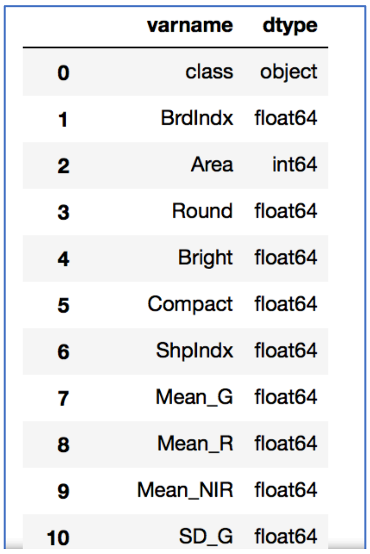
We have implemented this project on Python and its associated packages like Pandas, NumPy, Matplotlib and Sklearn. We upload it as below https://github.com/an46/UIUC_MLTeamProjects_SupervisedLearning.
Imported the datasets (Test and train). Unlike the usual scenario, train set was smaller than the test set.

1. Data cleaning:

Analysis to find if there are any duplicate column or missing value. It showed none

2. Data Exploration:

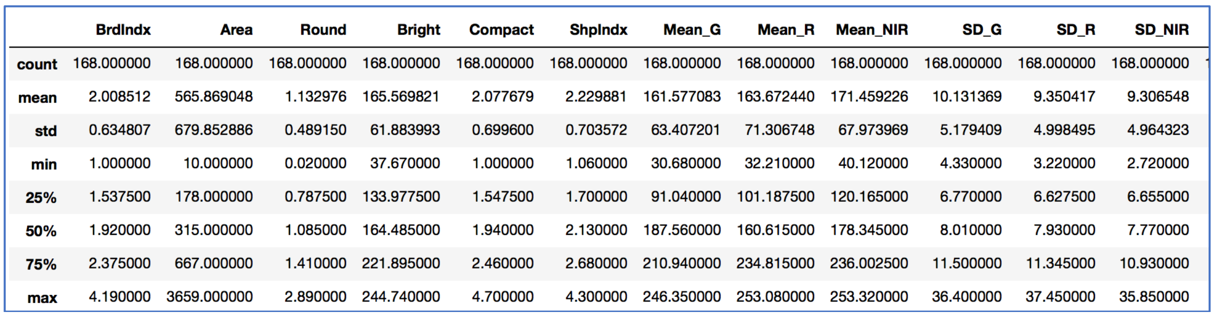
Columns and their respective data types were analyzed. There are three type of data, int, float and object(label).



	varname	dtype
0	class	object
1	BrdIndx	float64
2	Area	int64
3	Round	float64
4	Bright	float64
5	Compact	float64
6	ShpIndx	float64
7	Mean_G	float64
8	Mean_R	float64
9	Mean_NIR	float64
10	SD_G	float64

Figure 2.1 data type of features

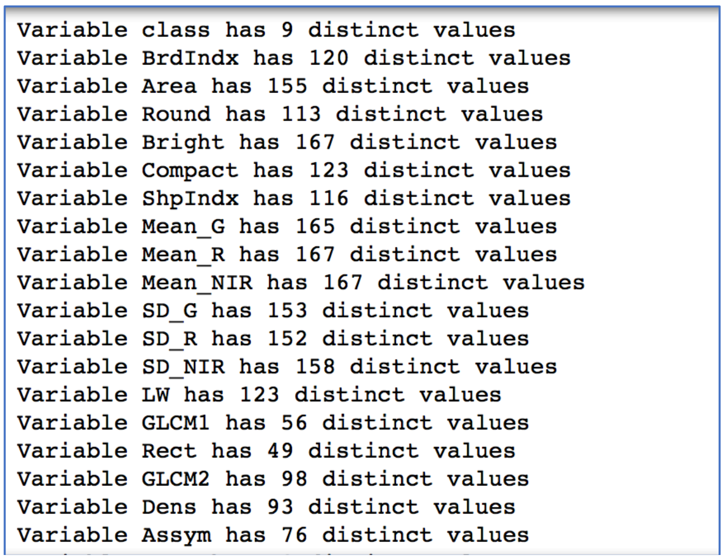
Distribution of data



	BrdIndx	Area	Round	Bright	Compact	ShpIndx	Mean_G	Mean_R	Mean_NIR	SD_G	SD_R	SD_NIR
count	168.000000	168.000000	168.000000	168.000000	168.000000	168.000000	168.000000	168.000000	168.000000	168.000000	168.000000	168.000000
mean	2.008512	565.869048	1.132976	165.569821	2.077679	2.229881	161.577083	163.672440	171.459226	10.131369	9.350417	9.306548
std	0.634807	679.852886	0.489150	61.883993	0.699600	0.703572	63.407201	71.306748	67.973969	5.179409	4.998495	4.964323
min	1.000000	10.000000	0.020000	37.670000	1.000000	1.060000	30.680000	32.210000	40.120000	4.330000	3.220000	2.720000
25%	1.537500	178.000000	0.787500	133.977500	1.547500	1.700000	91.040000	101.187500	120.165000	6.770000	6.627500	6.655000
50%	1.920000	315.000000	1.085000	164.485000	1.940000	2.130000	187.560000	160.615000	178.345000	8.010000	7.930000	7.770000
75%	2.375000	667.000000	1.410000	221.895000	2.460000	2.680000	210.940000	234.815000	236.002500	11.500000	11.345000	10.930000
max	4.190000	3659.000000	2.890000	244.740000	4.700000	4.300000	246.350000	253.080000	253.320000	36.400000	37.450000	35.850000

