Machine Learning Homework 6
Report
December 14, 2017

Karan, FNU (kx361) Chauhan, Romil (rhc294)

1. Feature Selection

The features in the problem are:

a. 'Summons_Number',
b. 'Registration_State',
c. 'Plate_Type',
d. 'Issue_Date',
e. 'Vehicle_Body_Type',
f. 'Vehicle_Make',
g. 'Issuing_Agency',
h. 'Street_Code1',
i. 'Street_Code2',
j. 'Street_Code3',
k. 'Violation_Precinct',
l. 'Issuer_Precinct',
m. 'Issuer_Code',
n. 'Violation_Time',
o. 'Front_Of_Or_Opposite',
p. 'From_Hours_In_Effect',
q. 'To_Hours_In_Effect',
r. 'Vehicle_Year',
s. 'Feet_From_Curb'

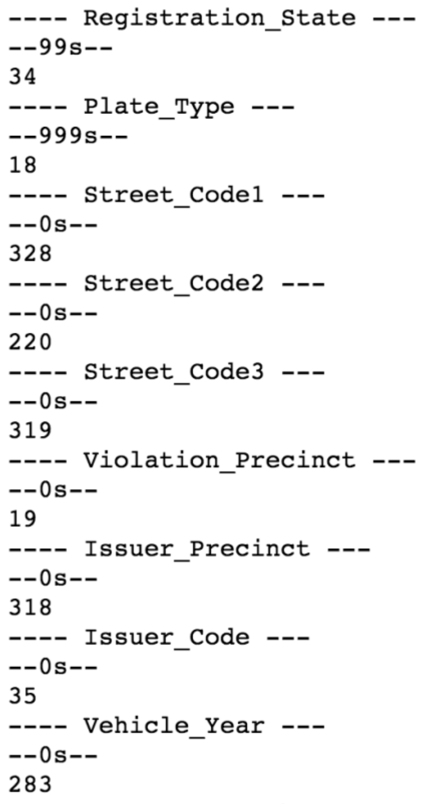
The target is: 'Violation_Code'

When we look at these columns, we can clearly say that some of these columns are irrelevant. So, we should not take them into account while making our model. These columns are:

a. 'Summons_Number',
b. 'Violation_Time',
c. 'Issue_Date',

For the columns, 'From_Hours_In_Effect' and 'To_Hours_In_Effect', these are times. We can keep the number of minutes, that is by subtracting 'From_Hours_In_Effect' from 'To_Hours_In_Effect'. But then there are cases where the resulting column has NULL values and that is because, there were 'ALL' values in original columns. We could replace these values by number of minutes in a day, assuming that these times were with respect to a certain day. After this we can also drop the columns, ‘From_Hours_In_Effect' and 'To_Hours_In_Effect’

Now we look at the data, and the value counts for each column. We realize that there are certain outliers and that might affect the performance of our model. These columns are shown below.



```
---- Registration_State ---
--99s--
34
---- Plate_Type ---
--999s--
18
---- Street_Code1 ---
--0s--
328
---- Street_Code2 ---
--0s--
220
---- Street_Code3 ---
--0s--
319
---- Violation_Precinct ---
--0s--
19
---- Issuer_Precinct ---
--0s--
318
---- Issuer_Code ---
--0s--
35
---- Vehicle_Year ---
--0s--
283
```

Since, number of observations with 99’s and 999’s is pretty less as compared to the total number of observations, we can drop them.

For the rest of the columns, we can replace them by the most frequently occurring value, since they are all categorical values.

Also, the column, ‘Feet_From_Curb’ has zeros, but we just cannot replace them by most frequently occurring value, since it is a measurement. So, we can just increment the values of this column by 1.

Now there are certain features that are categorical, and not numeric. They are: 'Registration_State', 'Plate_Type', 'Vehicle_Body_Type', 'Vehicle_Make', 'Issuing_Agency', 'Front_Of_Or_Opposite']

We can convert them to numerical by assigning a particular value to each of the values in the column, also called ‘Label Encoding’. This is not a good approach since it assigns values unequally to each of the values. We also tried ‘One Hot Encoding’, but unfortunately the

testing data consisted of some other columns as well. We were not quite sure how to handle those.
At this point the data looks pretty neat, except the fact that it is not scaled. So we performed scaling and the data is ready to be processed, and we are ready to build models.

