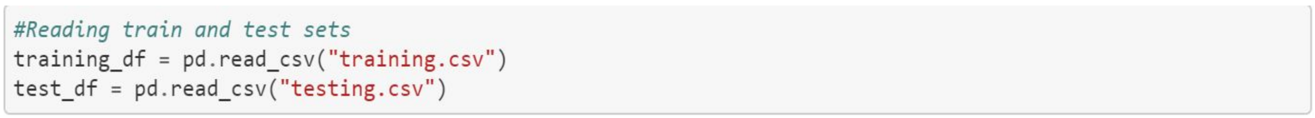
ASSIGNMENT-2
CLASSIFICATION

Group 8 members : Aarthi Mahalakshmi Shankar, Aayushi Vora

Workflow

1. **Imported test and training dataset as a pandas dataframe.**
   The Urban Land Dataset is divided in training and test. We imported it as a pandas dataframe.



```
#Reading train and test sets
training_df = pd.read_csv("training.csv")
test_df = pd.read_csv("testing.csv")
```

Fig 1. Data import

The shape of the training data frame is (168,148) and the shape of the test data frame is (507,148).
It has 148 features having data types like int and float.

2. **Exploratory analysis:**
   We checked if the dataset contains any null values. It do not contain any null values. There is one 'class' variable which is the target variable. This target variable has 9 labels.

   The distribution of the 9 labels in the data set have been visualized below:

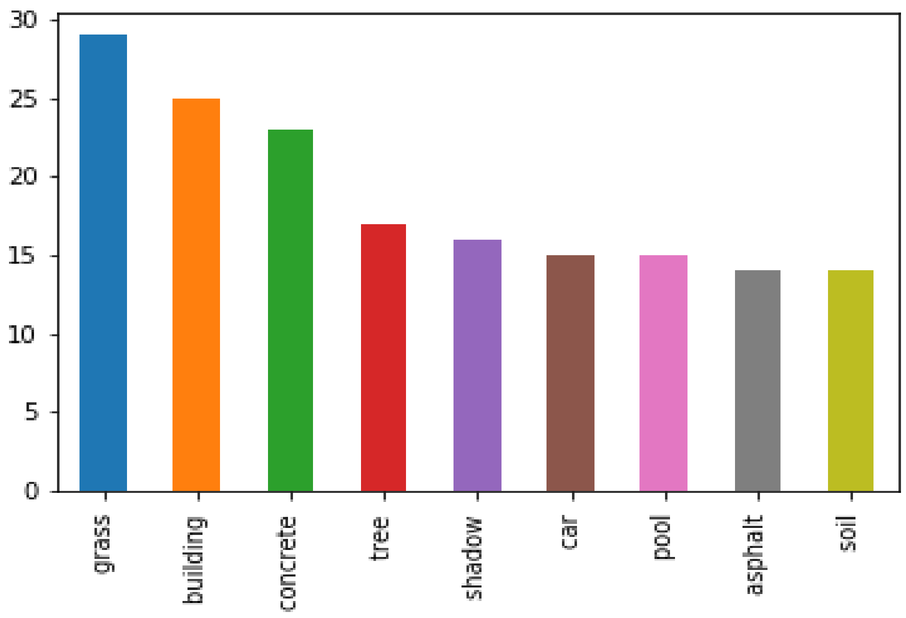


Fig 2. Distribution of labels

Fig 2 shows that grass, building and concrete have the highest number of occurrences. All the other features have almost equal number of occurrences. This also shows us that the data is not highly skewed.

Analysis of a particular feature - 'Area'

We checked on how one Area is distributed across the entire train dataset

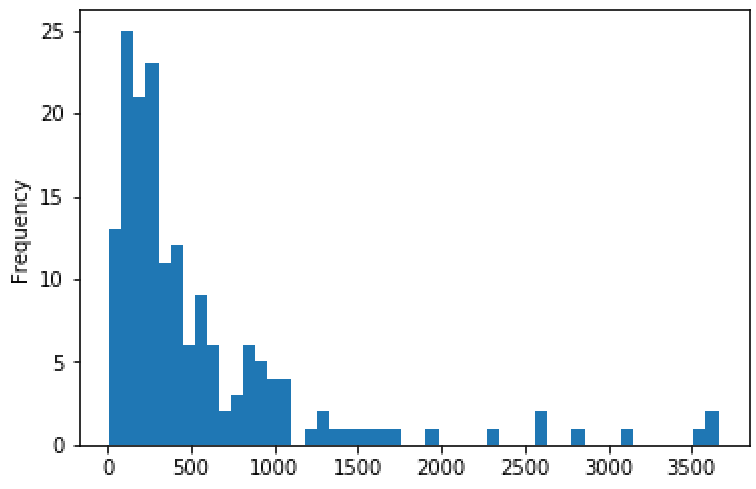


Fig 3. Distribution of 'Area' feature

Fig 3 shows how Area is distributed across the entire dataset. The frequency of occurrences of area shows that the distribution is mostly scattered.

