tile was imported into CloudCompare. Building features had to be individually selected and saved as CSV files prior to analysis. An additional column, ‘Type’, was added to the end of each file to determine whether a point was classified as a ‘door’, ‘roof’, or ‘window’. Eleven different attributes were used in our analysis to predict point classes. The attributes are listed in Figure 2 below.

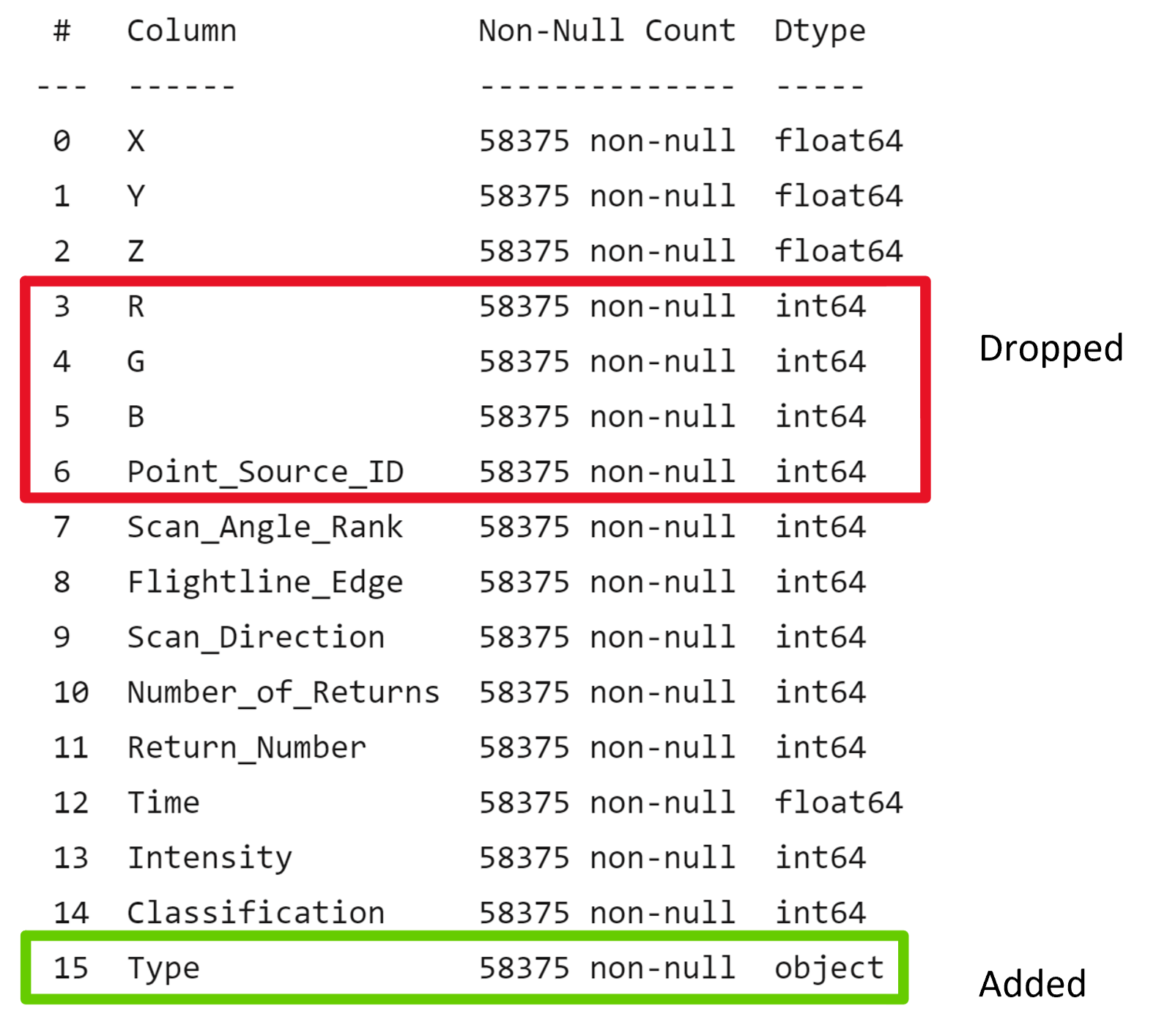


Figure 2: The ‘RGB’ columns were dropped from the dataset and replaced with an additional column, ‘Type’

* 1. **Show a pic of what it looks like (shown below)**

As shown in CloudCompare [7], building roofs are identified by an indigo color, windows are given an aqua color, and doors are identified by a magenta color. Points that are not classified as a door, window, or roof are shown in purple. For illustrative purposes, the color of the doors was changed to yellow in Figure **3**.