Figure 2.2 the measurement of features

From this table, we identified some feature that had low variation. In general features with low variation contribute less to the prediction of the target variable.



```
Variable class has 9 distinct values
Variable BrdIndx has 120 distinct values
Variable Area has 155 distinct values
Variable Round has 113 distinct values
Variable Bright has 167 distinct values
Variable Compact has 123 distinct values
Variable ShpIndx has 116 distinct values
Variable Mean_G has 165 distinct values
Variable Mean_R has 167 distinct values
Variable Mean_NIR has 167 distinct values
Variable SD_G has 153 distinct values
Variable SD_R has 152 distinct values
Variable SD_NIR has 158 distinct values
Variable LW has 123 distinct values
Variable GLCM1 has 56 distinct values
Variable Rect has 49 distinct values
Variable GLCM2 has 98 distinct values
Variable Dens has 93 distinct values
Variable Assym has 76 distinct values
```

Figure 2.3 distinct values of each features

From figure2.3 above, it shows that all variables have many distinct values.

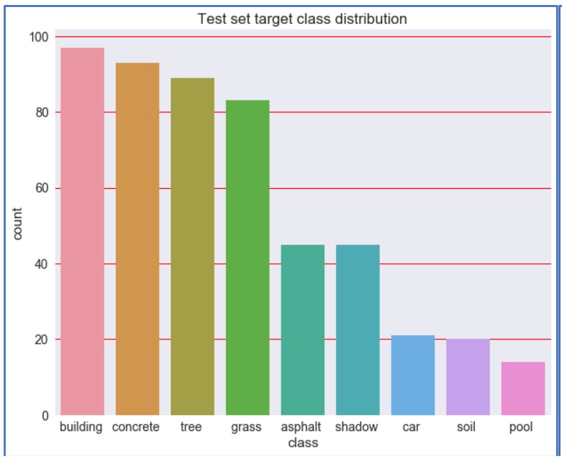


Figure 2.4 the distribution of label in test set

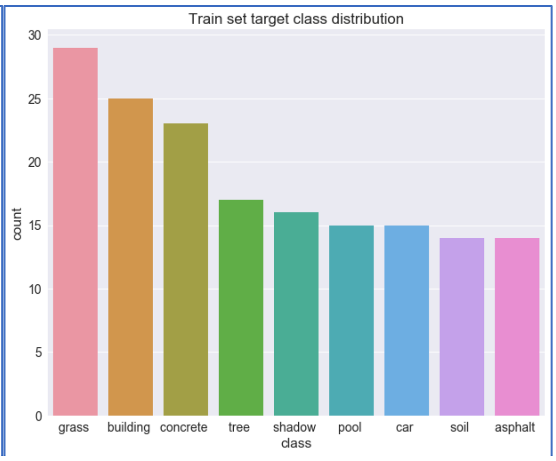


Figure 2.5 the distribution of label in train data

From figure 2.4-2.5, we can get the distribution of class in train and test sets.

3. Feature Selection

We remove one feature that has high correlation (0.9) with another. Then we got 77 features. It was one of the methods which we tried.

4. Fit Model

○ **Random Forest:**

We did Kfold cross-validation (10 fold) to the train data and then fit it into random forest model.

We extracted feature importance from it that can be seen from the below figure.