2. Models

a. Decision Trees
The two main parameters affecting the decision trees are the min_samples_split and min_sample_leaf. For both of these parameters we tried to build a model and calculated the accuracy for all combinations of these two parameters. Also for all these combinations we changed the criterion on which the decision tree is made. The two methods we chose are: entropy and gini.

b. Random Forest
The two main parameters affecting the decision trees are the min_samples_split and min_sample_leaf. For both of these parameters we tried to build a model and calculated the accuracy for all combinations of these two parameters. Also for all these combinations we changed the criterion on which the decision tree is made. The two methods we chose are: entropy and gini.

c. k-NN
The main two parameters that we worked on are k, and weights. ‘k’ being the number of neighbors and weight function used in prediction. Possible values:

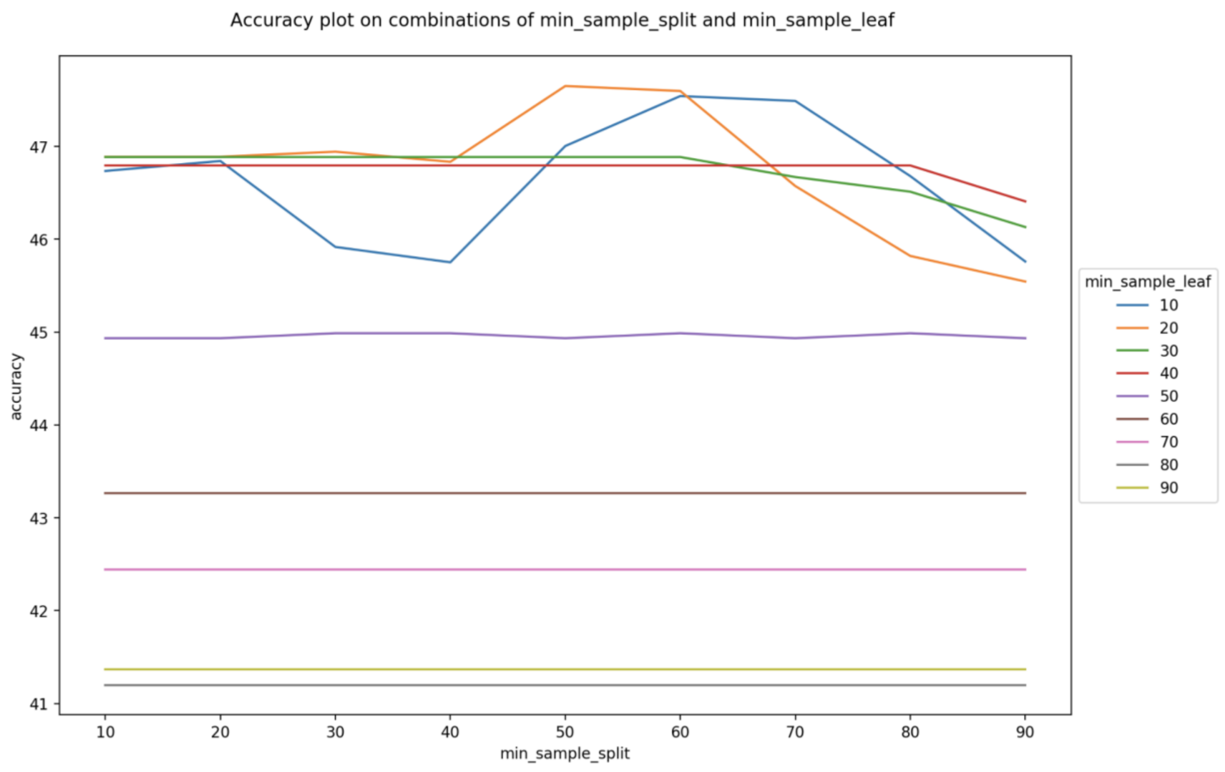
- ‘uniform’: uniform weights. All points in each neighborhood are weighted equally.
- ‘distance’: weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.
- [callable]: a user-defined function which accepts an array of distances, and returns an array of the same shape containing the weights.

We used ‘uniform’ and ‘distance’.