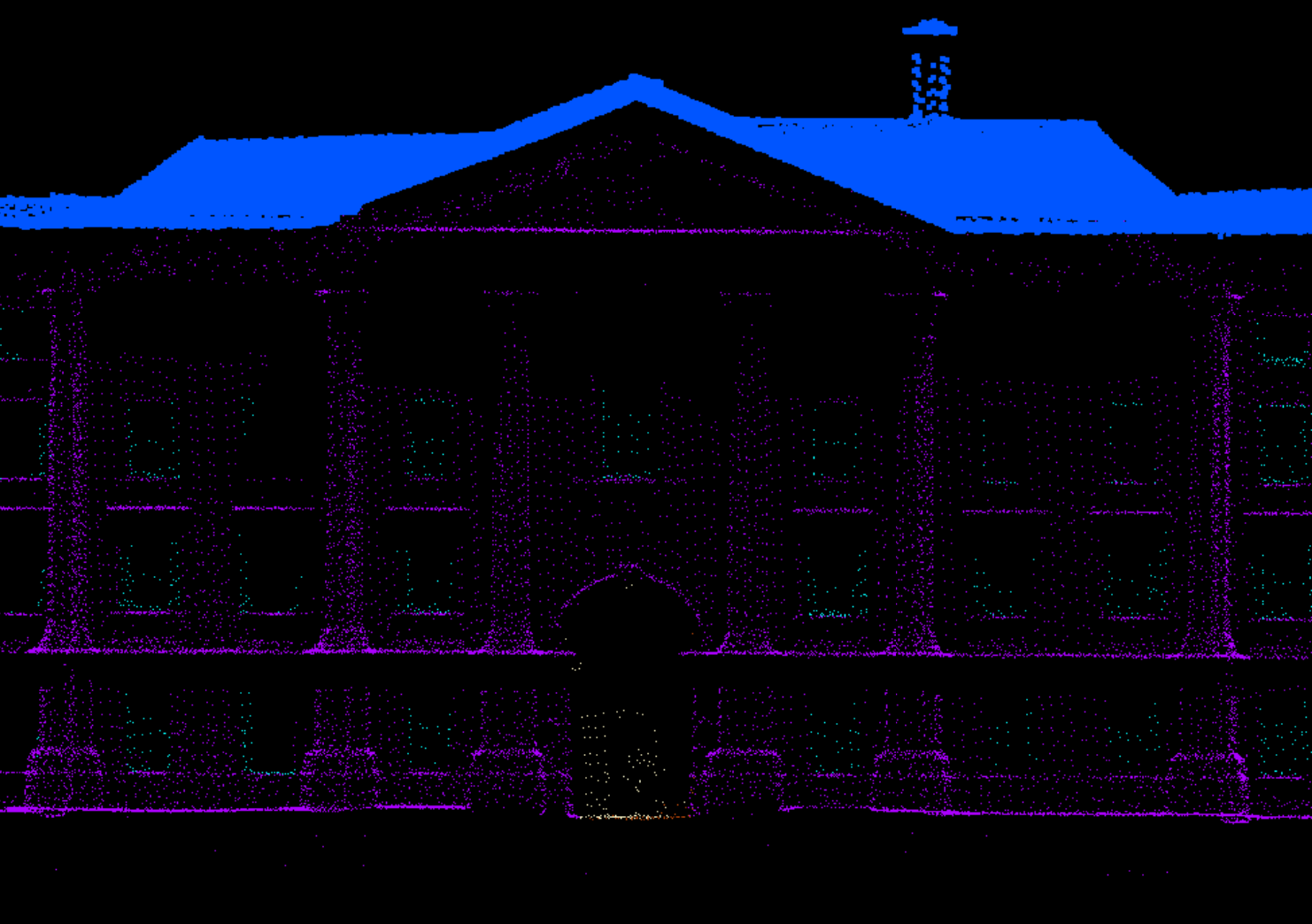


Figure 3: Sample building as shown in CloudCompare

* 1. **Preprocessing (split/resample)**

Naturally, some building classifiers, such as doors may have fewer data points than other building classifiers, such as roofs. In general, buildings have more roof surface area than door surface area. Most of the data would be comprised of roof records, so, additional door records had to be added to the dataset in order to equalize the amount of roof and door data points. Labeled data from CSV files were merged into one dataset and imported into Python for analysis. In Python, the ‘R,’ ‘G,’ ‘B,’ and ‘Point\_Source\_ID’ columns were dropped since the RGB values were replaced by a ‘Type’ column for identification. The Point\_Source\_ID number plays an insignificant role in having any correlation with classification. Note that the data remains as a three-dimensional point cloud after preprocessing phase.