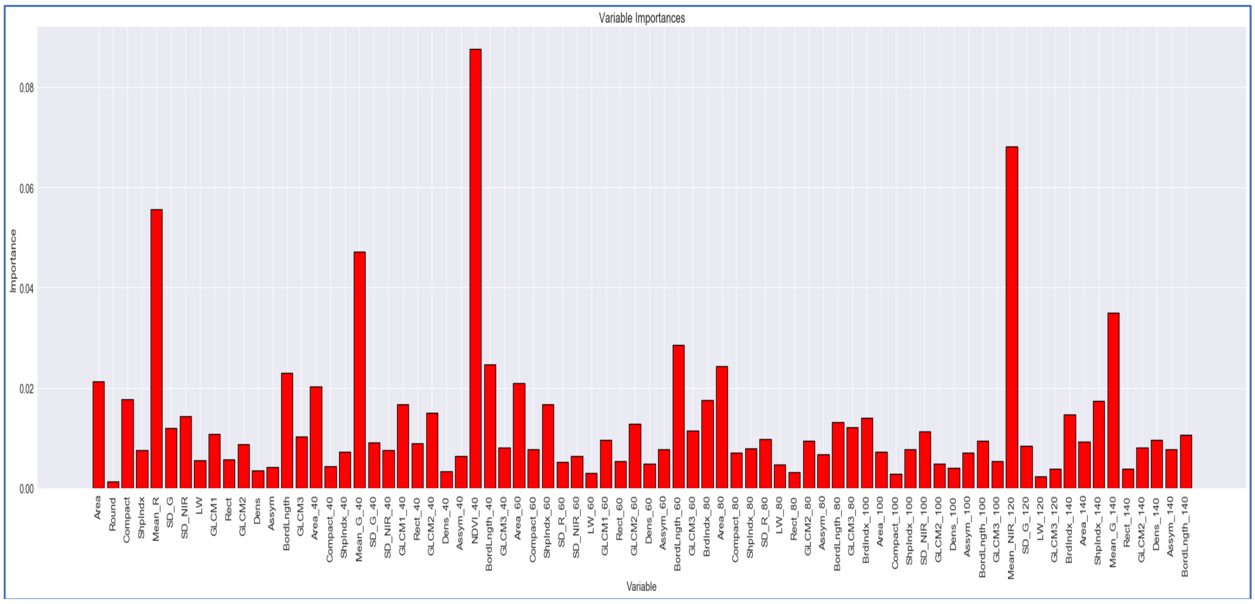
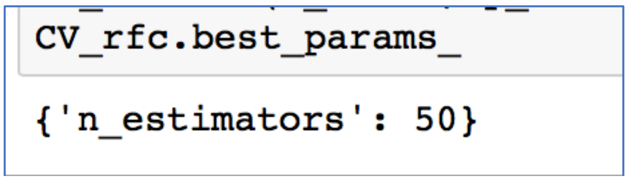


Figure 4.1 the importance of each feature in random forest

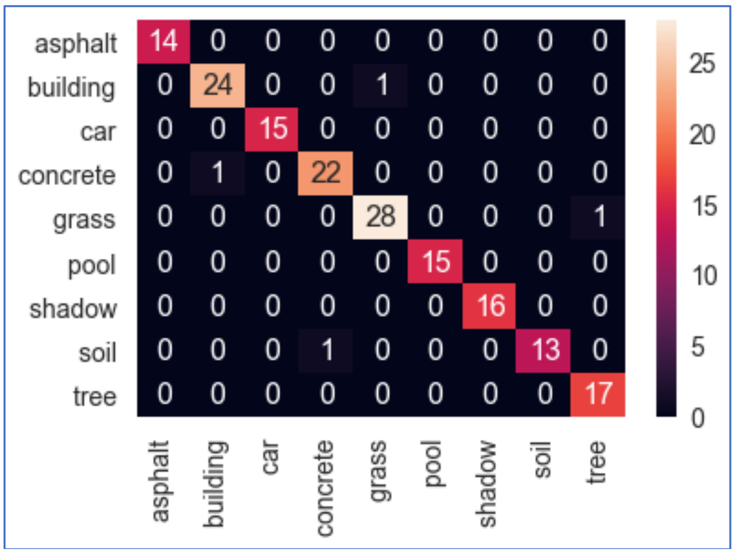
We ran the grid search only for the n_estimator hyperparameter. n_estimator is the parameter which is used to specify the number of trees in the forest. Although more trees gave higher results in our case, we found that any parameter after 50 did not give us considerable increase in the performance.



```
CV_rfc.best_params_

{'n_estimators': 50}
```

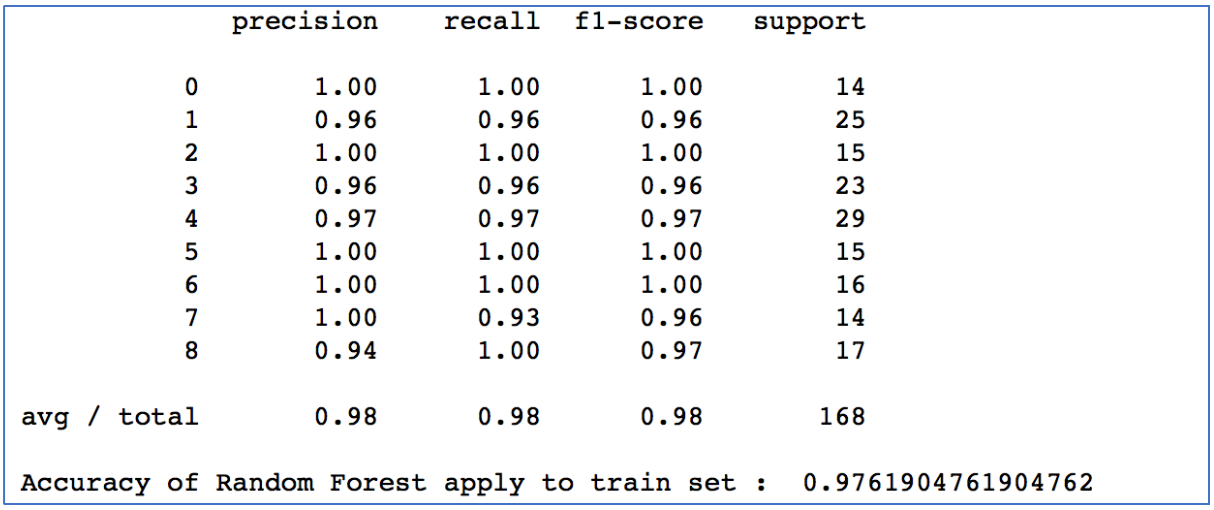
Figure 4.2 the best hyperparameter

And we also output the result of this model.
The figure 4.3-4.4 are the result of applying it to train set. Figure 4.5-4.6 are it applying to test set.



