Final Project Report

Class: CS6240
Section: 2
Professor: Nat tuck
Team Members: Mukul Bichkar & Sushil Thasale

Objective
In this project, we are solving a classification problem. To predict whether a Red-winged Blackbird (Agelaius phoeniceus) will be spotted or not. We were provided with labelled and unlabelled data.

Steps Followed: -

I. Introduction to Data Mining

In order to solve this problem, our first step was to understand data mining steps and how to apply it in scope of map reduce. The slides on nu online proved to be extremely helpful for this. I even completed a course on udacity (https://www.udacity.com/course/classification-models—ud978). It helped in understanding the classification model we were supposed to choose. So, what I learnt from this course was that we were supposed to use a binary classification model i.e. the outcome of our prediction will be either 0 or 1. We also learnt a little about Alteryx a platform that allows to visualize data and build model. This also led us in exploring the Weka GUI. So, the models that we were supposed to use for our problem i.e. binary classification were: -
a. Logistic Regression
b. Decision Tree
c. Naive Bayes
Also, we learned about confusion matrix and how it used to find the accuracy of the model that we are using. (http://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/)

II. Understanding Domain and Data.

Domain

For understanding domain, we went through many websites to help us understand about Red-Winged blackbird. Our finding can be summarized as below:-

1. The red-winged blackbird is found mostly in North and Central America. It also migrates to south of Mexica and Southern United states. So, it was important to know the state and location of the bird to predict it accurately.
2. Red-winged birds prefer wetlands, and habitats where there is both fresh water and salt water marshes. It prefers aquatic biomes near brackish water. So, this was another parameter we thought would be important in prediction.
3. Also, these species maintain a distance with human beings and prefer open areas like fields, marshes, agricultural areas. So, knowing about population density would be really useful.
4. The fourth important factor that affects all living being is the diet, so knowing about the ecology and diet about a place would be useful.
5. Also, since these birds are diurnal i.e. active during day time. Time was one of the important factors in training the data.

After gathering sufficient domain knowledge, we proceeded to step 2 i.e. understanding data

Data

For understanding the data, we read all the documentation provided on google drive (Provided in project docs) and understood all the parameters mentioned. Then we made an educated guess (based on domain knowledge) about the parameters that would be really useful in the prediction. The parameters that would be helpful were: - Latitude, Longitude, month, time, effort_hrs, group_id, caus_prec, caus_snow, caus_temp_avg, caus_temp_min, caus_temp_max, housing density, housing vacant, population persq mile, distance from flowing and standing fresh and brackish water. To verify our assumptions and also understand more about the data, we tried to unzip the labelled.bz2 file since it exceeded the memory size of the machine we could not unzip it. So we wrote a simple sampling job that read in the labelled.bz2 file and emitted random records from it. Different sampling rates like 1%, 4% and 20% were used.

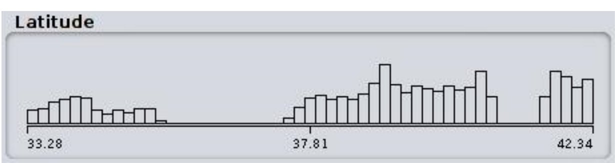
Pseudo-code for Sampling (map-only job)

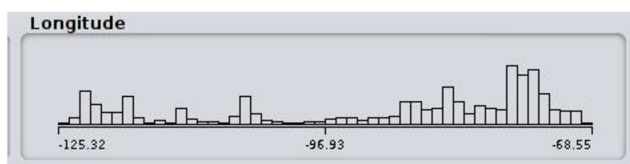
```
map(record)
generate a random number between 1 to n
if random_number < specific_number
        emit(null, val)
exit
```

Another spark job was used to extract the desired fields from the sampled data. The processed data was then loaded into a weka GUI and the nature of graphs for all the parameters were studied based on which attributes were chosen:-

1. ***Latitude*** 2***. Longitude***

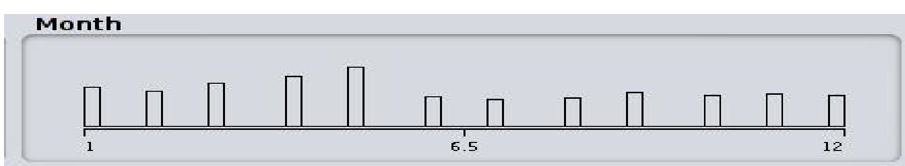
Variation in graph is seen because of different locations. And red-winged bird found in different locations.