Prior to building the models, downsampling with **random selection** was applied to equalize the number of points belonging to each type. Each type was downsampled to 6,000 points. Each model had a total of 12,000 points that were split into train and test groups. 75% of the 12,000 points were used to train the model, and the remaining 25% of the points were assigned to test model performance as shown in Table 1. An additional building’s data, independent of the train and test sets, was used to check if the model generalized well with unseen data.

|  |  |
| --- | --- |
| Number of Training Samples | Number of Testing Samples |
| 9,000 | 3,000 |

Table 1: Number of samples used to train/test each model

Moderate amounts of ridge regularization were added to each of the models using c-values that ranged from 0.001 to 100. A c-value that is less than 0.001 may start to underfit the data and focus on minimizing the coefficient weights for each attribute. A larger c-value attributes to overfitting and less generalization of values.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Smallest c-value |  |  |  |  | Largest c-value |
| 0.001 | 0.01 | 0.1 | 1 | 10 | 100 |

Table 2: c-values used for regularization

1. **Methodology** 
   1. **Describe Models**

Models were created using Logistic Regression, SVMs, and Random Forests [8,9,10] to differentiate Roof vs Door and Window vs Door points. The goal of the models is to predict the ‘Type’ of the data point with the best possible accuracy using a combination of 11 attributes. The models were tuned and improved through normalization and regularization techniques. Four transformation techniques were also applied to the SVM model that performs best, either from normalization or not normalizing our data. The kernel functions that were used in the SVM models were Linear, Polynomial, RBF (Radial Basis Function), and Sigmoid.

Table 3 shows each of the techniques grouped by case. Logistic Regression models are all classified by Case 1, SVM models are classified by Case 2, and Random Forest models are classified by Case 3. The letter that proceeds the case number describes if normalization was applied to the data. An ‘a’ symbolizes that the data was normalized, while a ‘b’ symbolizes that the data was not normalized. Regularization was applied to all models.

|  |  |  |  |
| --- | --- | --- | --- |
| Case | Techniques | Normalization | Regularization |
| 1a | Logistic Regression | No | Yes |
| 1b | Logistic Regression | Yes | Yes |
| 2a | SVM | No | Yes |
| 2b | SVM | Yes | Yes |
| 3a | Random Forest | No | Yes |
| 3b | Random Forest | Yes | Yes |

Table 3: List of models created

Since transformations were applied to SVM models, Table 4 illustrates all potential SVM models that may be considered. If normalizing the data produces better accuracies, all the cases labeled with ‘2a’ will be built. If not, then all the cases labeled with ‘2b’ will be built.

|  |  |  |  |
| --- | --- | --- | --- |
| Case | Normalization | Regularization | Transformation |
| 2a-linear | Yes | Yes | Linear |
| 2a-poly | Yes | Yes | Polynomial |
| 2a-RBF | Yes | Yes | RBF |
| 2a-sigmoid | Yes | Yes | Sigmoid |
| 2b-linear | No | Yes | Linear |
| 2b-poly | No | Yes | Polynomial |
| 2b-RBF | No | Yes | RBF |
| 2b-sigmoid | No | Yes | Sigmoid |

Table 4: List of all SVM models with normalization and transformation techniques.

* 1. **Evaluation of models**

The effect of normalization was measured by comparing two accuracy scores (see below) of the model, one without regularization and one with regularization. If normalization had a positive effect on the accuracy of the results on a specific machine learning algorithm, the data would be normalized before building a model that follows that specific algorithm. If normalization did not have a positive effect, then the data would remain unchanged.

The optimal value for regularization was chosen based on the model’s generalization accuracy. The c-value that performed the best on the generalized test set would be chosen as the optimal value. There may be cases where varying c-values would not change the overall performance of the model. In such cases, the c-value that would be chosen to be the optimal value would be c = 1 (default). Confusion matrices were created to determine how the model performed while classifying between building object types.

Each model’s accuracy and runtimes were noted. **The accuracy score was calculated in the Numpy library [11], which returns the number resulting from counting the total number of predictions over the total number of predictions made**. The total run time of the program was also taken into consideration when evaluating a model. The total run time of a model includes the time it takes for a model to fit a set of data points to a classifier and the amount of time it takes to predict a given number of points. The best overall models were selected from the obtained results.

**Results**

**For each algorithm:**

**Roof vs Door:**

* **Logistic Regression**

A standard logistic regression model was built using Python’s scikit-learn library[8].

* + **Normalization Results (accuracy+confusion Matrix)**

Two models were created, one with a normalization technique to normalize each feature to have a mean 0 and a standard deviation of 1 and the other without normalization. By default, Python applies regularization to the model, where c = 1.

Without normalization, the accuracy value was noted to be 0.987. This value is high, but the model does not perform as expected since it predicts ‘roof’ for each data point. Since there are about 78 times more roof points than door points, the accuracy score value may appear to be high due to this behavior. Normalizing the data yielded better results since the model predicts both classes, with an overall accuracy rate of 0.616.

