CS 6240 Final Project Report

# Technology Stack:

Apache spark + Python => Distributed computing

MLLib and Scikit learn => Machine learning libraries

PySpark is very helpful in implementing the program in distributed way. The built-in functions help us to implement the MapReduce jobs easily. Also, Saving the Models as RDD and reading it again for prediction is easy in PySpark. Also, Scikit learn is the best machine learning library.

# Design Outline:

1. Data Engineering/Cleaning
2. Model Training

M1

RDD 0

1. Prediction

Replicated RDD

Input File:

Training data

Cleaned Data



RDD 1

M2

RDD N

Mn

M1

RDD 0

Replicated RDD

Final Prediction

Input File:

Test data

Cleaned Data

RDD 1

M2

RDD N

Mn

Model Training:

First we clean the input data, replicate it using flatMap, so we get bigger RDD, now we do GroupBy key so the RDD is grouped into smaller RDDs, and we apply ML algorithm on each RDD to get N different mode.

Prediction:

We read and clean the data, Replicate using flatMap and we do GroupByKey to get N RDDs. We train each RDD in each Model and we get the prediction. Finally, we aggregate the prediction to get the final prediction.

# Data Engineering/Cleaning:

Processes involved in Data Engineering includes,

## Removing less important columns:

We are dropping the columns that are,

1. IDs -> Doesn’t have any mathematical properties so they won’t help in learning.
   1. Ex. SAMPLING\_EVENT\_ID, LOC\_ID, OBSERVER\_ID, GROUP\_ID, SUBNATIONAL2\_CODE
2. Categorical Columns -> Columns with multiple possible values.
   1. Ex. BCR, BAILEY\_ECOREGION, OMERNIK\_L3\_ECOREGION
   2. COUNTY, COUNTRY, STATE\_PROVINCE can be dropped since they are represented by Longitude, Latitude.
   3. YEAR contains only one value so its dropped as well.
3. Filtering Rows where PRIMARY\_CHECKLIST\_FLAG is not set to true.
   1. This is a way to get unique group id, we are interested only in rows with unique group ID
4. One-Hot Encoding -> Encode categorical columns with finite possible values So the ML algorithm learns better.
   1. COUNT\_TYPE: Only 20 possible protocols are present. So we encoded it. This is done by adding 20 Boolean columns Where only the particular protocol gets 1 others get 0.
   2. TIME: Column is Divided into four columns each represents 6-hour slot.
5. Manipulate certain values
   1. Longitude and latitude are converted into xyz plane. Because, WGS-84 representation doesn’t work well because of the earth’s spherical shape, So even if two places are too close, one half will gets positive value and another one gets negative value.
   2. ELEV\_GT, ELEV\_NED -> both are elevations so we have taken average and dropped it.
   3. Missing values in CAUS\_PREC, CAUS\_SNOW, CAUS\_TEMP\_MIN, CAUS\_TEMP\_MAX, CAUS\_TEMP\_AVG are replaced with respective values in extended covariates, So the columns in extended covariates are dropped.
   4. NLCD\_\* these columns are dropped.
6. Handle Missing values
   1. Any column that has ‘?’, its replaced with 0. ‘x’ in any of the bird column is replaced with a random number between (2,10) because the observer forgot the bird count. It can’t be 0.

Finally, every value is converted into float and target column in converted into binary (seen or not seen).

Now, we are converting this into sparse vector representation. Each record is represented as following:

[#of total Index, Array[Index of non-zero elements], Array[corresponding non-zero]]

This improves the efficiency of the program. We have only around 60-70 columns with non-zero values in the 997 columns in the preprocessed data. We have used PySpark to do the preprocessing step and feed it into model training program. Every function in a preprocessing step is included in a single custom function that is applied using map in spark. We just added logic to drop or manipulate values in the columns.

# Model Training

The cleaned RDD is replicated n times. We used flatMap on this RDD that duplicates the records and returns a list of records that are assigned keys from 1..n.

Rdd.flatMap( x => [(0,x), (1, x), (2, x), … (n, x)])

Now, we have a bigger RDD. This RDD is going to be grouped by Key so each group gets an RDD of whole input data.

Rdd.groupByKey()

Now, we have n groups of RDD, this RDD is given to different ML algorithm to build a model.

And these models(RDDs) are written into file.

## Different Models

## Model Tuning

# Prediction

The process is similar to the Model training until groupByKey.

In prediction, we are going to give the RDD to each Model and get the prediction.

The output from each model is aggregated into final result.

# Results