CS6240 - Project Report

Jakkappanavar Harsha ; Shenoy Priyanka

During lectures we learned about various modelling techniques and ways of implementing data mining. While implementing project we decided to focus more on understanding how the algorithm works rather than writing the algorithm ourself. For the same we decided on using MLlib which allowed us to focus on training, tuning, testing and understanding the functionality.

The thought process behind selecting random forest was, we had to get to a point where bias v/s variance ratio was optimal. This can be done easily by fine tuning the features which are part of declaring random forest using MLlib.Clustering could not be used for implementing because - →Using KNN had significant dependence on correct centers being identified and even if we did identify the centers there was a chance that we would end up getting local maxima.

Decision tree algorithm was to be used. We used Random forest over GBT for - → Faster training models → Random Forest are Easier to tune → Less prone to overfitting

Parameters Explored -

- COUNT_TYPE - P0P00_SQMI - DIST_IN_WET_VEG_FRESH
- EFFORT_DIST - HOUSING_DENSITY - NUMBER_OF_OBSERVERS
- EFFORT_HRS - ELEV_GT - ELEV_NED
- BCR -CAUS_TEMP_AVG -CAUS_PREC
- CAUS_SNOW - DIST_FROM_FLOWING_FRESH - MONTH
- DIST_IN_FLOWING_FRESH - DIST_FROM_STANDING_FRESH
- DIST_IN_STANDING_FRESH -DIST_FROM_WET_VEG_FRESH
- LONGITUDE - LATITUDE

Pseudocode - (Generalized for random forest)

Precondition: A training set S := (x1, y1), . . . ,(xn, yn), features F, and number of trees in forest NoT.

```
function RandomForest(S , F)
H ← ∅
for i belongs to 1, . . . , NoT
 Do
        S^(i) ← A bootstrap sample from S
        h(i) ← RandomizedTreeLearn(S^(i), F)
        H ← H ∪ {h(i)}
end for
return H
end function
function RandomizedTreeLearn(S , F)
At each node:
        f ← very small subset of F
        Split on best feature in f
return The learned tree
```

end function

Pre-processing Data -

- Clearing column header
- Clearing row if year | month | day have “?”
- If tuple contains “?” put value as 0
- If tuple contains “X” put value as 0
- Converting COUNT_TYPE, BCR from string to integer value to provide as categorical features.
- Categorical features include MONTH (12), COUNT_TYPE (5), BCR (37), CAUS_TEMP_AVG(9), CAUS_PREC(9), CAUS_SNOW(9)

Random Forest Algorithm

Parameters tuned :

numClasses = 2 (1|0)

Explanation :

Express if feature is present or not.

featureSubsetStrategy = auto

Explanation : I

If "auto" is set, this parameter is set based on numTrees

if numTrees == 1, set to "all"

if numTrees is greater than 1 (forest) set to "sqrt" for * classification and to "onethird" for regression.

numTrees = 75

impurity = gini

Explanation :

Using entropy did not change the accuracy hence used gini since computation is supposed

to be faster

maxDepth = 15

maxBins = 40

Explanation :

Slightly greater than largest size of categoricalFeaturesInfo

categoricalFeaturesInfo = [month, count_type, caus_temp_avg, bcr, caus_prec, caus_snow, dist_from_wet_veg, dist_in_wet_veg, dist_from_flowing_fresh, dist_in_flowing_fresh, dist_from_standing_fresh, dist_in_standing_fresh]

The random forest algorithm works in the following way -

- The algorithm creates numTrees number of tree nodes in parallel
- Since we set featureSubsetStrategy = auto, total features considered while creating decision tree is sqrt total number of features
- Selects best features for creating decision tree from square rooted sample of features that give max information gain

- While selecting features, categoricalFeaturesInfo parameters get higher priority as compared to all others
- Catergorical features are sampled at a node and branched on these values
- For features having continuous data, branching happens on the basis on inequality
- Splitting at each level happens only if information gain > 0.0 (through gini)
- After first split, goes to step 3 and repeats for maximum of maxDepth

The model created has prediction 1.0 or 0.0 for different parameters

Running Time -

Training Time : 38 min : 11 machines 52min: 6 machines
Prediction : 10 min : 11 machines

Accuracy-

Number of Parameters : 19	Accuracy : 81.2%	// increasing depth
Number of Parameters : 19	Accuracy : 79.3%	// using dense matrix
Number of Parameters: 10-950	Accuracy : 72%	// using sparse matrix
Number of Parameters: 19	Accuracy : 77%	// reducing depth to 5

Data partitioning and assignment to workers -

Training & Prediction : Data is being partitioned into 80 partitions
The 70% data gets partitioned for building each model. The result from this data is consolidated and final prediction is obtained. This ensures well distributed running of data and creation of decision trees according to the data partitioning.