We also checked how the sum of area for the entire dataset is distributed across the 9 labels

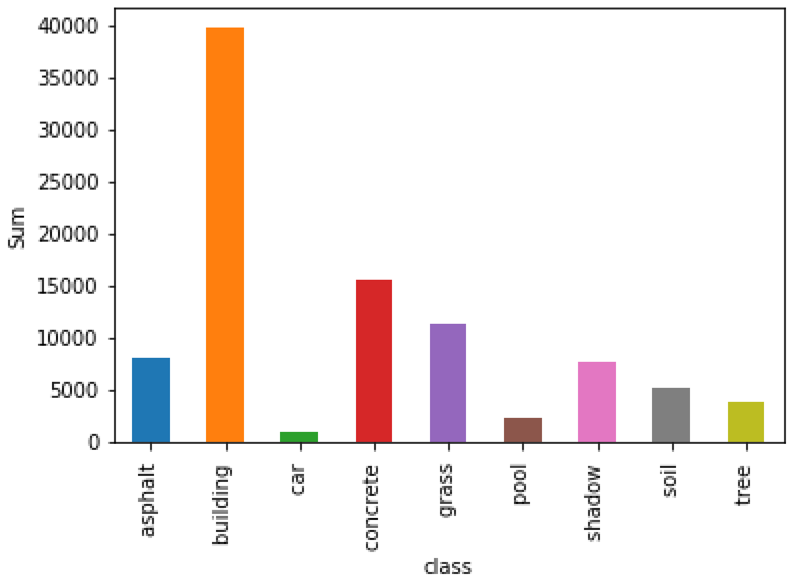


Fig 4. Distribution of Sum of Area for each class type

Fig 4 shows how the total sum of area is distributed across the dataset in terms of the 9 labels that are present. We can see that building has the highest area and concrete has the second highest.

Now, we have 4 main data to work with:
trainData : It has all the variables in the training data except the 'class' variable.
testData : It has all the variables in the test data except the 'class' variable.
trainLabel : It has 'class' variable from the training data.

testLabel : It has ‘class’ variable from the test data.

After Separating out the label from training and test data, we found that the feature set data was not normalized. Therefore, we normalized the data in the test and train set.

Finding correlation between variables

CORRELATION : This is a statistical technique that shows the relation between the pairs of variables.

Correlation Coefficient : The result of correlation is correlation coefficient(‘r’). It ranges from -1.0 to +1.0. The closer r is to +1 or -1, the more closely the two variables are related.

We then found the correlation between variables (all features) in this dataset to get a better idea of how they are related.