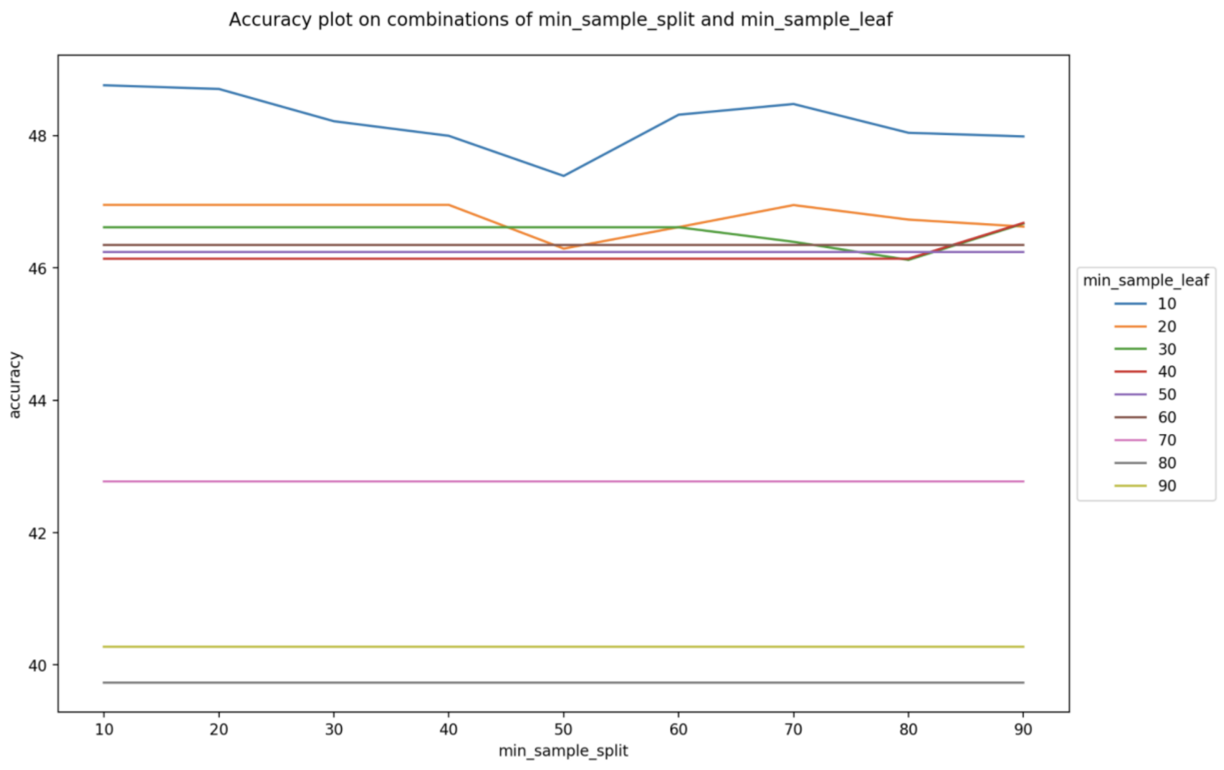
d. Logistic Regression

3. The parameter values and accuracies

a. Decision Trees

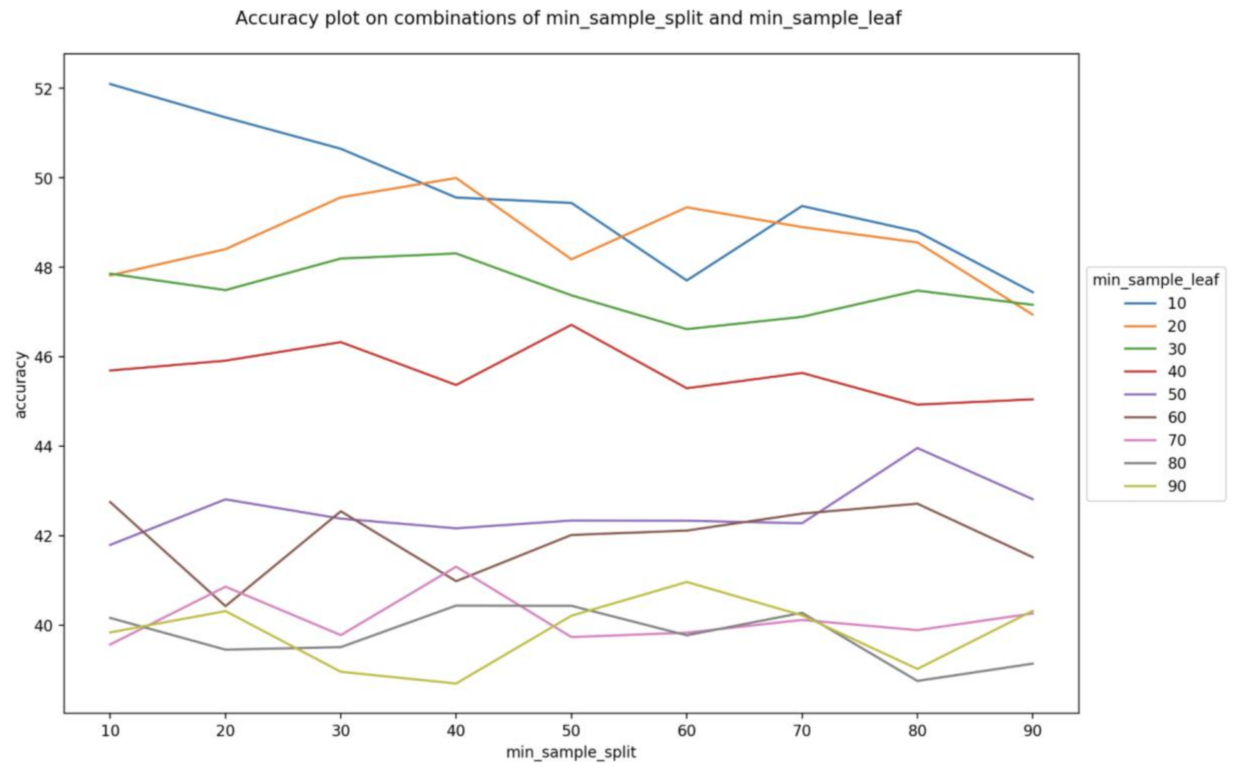


Above figure with criterion as 'entropy'.