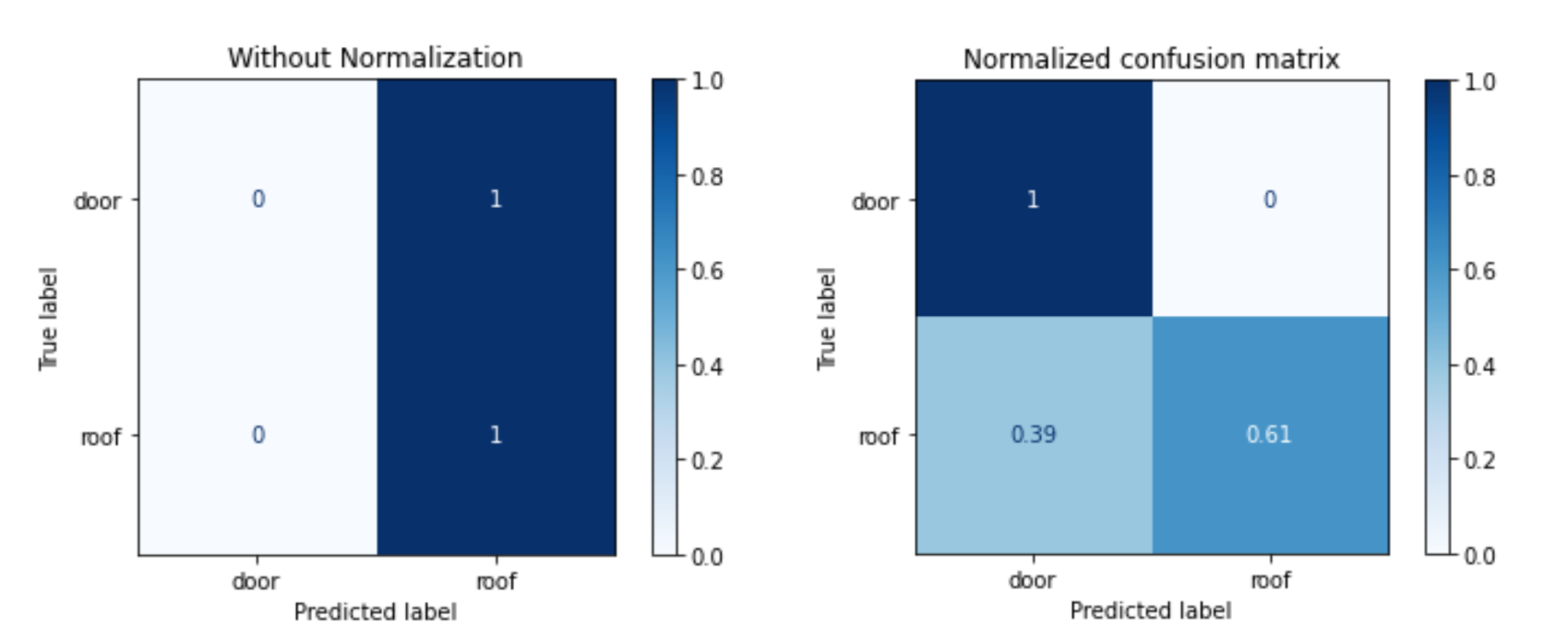


Figure 4: Confusion matrices for non-normalized vs normalized data for Logistic Regression models

* + **Regularization results**

As discussed in the methodology section, moderate amounts of regularization were added to each of the models using c-values that ranged from 0.001 to 100. The resulting accuracies from the test set, along with their corresponding c-values are shown in Table 5.

|  |  |
| --- | --- |
| c-value | Accuracy rate |
| 0.001 | 0.573 |
| 0.01 | 0.587 |
| 0.1 | 0.602 |
| 1 | 0.615 |
| 10 | 0.625 |
| 100 | 0.632 |

Table 5: Performance of a Logistic Regression model with normalization and varying degrees of regularization

Since a c-value of 100 was able to produce the best accuracy results, it was chosen to be most suitable for the Logistic Regression model. An accuracy rate of 0.631966 was achieved.

* **SVM**

A standard Support Vector Machine (SVM) model was built using Python’s scikit-learn library [9].

* + **Normalization Results (accuracy+confusion Matrix)**

All feature columns were normalized to have a mean of 0 and a standard deviation of 1. With normalization, an accuracy of 0.601 was achieved using the polynomial kernel, compared to an accuracy of 0.893 without normalization. It was determined that normalization would be used in all future models.