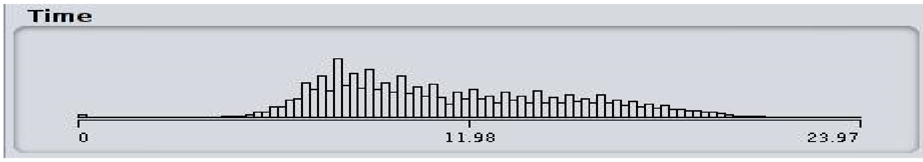
3. ***Month***

Different seasons occur within different months. So, month plays an important role. For example, breeding season begins in early spring and continues till mid-summer.



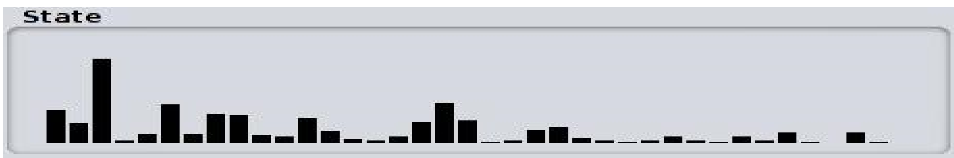
4. ***Time***

The nature of the graph verifies the diurnal nature of these birds.



5.***State***

Since the birds are found in certain states and also they migrate to particular states depending on the seasons, state can be a good indicator for prediction.

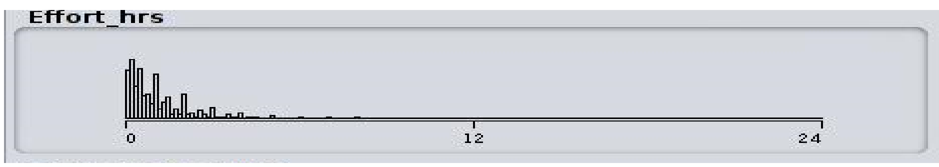


6. ***County***

No graph available due to large number of records.

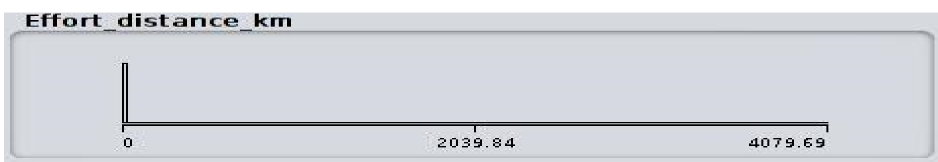
7. ***Effort Hours***

More the effort hours more the chances of spotting a red-winged bird. Obvious choice.



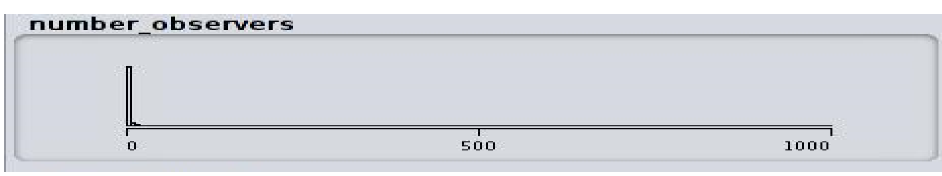
8. ***Effort distance***

More the effort distance more the chances of spotting. Again an obvious choice.



9. ***Number of Observers***

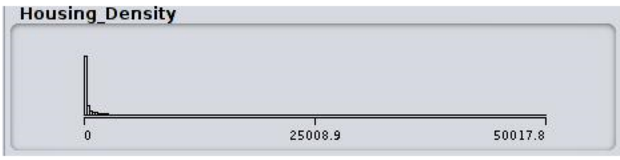
Number of observers can affect the number of spotting’s as well as chances of being spotted.