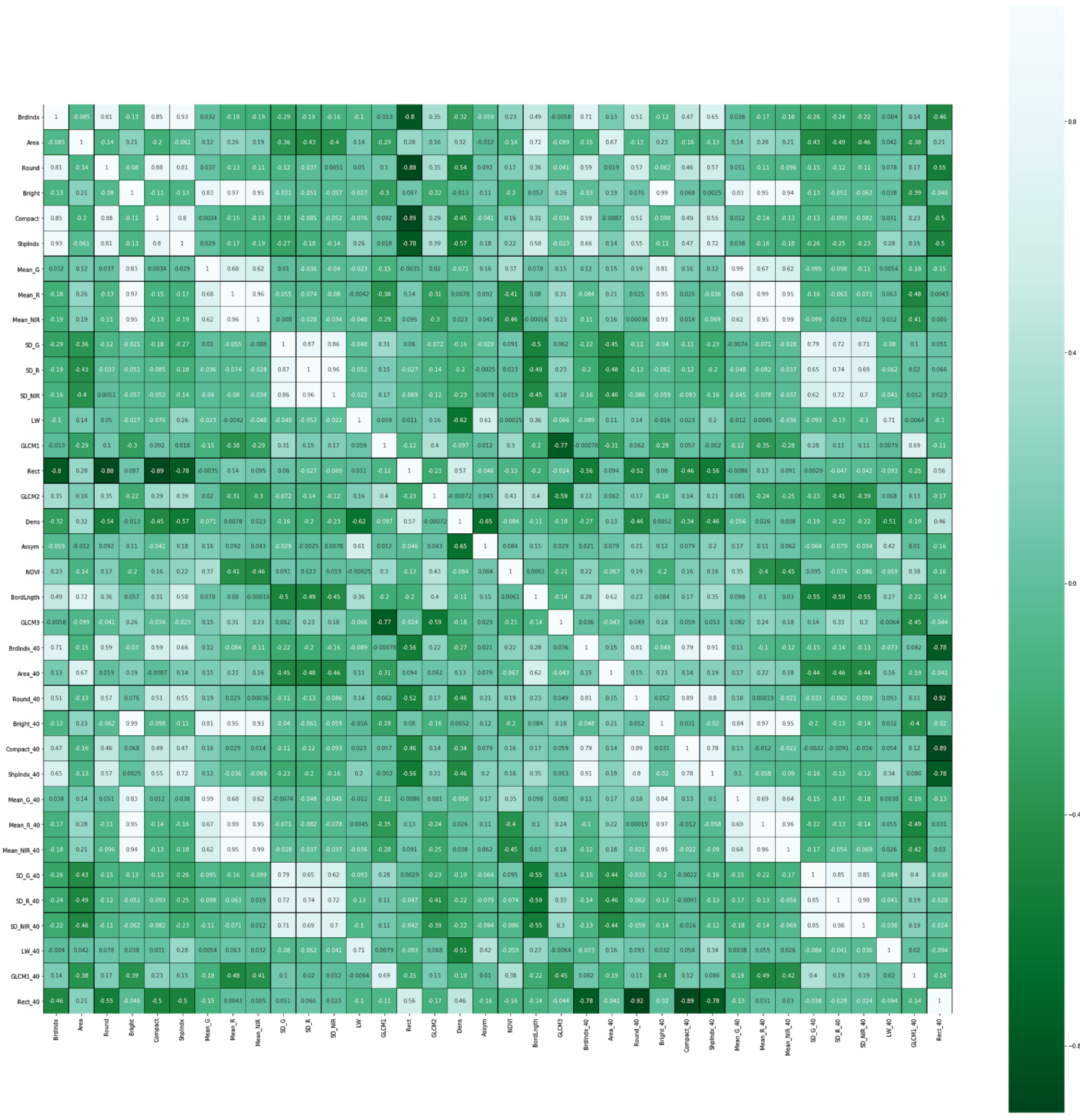


Fig 5. Correlation between all features

Fig 5. Shows the correlation between all the variables in the dataset. There are about 148 features. This is the reason why this figure is a little messed up and difficult to read/understand. Therefore, we decided to calculate the top most important features.

3. **Identifying important features**

We have a lot of features in this dataset. We wanted to observe what the important features are in the dataset. Below is the feature importance based on random forest classifier:

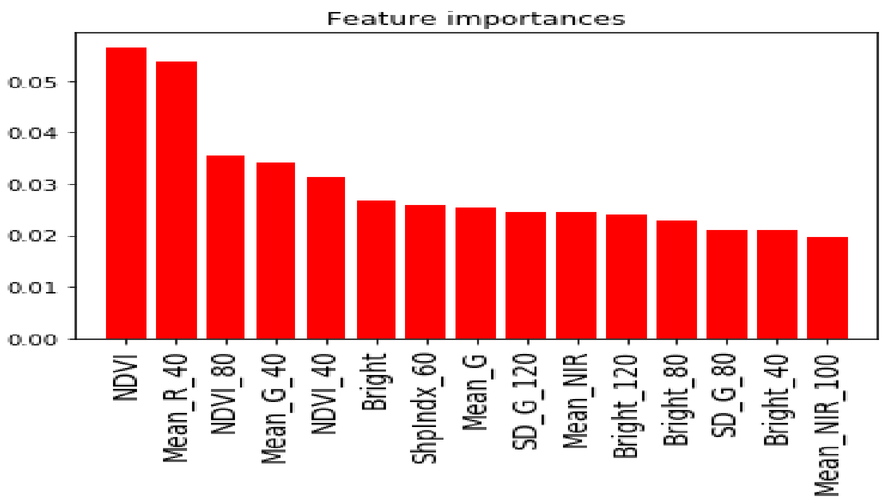


Fig 6. Feature importance based on random forest

Fig 6 shows the top most important features using randomForest.feature_importances_ using importance scores.