	asphalt	building	car	concrete	grass	pool	shadow	soil	tree
asphalt	14	0	0	0	0	0	0	0	0
building	0	24	0	0	1	0	0	0	0
car	0	0	15	0	0	0	0	0	0
concrete	0	1	0	22	0	0	0	0	0
grass	0	0	0	0	28	0	0	0	1
pool	0	0	0	0	0	15	0	0	0
shadow	0	0	0	0	0	0	16	0	0
soil	0	0	0	1	0	0	0	13	0
tree	0	0	0	0	0	0	0	0	17

Figure 4.3 the confusion matrix of training data

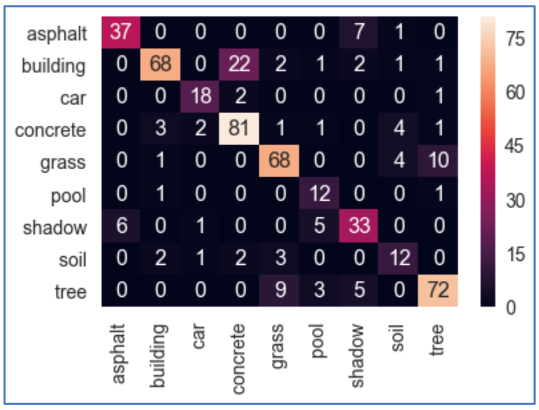


```
             precision    recall  f1-score   support

          0       1.00      1.00      1.00        14
          1       0.96      0.96      0.96        25
          2       1.00      1.00      1.00        15
          3       0.96      0.96      0.96        23
          4       0.97      0.97      0.97        29
          5       1.00      1.00      1.00        15
          6       1.00      1.00      1.00        16
          7       1.00      0.93      0.96        14
          8       0.94      1.00      0.97        17

avg / total       0.98      0.98      0.98       168

Accuracy of Random Forest apply to train set :  0.9761904761904762
```

Figure 4.4 the accuracy and other measurement of training data

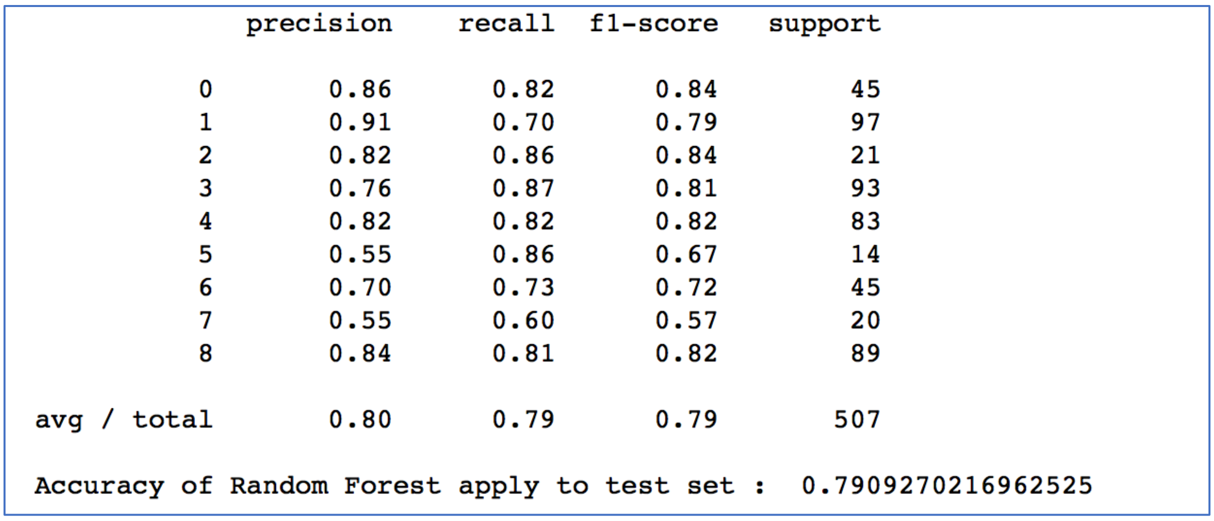
The diagrams and performance scores are for the train set after cross validation. We also performed the same test on train set without cross validation and the results were comparatively low.