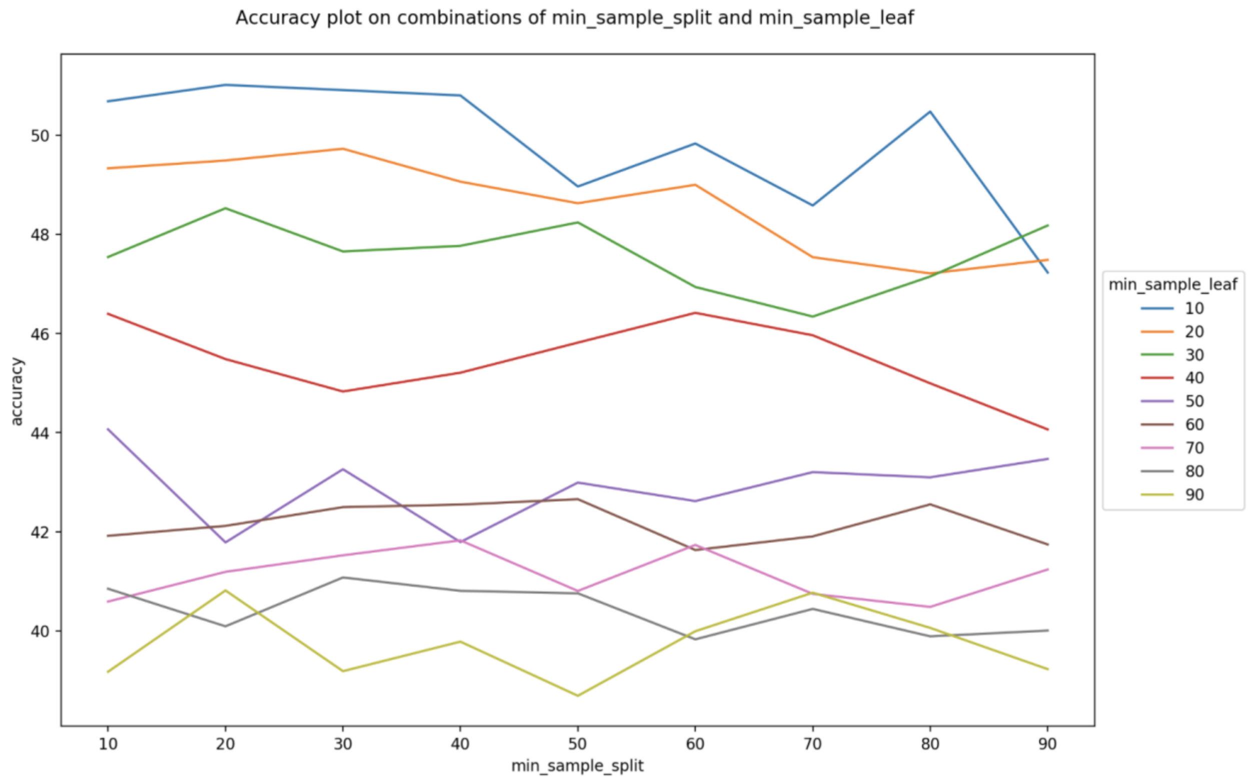


Above figure with criterion as 'gini'.

b. Random Forest

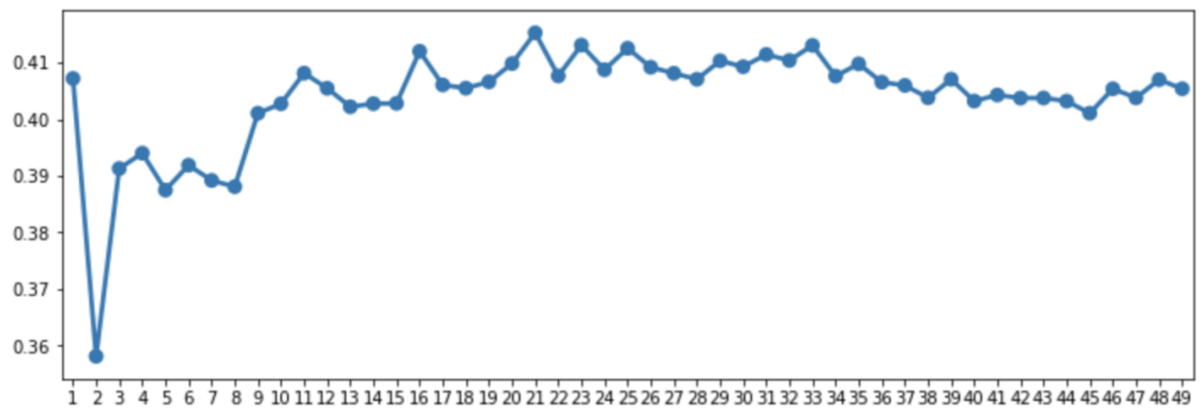


Above figure with criterion as 'entropy'.

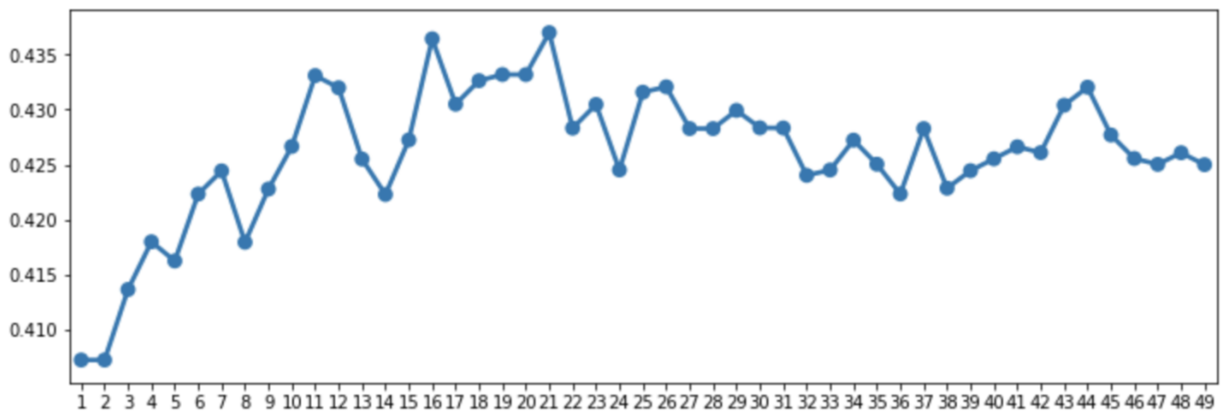


Above figure with criterion as 'gini.

c. k-NN



Above figure with weights as 'uniform'



Above figure with weights as 'distance'

d. Logistic Regression

Here we achieved an accuracy of 42.5%.