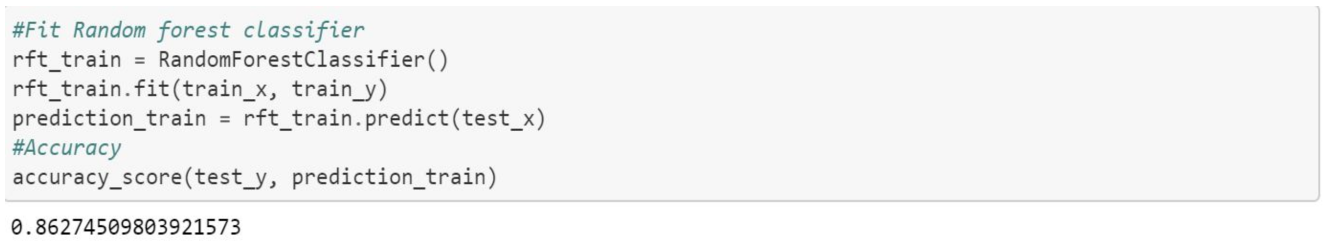
Model Prediction:

We have done the model prediction in 3 ways.

METHOD 1: Straight forward approach for classification without feature selection

TYPE 1: Split Train and Test set

We split the training set into train and test set. This was done to compare how well a model gets trained to predict with its own split test set. The accuracy for this model was **86%** with all the features.



```
#Fit Random forest classifier
rft_train = RandomForestClassifier()
rft_train.fit(train_x, train_y)
prediction_train = rft_train.predict(test_x)
#Accuracy
accuracy_score(test_y, prediction_train)
```

```
0.86274509803921573
```

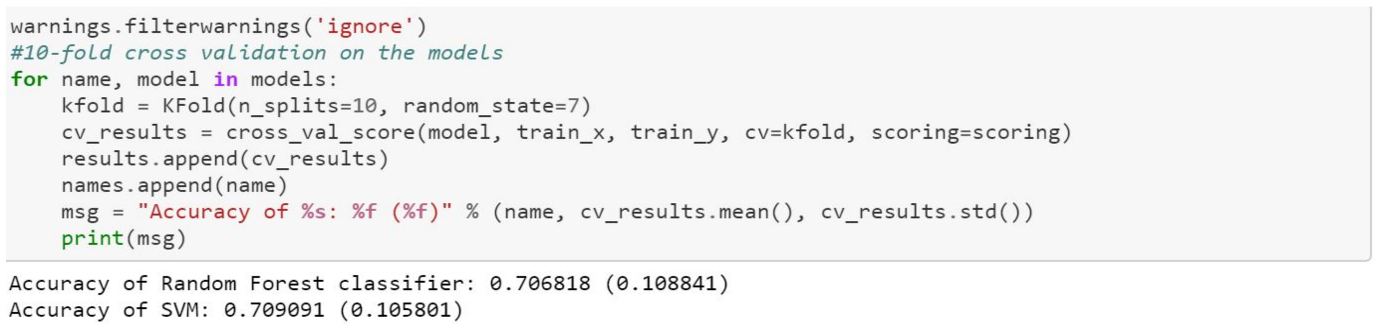
Fig 7. Applying Random Forests on training split of the Training data

Fig 7 shows how random forest classifier has been applied on the train data and predicted using test data.

TYPE 2: Use given train and test set with cross-validation

We used the given training and test set to predict the model. Two classifiers were used in this case - Random forest classifier and Support Vector Machine (which is the Optional task - to check how the results vary)

- The model was first fit using both the classifiers.
- 10- fold Cross-validation was performed on the fit models



```
warnings.filterwarnings('ignore')
#10-fold cross validation on the models
for name, model in models:
    kfold = KFold(n_splits=10, random_state=7)
    cv_results = cross_val_score(model, train_x, train_y, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "Accuracy of %s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
Accuracy of Random Forest classifier: 0.706818 (0.108841)
Accuracy of SVM: 0.709091 (0.105801)
```

Fig 8. Displaying accuracy of Random Forests and SVM after applying cross validation