The scores below are for the test. The confusion matrix below shows the labels that were not classified correctly.



	asphalt	building	car	concrete	grass	pool	shadow	soil	tree
asphalt	37	0	0	0	0	0	7	1	0
building	0	68	0	22	2	1	2	1	1
car	0	0	18	2	0	0	0	0	1
concrete	0	3	2	81	1	1	0	4	1
grass	0	1	0	0	68	0	0	4	10
pool	0	1	0	0	0	12	0	0	1
shadow	6	0	1	0	0	5	33	0	0
soil	0	2	1	2	3	0	0	12	0
tree	0	0	0	0	9	3	5	0	72

Figure 4.5 the confusion matrix of test data



```
             precision    recall  f1-score   support

          0       0.86      0.82      0.84        45
          1       0.91      0.70      0.79        97
          2       0.82      0.86      0.84        21
          3       0.76      0.87      0.81        93
          4       0.82      0.82      0.82        83
          5       0.55      0.86      0.67        14
          6       0.70      0.73      0.72        45
          7       0.55      0.60      0.57        20
          8       0.84      0.81      0.82        89

avg / total       0.80      0.79      0.79       507

Accuracy of Random Forest apply to test set :  0.7909270216962525
```

Figure 4.6 the accuracy and other measurement of test data

- **Random Forest(only top 25 importance feature):**

We decide to use only top 25 in feature importance. We used the festure_importances_ attribute from the random forest model to select the features. Decision trees, and the ensemble methods that are built on them, work by splitting the data into subsets. The tree will continue to build different subsets until it understands and represents the relationship of the variables with the target. Boosting ensemble trees take it a step further and learn from their mistakes, adjusting the trees iteratively to better classify the data. All kinds of tree methods calculate their splits by mathematically determining which split will most effectively help distinguish the classes. Because that is their method, the sklearn instances of these models have a feature_importances_ attribute, which returns an array of each feature's importance in determining the splits.
The figure 4.7-4.8 are the result of applying it to train set.
Figure 4.9-4.10 are the results of applying to test set.

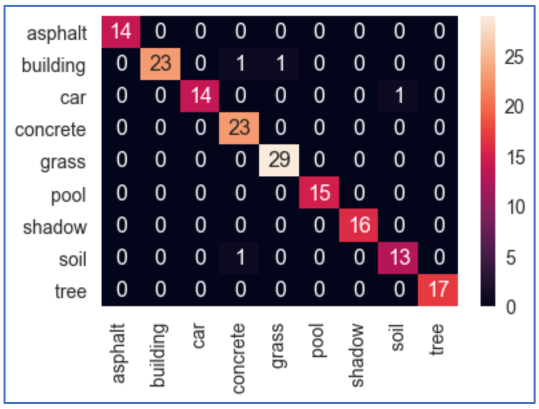
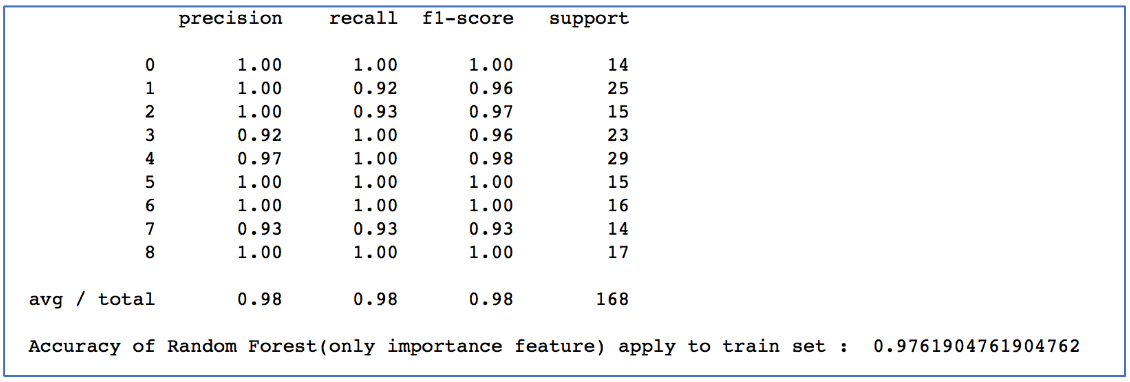


Figure 4.7 the confusion matrix of training data


```
              precision    recall  f1-score   support

          0       1.00      1.00      1.00        14
          1       1.00      0.92      0.96        25
          2       1.00      0.93      0.97        15
          3       0.92      1.00      0.96        23
          4       0.97      1.00      0.98        29
          5       1.00      1.00      1.00        15
          6       1.00      1.00      1.00        16
          7       0.93      0.93      0.93        14
          8       1.00      1.00      1.00        17

avg / total       0.98      0.98      0.98       168

Accuracy of Random Forest(only importance feature) apply to train set :  0.9761904761904762
```
Figure 4.8 the accuracy and other measurement of training data