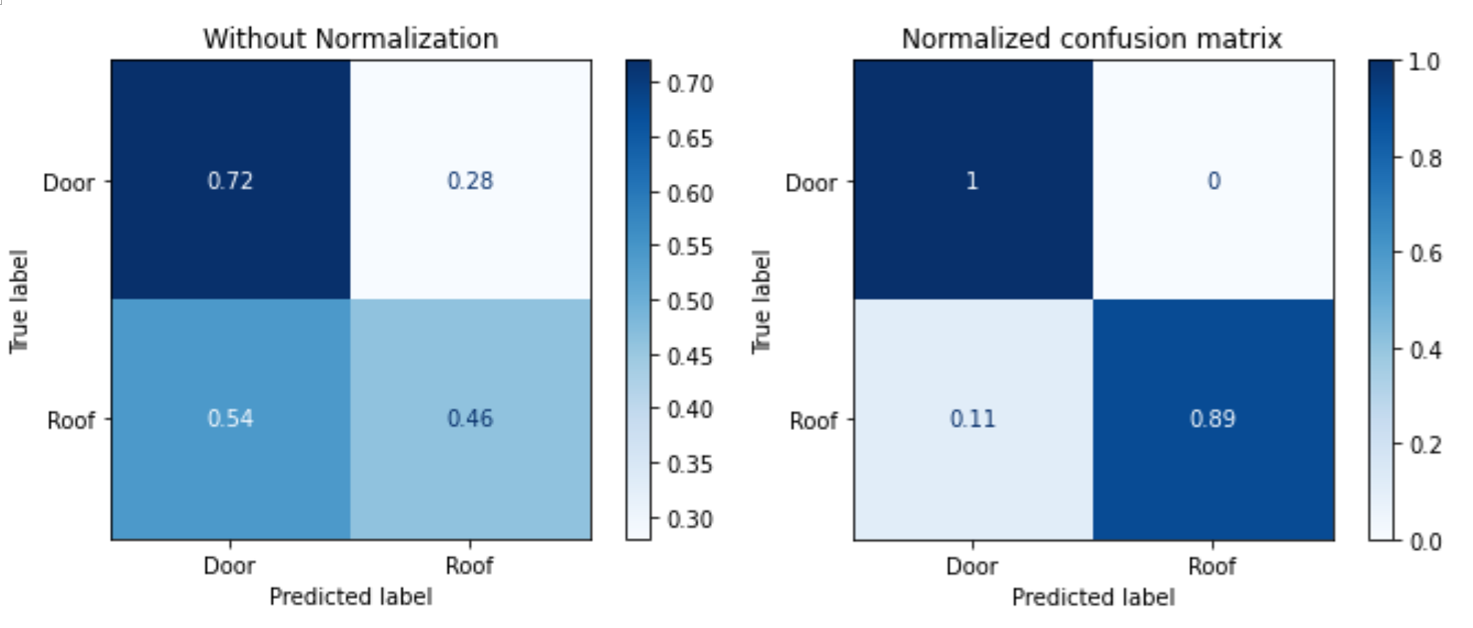


Figure 5: Confusion matrices for non-normalized vs normalized data for SVM models

* + **Regularization results**

Different values of regularization were applied to an SVM model with normalized data. The results are shown in Table 6.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| c-value | Accuracy rate – Linear | Accuracy rate - Polynomial | Accuracy rate - RBF | Accuracy rate - Sigmoid |
| 0.001 |  | 0.669 |  |  |
| 0.01 | 0.734 | 0.776 | 0.708 | 0.589 |
| 0.1 | 0.717 | 0.869 | 0.744 | 0.576 |
| 1 | 0.717 | 0.895 | 0.761 | 0.678 |
| 10 | 0.717 | 0.900 | 0.762 | 0.801 |
| 100 | 0.717 | 0.900 | 0.762 | 0.538 |

Table 6: Performance of an SVM model with normalization and varying degrees of regularization

The accuracy rate reaches its best value at c = 10 with the polynomial kernel. After this point, the accuracy rate does not increase any further as the c-value is increased, so c = 10 is chosen to be the optimal hyperparameter for this model. An accuracy rate of 0.900253

* **Random Forest** 
  + **Normalization Results (accuracy+confusion Matrix)**

As shown in the confusion matrices below, the model performs better when the data is normalized. If the data is not normalized prior to training a Random Forest classifier, all samples will be classified as ‘roof’. However, once the data is normalized, the model successfully predicts both classes.