- Compared both cross validation scores using a box plot - given below

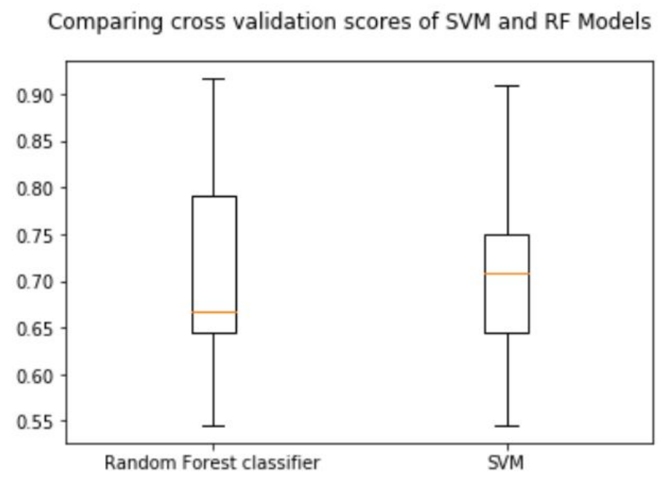


Fig 9. Comparing the cross validation scores of both the classifiers

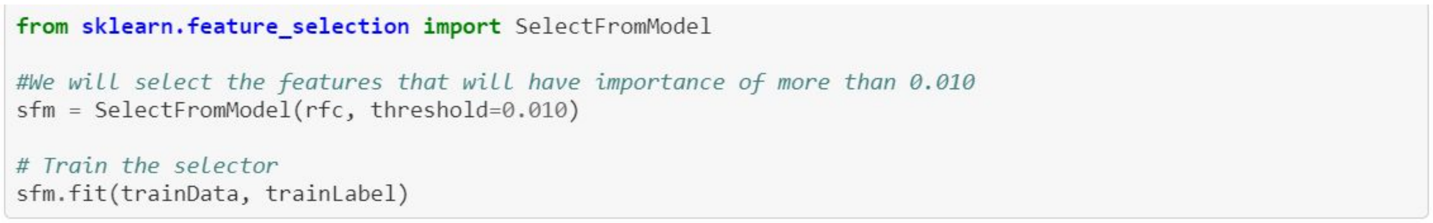
From Fig 9, we can see that the cross validation score for Random forest model is 0.706 and the Cross validation score for Support vector machine model is 0.709

- Predicted using test set and calculated accuracy

Accuracy using Random forest: **79%**
Accuracy using Support vector machine: **74%**

METHOD 2: Using feature selection and cross-validation

In this method, we have done Feature Selection. **Feature selection** has 3 benefits. First it makes the model more simple and interpretable. Second, we can reduce variance. This will eventually avoid overfitting. Also, we can reduce the computational costs for training the model. We have used Random forests for feature selection. Tree based strategies used by Random forests calculates purity of the node, thus they rank the features well. Impurity occur at the end of the trees. This is how we prune tree below a particular node.



```
from sklearn.feature_selection import SelectFromModel

#We will select the features that will have importance of more than 0.010
sfm = SelectFromModel(rfc, threshold=0.010)

# Train the selector
sfm.fit(trainData, trainLabel)
```

Fig 10.Code to specify threshold for feature selection

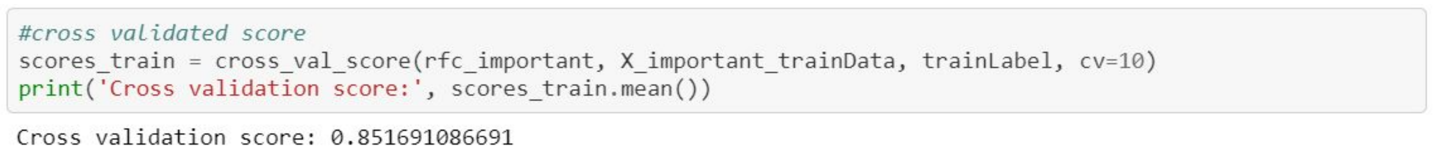
Fig 10 shows that we have kept a threshold of 0.010 to identify the features. The features that have an importance score of 0.010 and above are selected. 35 features were selected.

Features that were selected:

Area
Compact
Mean_G
Mean_R
Mean_NIR
SD_R
NDVI
Mean_G_40
Mean_R_40
Mean_NIR_40
NDVI_40
Mean_G_60
Mean_R_60
Mean_NIR_60
NDVI_60
BordLngth_60
Bright_80
Mean_R_80
Mean_NIR_80
SD_G_80
NDVI_80
Bright_100
ShpIndx_100
Mean_G_100
Mean_R_100
SD_G_100
SD_R_100
NDVI_100
Bright_120
Mean_NIR_120
NDVI_120
BrdIndx_140
Bright_140

Mean_R_140
NDVI_140

First, we have selected the features and created new data set from the new selected features. We have applied cross validation on the training data.

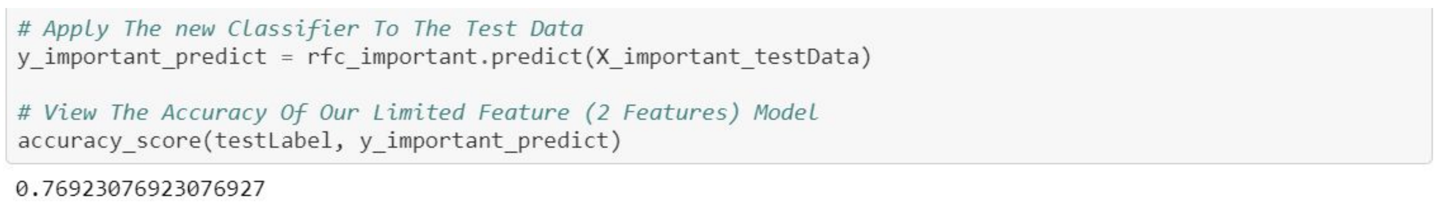


```
#cross validated score
scores_train = cross_val_score(rfc_important, X_important_trainData, trainLabel, cv=10)
print('Cross validation score:', scores_train.mean())
```

```
Cross validation score: 0.851691086691
```

Fig 11. Cross validation score for training data

We can now see the accuracy, precision and recall values.