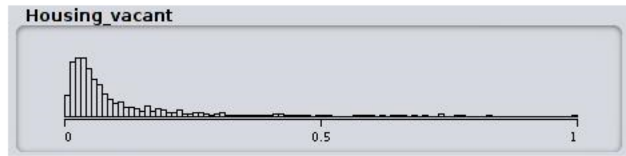


10. ***Population per square mile***

The population per square mile, housing density and housing vacant all affect the probability of spotting the bird as these birds tend to be away from human beings.

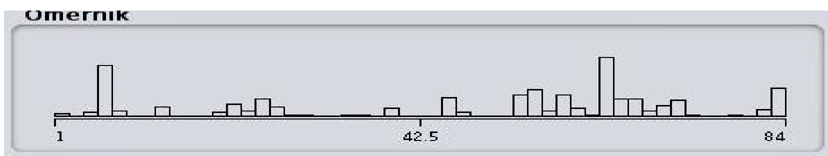
11***. Housing Density*** 12. ***Housing Vacant***



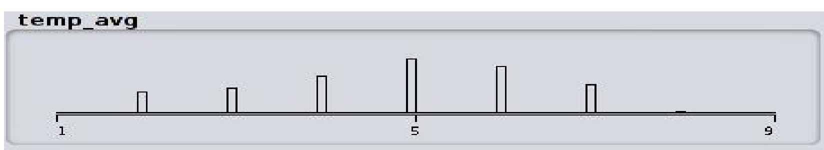


13. ***Omernik L3 Ecoregion***

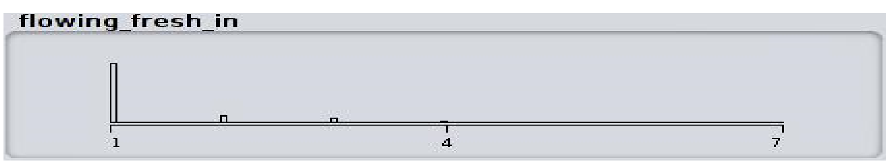
Ecoregion is important from soil, water, food and vegetation point of view. It matter because during the breeding season, red-winged blackbirds eat mostly insects and other invertebrates. At other times of the year, they feed themselves on weed seeds, crop grains.



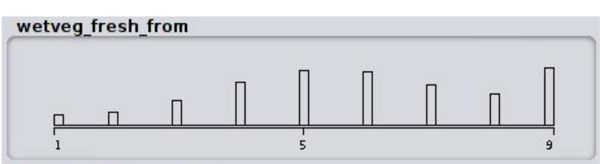
14. ***Average Temperature***

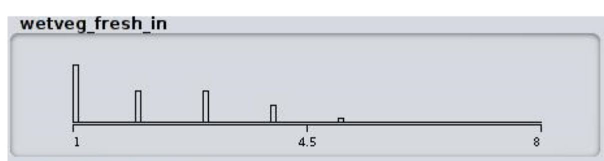


15. ***Flowing Fresh Water In***

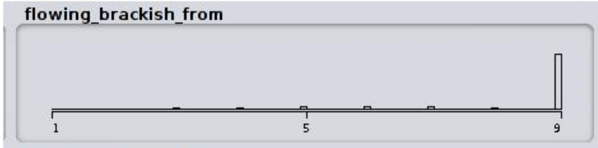


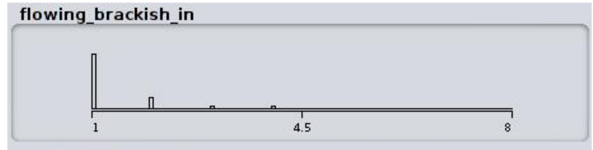
16. ***WetVeg Fresh (from and in)***



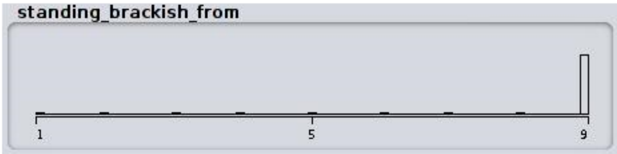


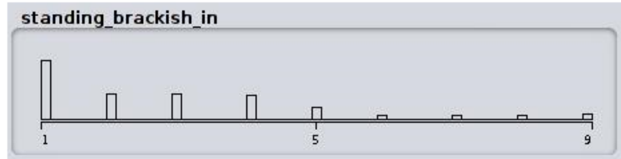
17. ***flowing brackish (from and in)***