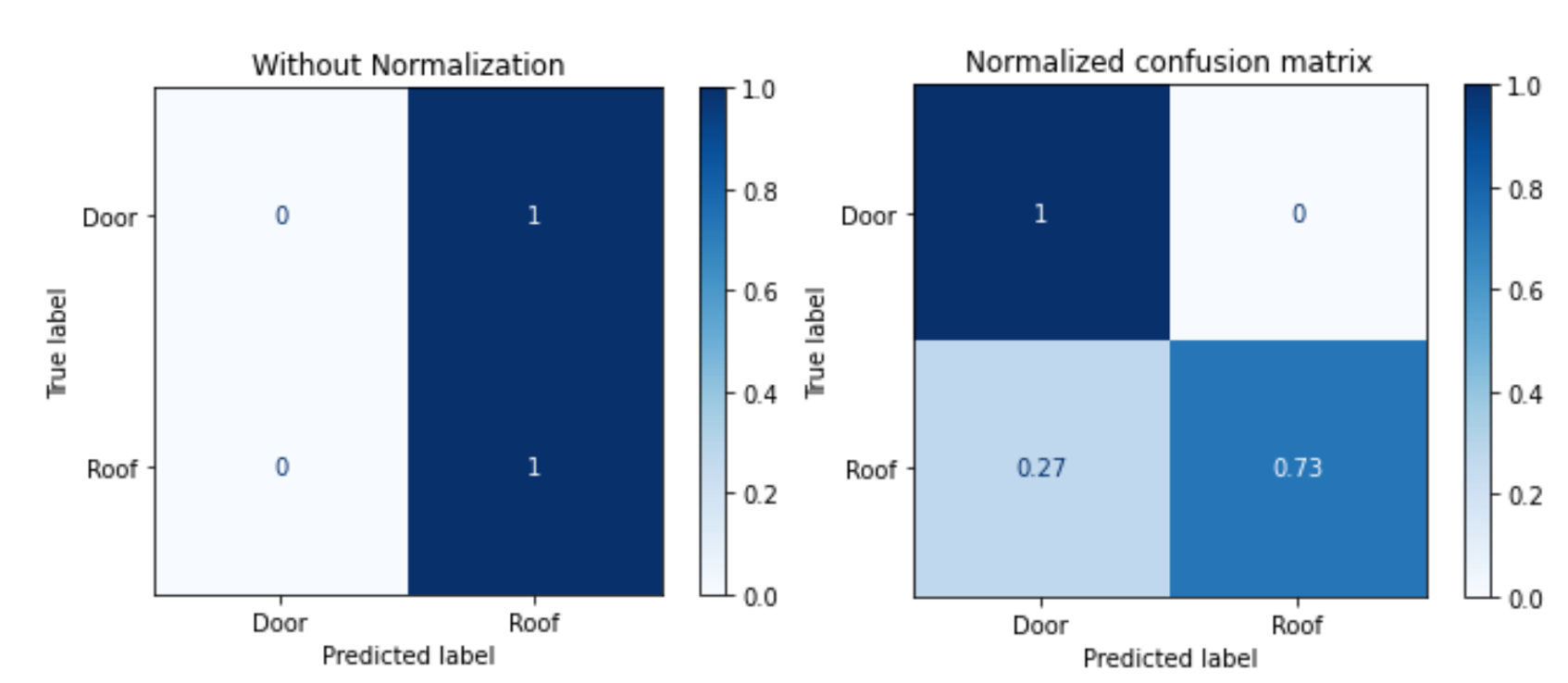


Figure 6:

* + **Regularization results**

Unlike the logistic regression and SVM models, RandomizedSearch was performed on Random Forest models to optimize hyperparameters. The RandomizedSearch algorithm chooses random combinations of parameters from a user-defined parameter grid and outputs the best performing parameters.



Figure 6: Performance of a Random Forest model with normalization and varying degrees of regularization [12]

The parameter n\_estimators, or the number of trees in the forest, was set to a number between 10 and 200 for best practices. Limiting the number of trees in a forest reduces runtime and helps with reducing model complexity. max\_depth explains the maximum depth of a tree in the random forest.

The best performing hyperparameters (as found from RandomizedSearch) were determined to have 56 trees, with a maximum tree depth of 14. The feature importance for each attribute is shown in Figure 7. The feature importance signifies how much contribution an attribute carries to determining the classification of an object. As noted in the chart,