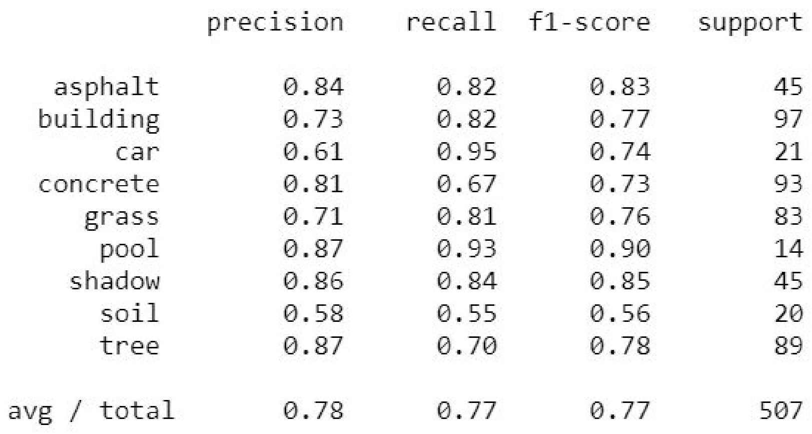
```
# Apply The new Classifier To The Test Data
y_important_predict = rfc_important.predict(X_important_testData)

# View The Accuracy Of Our Limited Feature (2 Features) Model
accuracy_score(testLabel, y_important_predict)
```

```
0.76923076923076927
```

Fig 12. Accuracy

Fig 12 shows that the accuracy of the model with the top most important features was found to be ***76%.***



```
             precision    recall  f1-score   support

    asphalt       0.84      0.82      0.83        45
   building       0.73      0.82      0.77        97
        car       0.61      0.95      0.74        21
   concrete       0.81      0.67      0.73        93
      grass       0.71      0.81      0.76        83
       pool       0.87      0.93      0.90        14
     shadow       0.86      0.84      0.85        45
       soil       0.58      0.55      0.56        20
       tree       0.87      0.70      0.78        89

avg / total       0.78      0.77      0.77       507
```