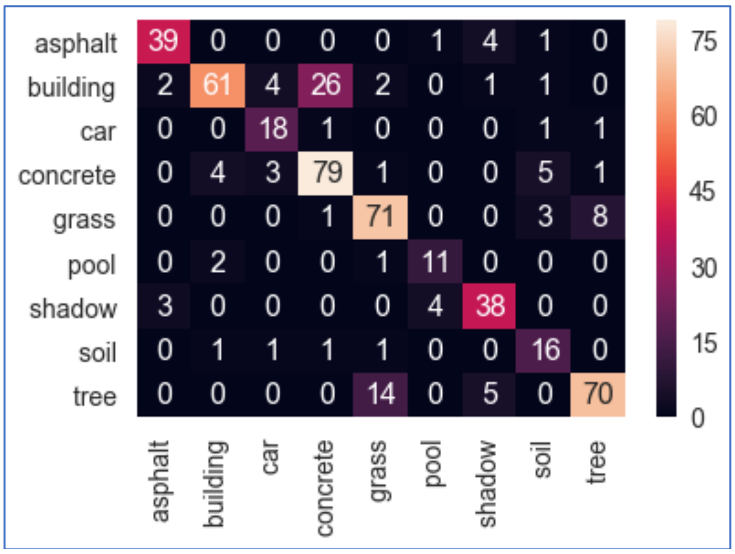
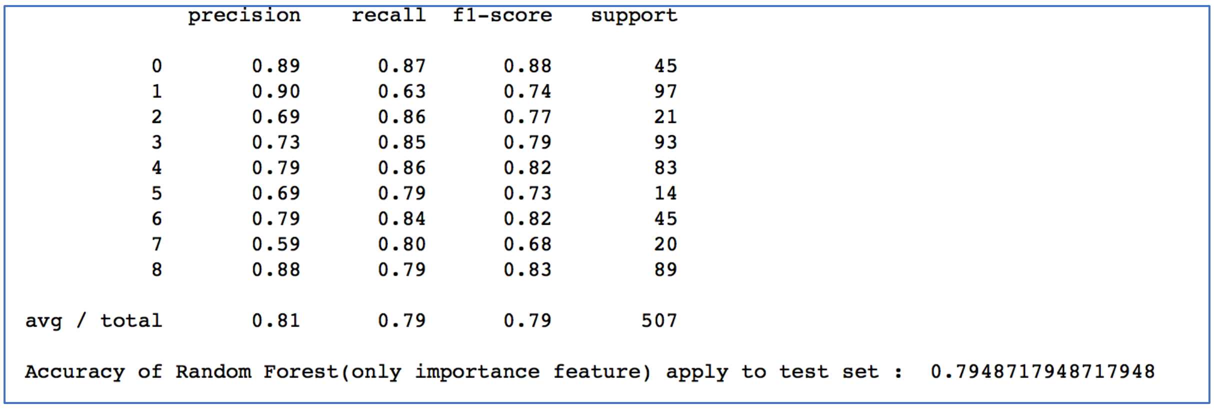


Figure 4.9 the confusion matrix of test data



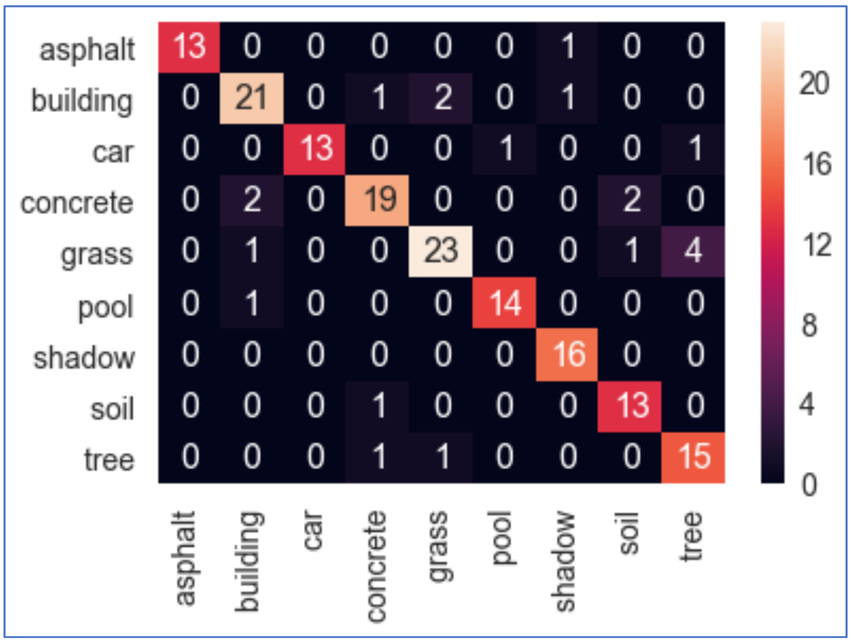
```
             precision    recall  f1-score   support

          0       0.89      0.87      0.88        45
          1       0.90      0.63      0.74        97
          2       0.69      0.86      0.77        21
          3       0.73      0.85      0.79        93
          4       0.79      0.86      0.82        83
          5       0.69      0.79      0.73        14
          6       0.79      0.84      0.82        45
          7       0.59      0.80      0.68        20
          8       0.88      0.79      0.83        89

avg / total       0.81      0.79      0.79       507

Accuracy of Random Forest(only importance feature) apply to test set :  0.7948717948717948
```

Figure 4.10 the accuracy and other measurement of test data

We find it performance is a little bit better than 77 features.