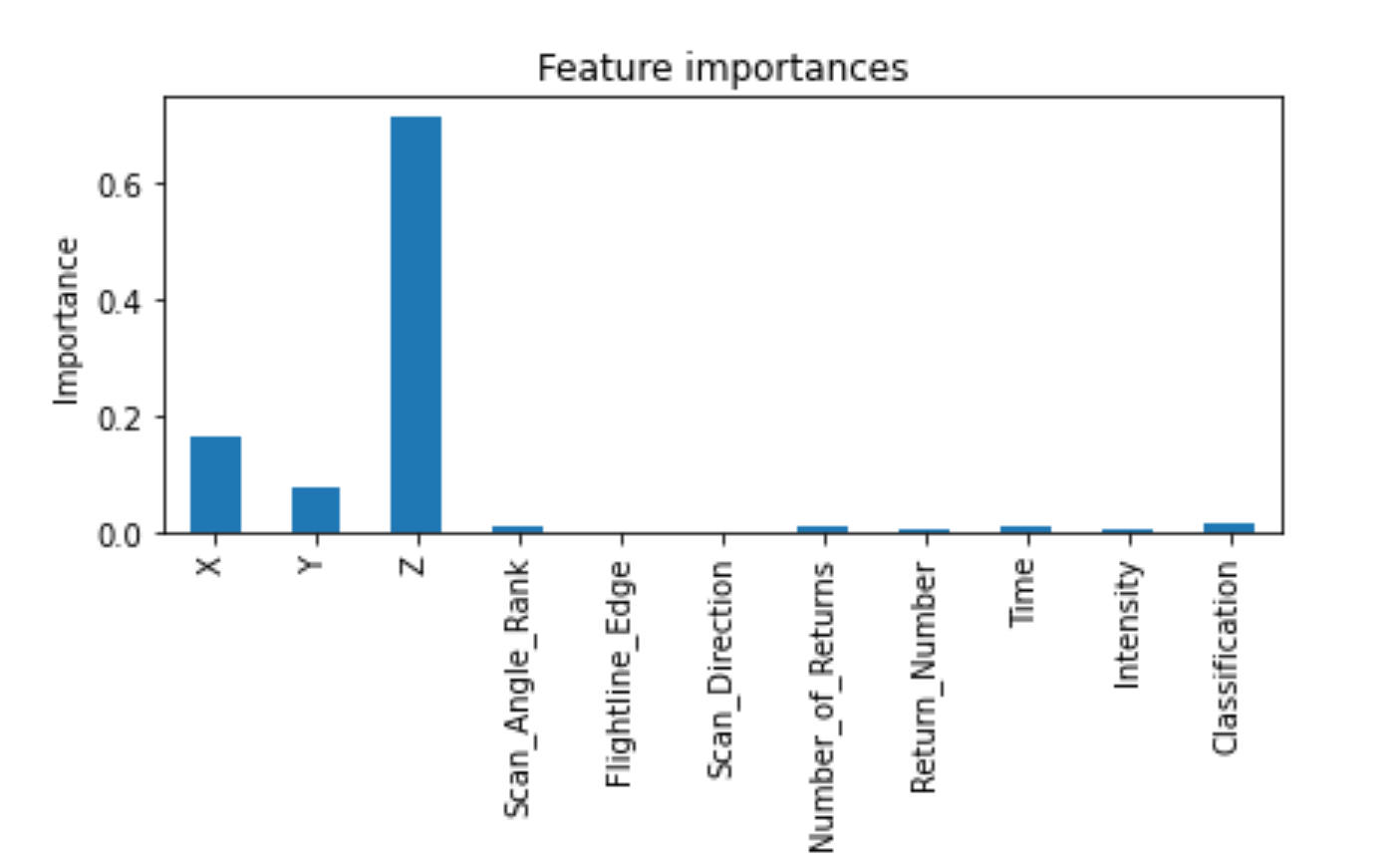


Figure 7: Random Forest feature importance of attributes for classifying Doors vs Roofs

**Door vs Window**

* **Logistic Regression** 
  + **Normalization Results**

The logistic regression model performs better when normalization is added to the data. As seen in Figure 8, the model predicts only one ‘door’ for each record that is fed into the model. With normalization, this model achieved an accuracy of 0.971 while predicting both ‘door’ and ‘window’ classes. From these results, normalization should be used in future models.

Chart

Description automatically generated

Figure 8: Confusion Matrices of Normalized vs Non-Normalized Logistic Regression models for predicting ‘door’ vs ‘window’

* + **Regularization Results**

Moderate amounts of regularization were added to the logistic regression model. The results are shown in Table 7. All c-values greater than 1 led to the same accuracy rate of 0.973, so the optimal c-value in this scenario would be c = 1.

|  |  |
| --- | --- |
| c-value | Accuracy rate – Linear Kernel |
| 0.001 | 0.950 |
| 0.01 | 0.964 |
| 0.1 | 0.971 |
| 1 | 0.973 |
| 10 | 0.973 |
| 100 | 0.973 |

Table 7: Accuracy rates of using a logistic regression model with varying degrees of regularization.

* **SVM** 
  + **Normalization**

As shown by the confusion matrices in Figure 9, normalization is able to help scale the data such that both ‘door’ and ‘window’ classes will be predicted. Since normalization is able to help improve model performance, all SVM models will incorporate the use of normalization.

Chart

Description automatically generated

Figure 9: Confusion matrices for an SVM with a linear kernel to classify doors and windows.

* + **Regularization**

The ideal c-value that should be used for the SVM model is c = 1, which presents an accuracy rate of 0.975. We see that all c-values larger than 1 produce the same accuracy scores.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| c-value | Accuracy rate – Linear | Accuracy rate - Polynomial | Accuracy rate - RBF | Accuracy rate - Sigmoid |
| 0.001 | 0.947 | 0.893 |  |  |
| 0.01 | 0.970 | 0.936 | 0.823 | 0.895 |
| 0.1 | 0.974 | 0.908 | 0.793 | 0.764 |
| 1 | 0.975 | 0.933 | 0.792 | 0.740 |
| 10 | 0.975 | 0.906 | 0.768 | 0.738 |
| 100 | 0.975 | 0.858 | 0.744 | 0.738 |

Table 8: Regularization results for SVM with a linear kernel

* **Random Forest** 
  + **Normalization**

Normalization was applied to the data and had a positive outcome on model effectiveness. After normalization was applied, Chart, waterfall chart

Description automatically generated

* + **Regularization**

Randomized search was used to carry out regularization for random forests. Figure 10 shows the hyperparameter grid that was used to train the random forest.