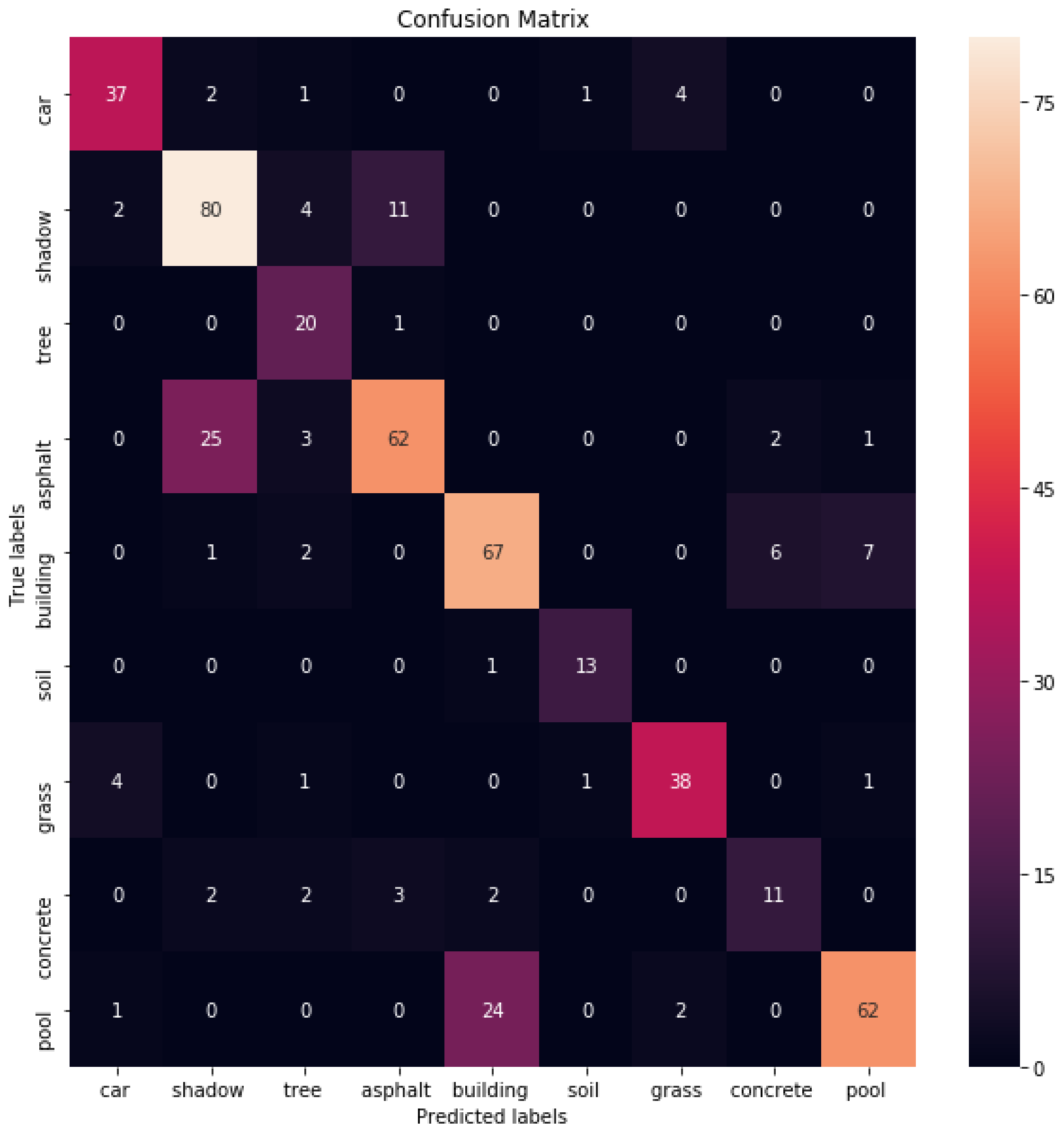
Fig 13. Precision, recall and f1-score

Confusion Matrix :

We need to understand some simple terms before reading this confusion matrix.
ACCURACY : This shows how often the classifier is correct.
ERROR RATE: How often is the classifier wrong.
TRUE POSITIVE: When does the classifier predict a YES, when it is actually a YES.
FALSE POSITIVE: When does the classifier predict a YES, when it is actually a NO.

So, confusion matrix for a binary classifier will be in this format:

	PREDICTED NO	PREDICTED YES
ACTUAL NO	✔	✘
ACTUAL YES	✘	✔



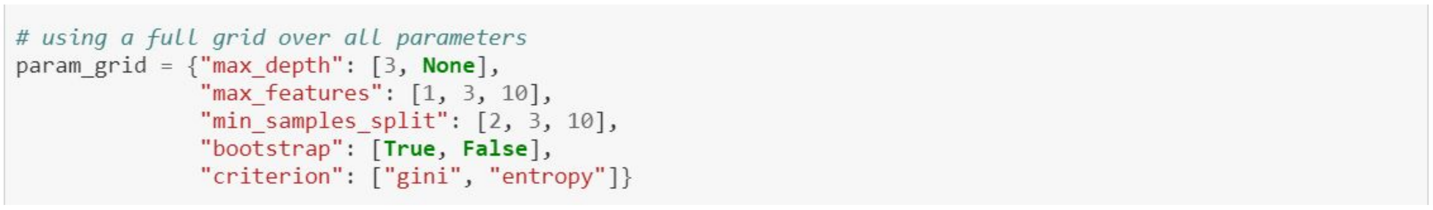
True \ Predicted	car	shadow	tree	asphalt	building	soil	grass	concrete	pool
car	37	2	1	0	0	1	4	0	0
shadow	2	80	4	11	0	0	0	0	0
tree	0	0	20	1	0	0	0	0	0
asphalt	0	25	3	62	0	0	0	2	1
building	0	1	2	0	67	0	0	6	7
soil	0	0	0	0	1	13	0	0	0
grass	4	0	1	0	0	1	38	0	1
concrete	0	2	2	3	2	0	0	11	0
pool	1	0	0	0	24	0	2	0	62

Fig 14 Confusion matrix

In Fig 14 the diagonal shows that those classes are predicted properly. The triangle below and above the diagonal have some misclassifications on the labels. For example, grass has been misclassified as pool 2 times.

METHOD 3: Grid search for Random Forest

Grid search is a technique for optimizing the hyperparameters. We worked on these hyperparameters : maximum depth, maximum features, split, bootstrap values, criterion. We have used GridSearch from sklearn.