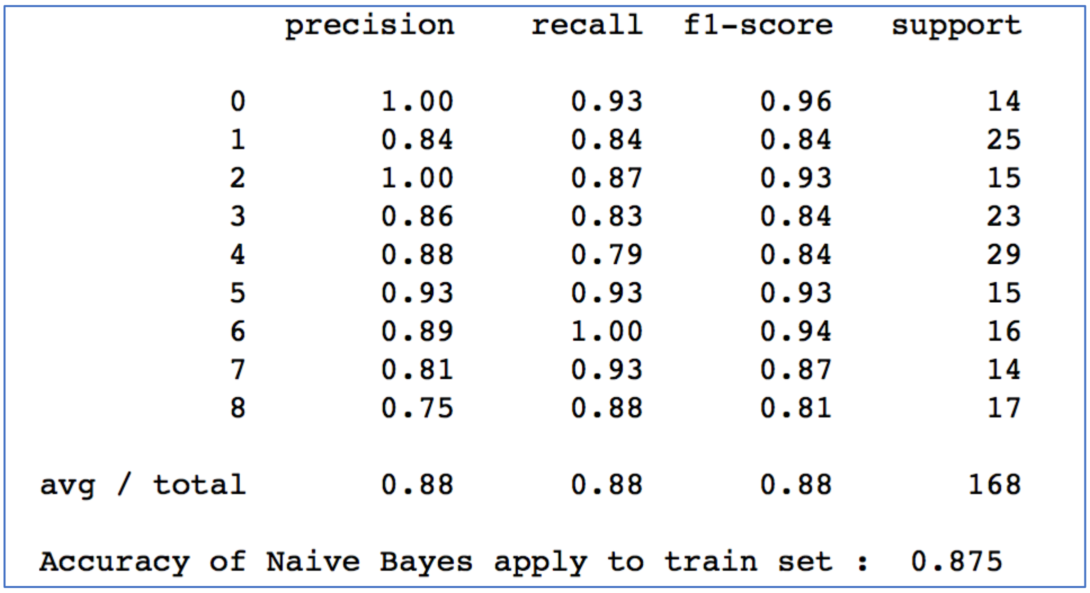
○ **Naïve Bayes**

The results are shown as below. The figure 4.11-4.12 are the result of applying it to train set. Figure 4.13-4.14 are results applying to test set.



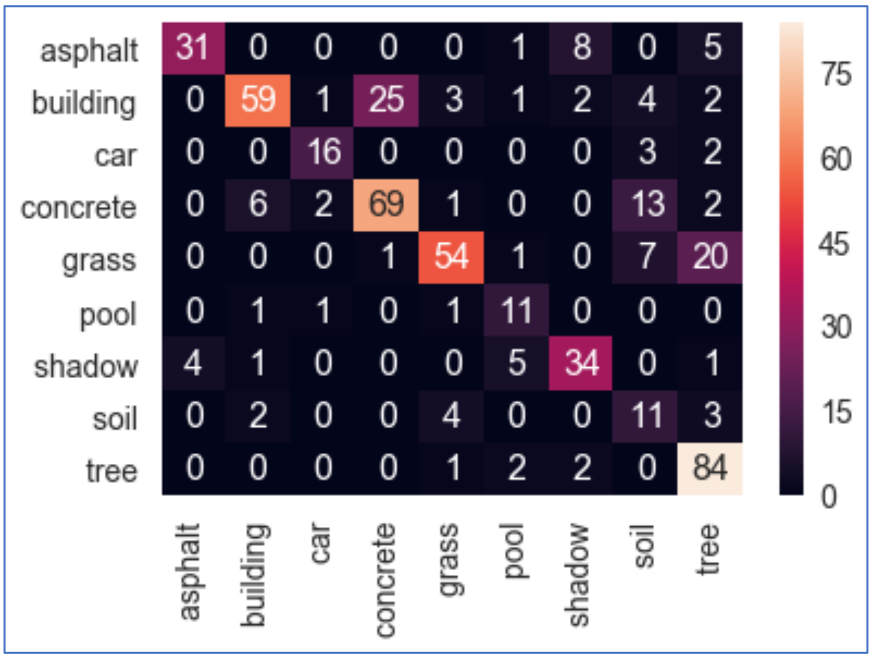
	asphalt	building	car	concrete	grass	pool	shadow	soil	tree
asphalt	13	0	0	0	0	0	1	0	0
building	0	21	0	1	2	0	1	0	0
car	0	0	13	0	0	1	0	0	1
concrete	0	2	0	19	0	0	0	2	0
grass	0	1	0	0	23	0	0	1	4
pool	0	1	0	0	0	14	0	0	0
shadow	0	0	0	0	0	0	16	0	0
soil	0	0	0	1	0	0	0	13	0
tree	0	0	0	1	1	0	0	0	15