Text

Description automatically generated

Figure 10: Parameter values that were used to train a random forest to identify doors and windows.

The optimal values as determined by the output of RandomizedSearch had 21 trees and a max tree depth of 10. The importance of each attribute was found using the RandomForest feature importance method. The z value contributes most of the results of this model as shown in Figure 11. Using these parameters resulted in an accuracy of 0.934.

Chart, waterfall chart

Description automatically generated

Figure 11: Feature importance for classifying doors and windows by Random Forests

Discussion

From the results above, normalizing the data in every case helped the model perform better. It can be noted that each model without normalization predicted only one class, instead of two. SVMs performed better in identifying all building features. While logistic regression models tend to run faster than SVM models, adding an additional 2000 samples to the logistic regression models still results in an SVM model with a smaller number of training examples. In the Door vs Window models, the accuracy rate is 0.973 with a logistic regression model, compared to 0.975 with SVMs. As seen in this example, logistic regression may work accurately and efficiently with large and varied point cloud data sets.

Several limitations that were encountered in this study were the lack of high computing power for running larger datasets. The Dublin City point cloud data was dense, so only one tile was able to be processed for analysis. While the models have been performing fairly well in generalizing to buildings within the same tile, the same cannot be concluded about other tiles or other urban buildings.

Conclusion

Acknowledgments:

I would like to thank NYU Tandon’s Vertically Integrated Project team for funding this research project. I would also like to express my gratitude to the staff at NYU’s Center of Urban Science and Progress, especially Professor Debra Laefer and Michael Stanley for their advisement and guidance throughout this project.

References:

[1] Debra F Laefer, Saleh Abuwarda, Anh-Vu Vo, Linh Truong-Hong, and Hamid Gharibi. 2015 aerial laser and photogrammetry survey of dublin city collection record. [doi:10.17609/N8MQ0N](https://doi.org/10.17609/N8MQ0N)

[2] S M Iman Zolanvari, Susana Ruano, Aakanksha Rana, Alan Cummins, Rogerio Eduardo da Silva, Morteza Rahbar, Aljosa Smolic. 2019 DublinCity: Annotated LiDAR Point Cloud and its Applications. 30th BMVC, September 2019.

[3] https://v-sense.scss.tcd.ie/dublincity/

[4] Wu Y, Blunden LS, Bahaj AS. City-wide building height determination using light detection and ranging data. Environment and Planning B: Urban Analytics and City Science. 2019;46(9):1741-1755. doi:10.1177/2399808318774336

[5] [Zheng, Z J](https://www.proquest.com/indexinglinkhandler/sng/au/Zheng,+Z+J/$N;jsessionid=980A24A64354494BBB54B0EBF71FB354.i-01aa74dda7839d5cc); [Pan, M](https://www.proquest.com/indexinglinkhandler/sng/au/Pan,+M/$N;jsessionid=980A24A64354494BBB54B0EBF71FB354.i-01aa74dda7839d5cc); [Pan, W](https://www.proquest.com/indexinglinkhandler/sng/au/Pan,+W/$N;jsessionid=980A24A64354494BBB54B0EBF71FB354.i-01aa74dda7839d5cc).[**ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction**](https://www.proquest.com/pubidlinkhandler/sng/pubtitle/ISARC.+Proceedings+of+the+International+Symposium+on+Automation+and+Robotics+in+Construction/$N/1646340/OpenView/2526369993/$B/23A68BE4F6DC44CEPQ/1;jsessionid=980A24A64354494BBB54B0EBF71FB354.i-01aa74dda7839d5cc)**; Waterloo** Vol. 37, : 16-23. Waterloo: IAARC Publications. (2020)

[6] <https://www.nbcnewyork.com/investigations/an-inside-look-at-how-nycs-dept-of-buildings-inspects-facades-to-ensure-safety/2293595/> (needs to be properly cited)

[7] <https://www.danielgm.net/cc/> (cloud compare, needs to be properly cited)

[8] <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html> (sci-kit learn for Logistic Regression, needs to be properly cited)

[9] <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html> (sci-kit learn for SVMs, needs to be properly cited)

[10] <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html> (sci-kit learn for RFs, needs to be properly cited)

[11] <https://numpy.org/doc/> (Numpy documentation)

[12] <https://github.com/WillKoehrsen/Machine-Learning-Projects/blob/master/Random%20Forest%20Tutorial.ipynb>