4. Model Selection

We finally come to a result that random forest model would be a good model to work with. The problem here was to work on a multiclass classification problem. So, the common models for this kind of a problem are Random Forest, Logistic Regression, Decision Tree, and K Nearest Neighbors.

The main idea behind choosing a certain algorithm is the accuracy that it gives. For the dataset and the number of features that have been provided, Random Forest is the only algorithm which gives us the highest accuracy, that is around 52%.

The comparison is made on the accuracies achieved by the results of 10-fold evaluation.

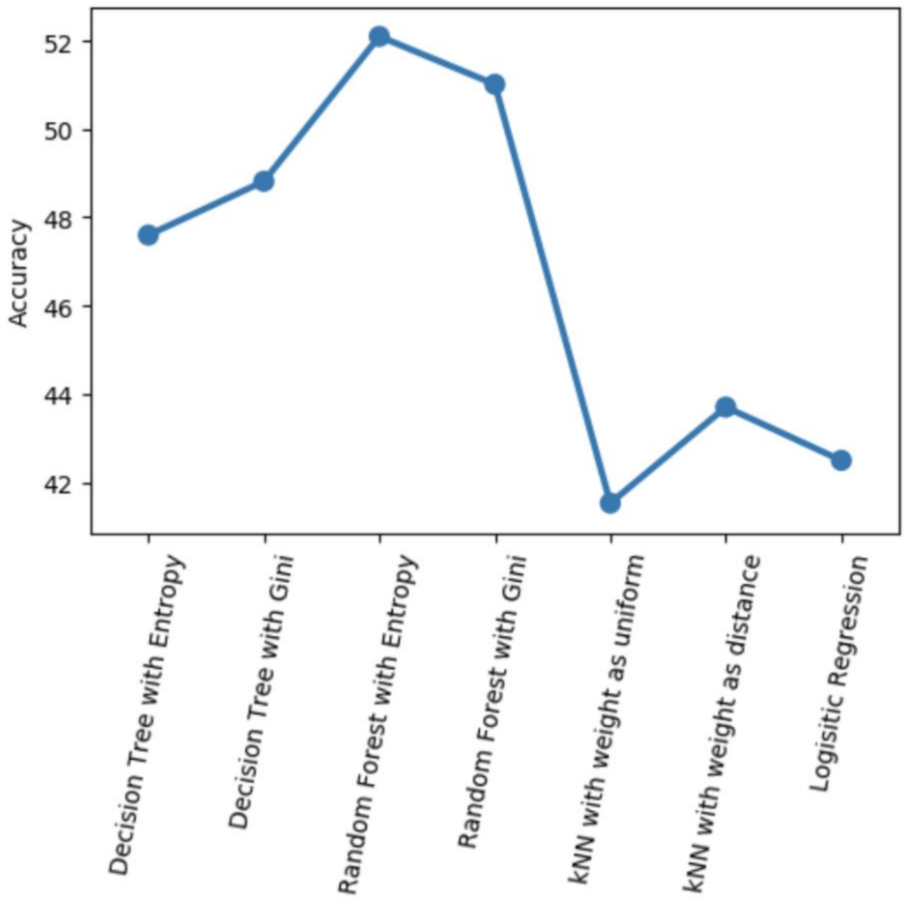
5. Final Model

For our final method, the accuracy for 10 folds is:

0.55080214,	0.49456522,
0.50802139,	0.54098361,
0.55135135,	0.4863388,
0.49189189,	0.5,
0.51891892,	0.47513812

Average accuracy: 0.5218.

Here is the comparison chart of accuracies of all the methods explored.



6. Libraries Used

a. Python Pandas
b. Python Numpy
c. Python Scikit learn