Figure 4.11 the confusion matrix of training data



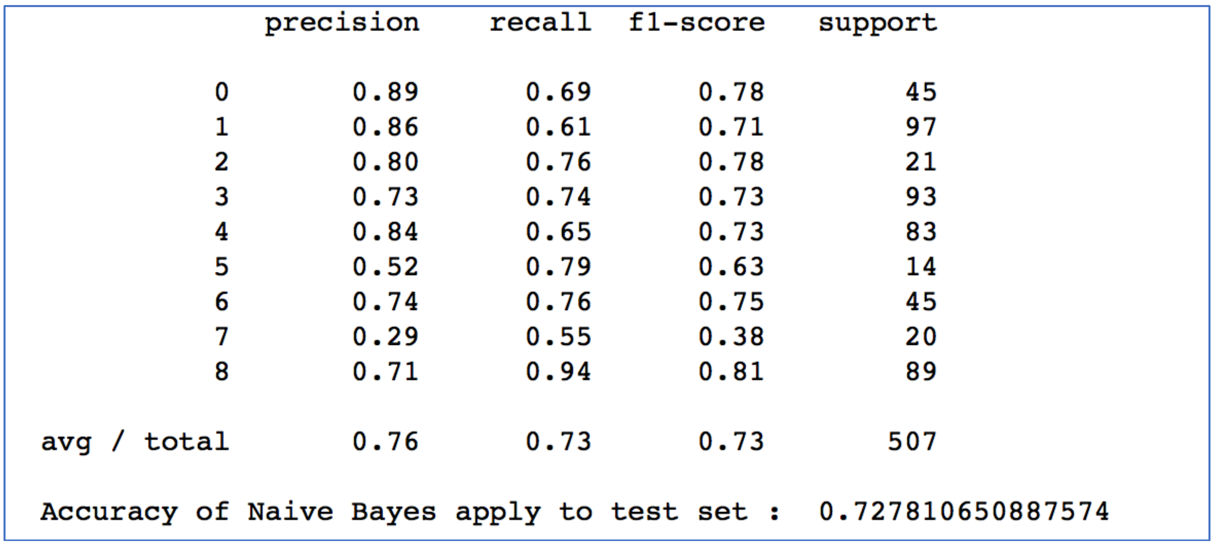
```
             precision    recall  f1-score   support

          0       1.00      0.93      0.96        14
          1       0.84      0.84      0.84        25
          2       1.00      0.87      0.93        15
          3       0.86      0.83      0.84        23
          4       0.88      0.79      0.84        29
          5       0.93      0.93      0.93        15
          6       0.89      1.00      0.94        16
          7       0.81      0.93      0.87        14
          8       0.75      0.88      0.81        17

avg / total       0.88      0.88      0.88       168

Accuracy of Naive Bayes apply to train set :  0.875
```

Figure 4.12 the accuracy and other measurement of training data



	asphalt	building	car	concrete	grass	pool	shadow	soil	tree
asphalt	31	0	0	0	0	1	8	0	5
building	0	59	1	25	3	1	2	4	2
car	0	0	16	0	0	0	0	3	2
concrete	0	6	2	69	1	0	0	13	2
grass	0	0	0	1	54	1	0	7	20
pool	0	1	1	0	1	11	0	0	0
shadow	4	1	0	0	0	5	34	0	1
soil	0	2	0	0	4	0	0	11	3
tree	0	0	0	0	1	2	2	0	84

Figure 4.13 the confusion matrix of test data



```
             precision    recall  f1-score   support

          0       0.89      0.69      0.78        45
          1       0.86      0.61      0.71        97
          2       0.80      0.76      0.78        21
          3       0.73      0.74      0.73        93
          4       0.84      0.65      0.73        83
          5       0.52      0.79      0.63        14
          6       0.74      0.76      0.75        45
          7       0.29      0.55      0.38        20
          8       0.71      0.94      0.81        89

avg / total       0.76      0.73      0.73       507

Accuracy of Naive Bayes apply to test set :  0.727810650887574
```

Figure 4.14 the accuracy and other measurement of test data

Questions:

How did you choose which classification algorithm would be appropriate for these data?

Random Forests are efficient when predicting categories which are multi-class (two or more categories). It is accurate and fast to train in those situations. Although multiclass logistic regression was also a choice, it is fast but not generally as accurate as random forests. Also, Random Forests do not overfit as much as decision trees.

For the bonus part we decided to pick Naïve Bayes because it is one of those algorithms which is efficient for performing multi class classification for relatively small datasets. So we thought it would be meaningful to compare Random Forest and Naïve Bayes. As seen above, the accuracy difference was not very high between the two algorithms.

How did you choose which performance metrics to report?

We use confusion matrix to see which label are labeled by mistakes to which one. Then find the accuracy, precision, recall and f1-score to measure their performance. It gives a comprehensive information like precision recall and F1-score.

Are there any issues with your model that are apparent from the confusion matrix, but not from your performance metrics?

From confusion matrix, we can clearly see which has been mis-predict to which label. But in performance, we only know the accuracy of predict, how many labels are error predict. Although, the predictions are wrong it is related. For instance, in Figure 4.9 from confusion matrix, we can see that buildings have been predicted as concrete and trees have been predicted wrongly as grass. It is reasonable that model has predicted something which is closely related to the actual prediction that shows the reasonable performance of the model.

References:

1. Feature Importance: http://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html
2. GridSearch: https://machinelearningmastery.com/how-to-tune-algorithm-parameters-with-scikit-learn/
3. Cross Validation: https://machinelearningmastery.com/k-fold-cross-validation/