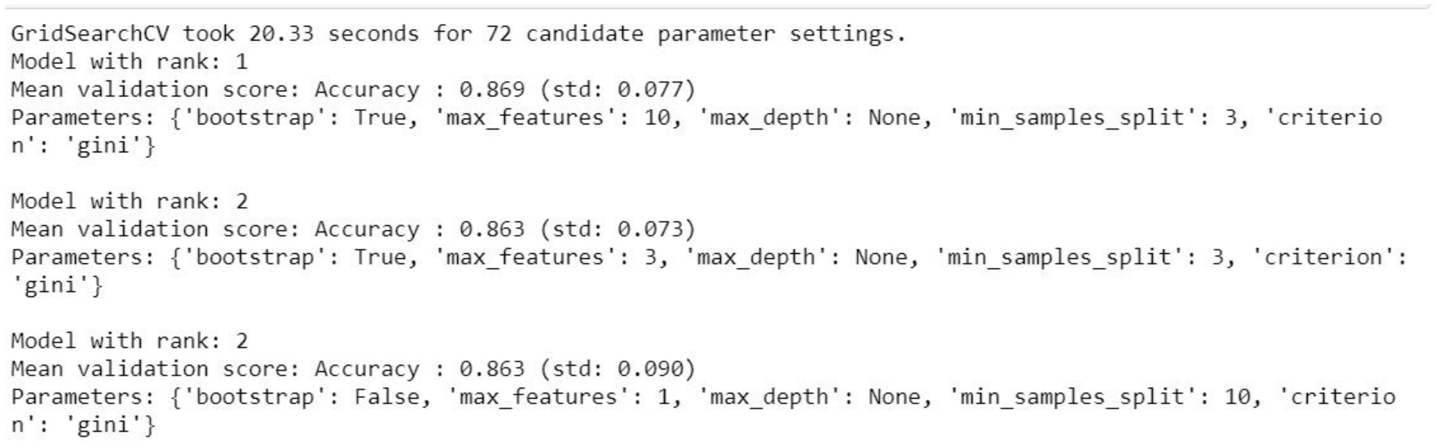


```
# using a full grid over all parameters
param_grid = {"max_depth": [3, None],
              "max_features": [1, 3, 10],
              "min_samples_split": [2, 3, 10],
              "bootstrap": [True, False],
              "criterion": ["gini", "entropy"]}
```

Fig 15. Setting hyper parameters

Fig 15 shows how we set-up the hyper parameters for grid search

Grid search is used to tune the model by searching for the best hyper parameters while keeping the classifier with the highest recall score. We first fit the random forest model. Then we selected the parameters as shown in Fig 15. We then applied the GridSearchCV using the fitted random forest model and displayed the result.



```
GridSearchCV took 20.33 seconds for 72 candidate parameter settings.
Model with rank: 1
Mean validation score: Accuracy : 0.869 (std: 0.077)
Parameters: {'bootstrap': True, 'max_features': 10, 'max_depth': None, 'min_samples_split': 3, 'criterio
n': 'gini'}

Model with rank: 2
Mean validation score: Accuracy : 0.863 (std: 0.073)
Parameters: {'bootstrap': True, 'max_features': 3, 'max_depth': None, 'min_samples_split': 3, 'criterion':
'gini'}

Model with rank: 2
Mean validation score: Accuracy : 0.863 (std: 0.090)
Parameters: {'bootstrap': False, 'max_features': 1, 'max_depth': None, 'min_samples_split': 10, 'criterio
n': 'gini'}
```

Fig 16. Displaying top 3 hyper results of Grid Search

From fig 16, we can see the top 3 models that were generated from grid search using different hyper parameters - the first model with 10 features, second with 3 and the third with 1 , all having a good accuracy.

Overall Result:

The accuracy, precision and recall of all the three methods are given below:

	Random forest (without feature selection but with cross validation)	SVM (without feature selection but with cross validation)	Random forest (With feature selection and cross validation)	Grid search for Random Forest
Accuracy	0.79	0.73	0.77	0.87

Precision	0.80	0.74	0.78	0.77
Recall	0.79	0.73	0.77	0.87

We can conclude from the above table that grid search with a maximum of 10 features has given the highest accuracy of 87%. We chose to do the classification using 3 methods just to experiment different ways of predicting. The main aim was to learn through different methods, where the final goal was the same. We got to learn how to choose the important features, how to reduce overfitting using cross validation and discovered a new method called as grid search - which surprisingly gave us a better result compared to the other methods.

References:

- http://scikit-learn.org/stable/auto_examples/model_selection/plot_randomized_search.html
- http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- http://scikit-learn.org/stable/modules/svm.html
- http://scikit-learn.org/stable/modules/cross_validation.html
- http://scikit-learn.org/stable/