18. ***standing brackish (from and in)***





All the above fields i.e. distances from water bodies are important because the red winged bird tend to stay near marshes, swamps and prefer aquatic biomes.

III. Algorithm Design and Pseudo Code

The Algorithm we thought to tackle this classification problem is as follows:-

1. Use multiple models to train the data.
2. Pass the test data to each model.
3. Finally find the prediction value i.e. majority result either 0 or 1.

Pseudo-Code:-

1. ***Sampling Event Details class***

//This class contains all the attributes of importance
//initialize the variables using correct data types
  Default constructor ():
        Initialize the fields.
  readFields(in):
        fields.readFields(in)
  writeFields(out):
        fields.writeFields(out)
  toString():
        return the String of this object

2. ***DataHandler class***

//rec is a String from input data and type is an int indicating training or testing
Parse(String rec, int type):
        Preprocess the desired attributes from the rec and handle missing values appropriately.
        Return an array of desired values after processing.
getSamplingDetails():
        return SamplingEventDetails object.

3. ***Partition Data based on the key***

RandomKeyPartitoner (randomKey, samplingEventDetails):
        Return randomKey % number of reducers

4. ***InstanceHandler class***

Initialize the instances
initInstance():
        Declare the attributes of interest
        Add them to weka attributes
getInstance():
        return instance

5. ***Job-1 PreProcess and train models***

TrainingMapper(record):

Setup():

get totalModels // Fetch the number of models set in the configuration variable

dataProcessor = new DataHandler() //Initialize a DataHandler Obj. which preprocesses the data

Map(key, value, context):

samplingEventDetails = dataProcessor.parse(value, TRAIN) //TRAIN is for training mode = 1

Random_number //generate a random number

Emit (random_number, samplingEventDetails)

TrainingReducer(record):

Setup():

get totalModels // Fetch the number of models set in the configuration variable

classify(trainingSet, model):

pass the training set to each model so that each model can train itself on the training set.

```
switch(model):
    case 1:NaiveBayes(trainingSet)
      case 2: RandomTree(trainingSet)
    case 3: RandomForest(trainingSet)  //Example
```

Reduce(key, values, context):

```
        //Create a training set
    trainingSet = InstanceHandler.initInstances()
      For(i=0 to values):
          trainingSet.add(instances)
    For(i=0 to totalModels):
          Model = classiy(trainingSet, i)     //Build a model and train it on training data
    Write output to disk
```

//End of Job-1

6***. Job-2 Validate the build and tested models against testing data***

TestingMapper(record):

Setup():

get totalModels // Fetch the number of models set in the configuration variable

dataProcessor = new DataHandler() //Initialize a DataHandler Obj.

Map(key, value, context):

samplingEventDetails = dataProcessor.parse(value, TEST) //TEST is for testing mode = 2

Random_number //generate a random number

Emit(random_number, samplingEventDetails)

TestingReducer(record):

Setup():

```
    get totalModels // Fetch the number of models set in the configuration variable
        get totalTypes //Fetch the type classification types set in configuration variable
    set instances // initialize the Instances obj
    classifiers //fetch the classifers an store it in a list
    for(i=0 to totalTypes):
```

```
        for(model=0 to totalModels):
            classifiers.add(getClassifier()) //getClassifier method returns the already
trained classifier
        end
        end
```

getClassifier(model,type):

    fetch the model i.e. read it from disk and return the already trained model

classify(trainingSet, model):

pass the training set to each model so that each model can train itself on the training set.

```
    switch(model):
        case 1:NaiveBayes(trainingSet)
          case 2: RandomTree(trainingSet)
        case 3: RandomForest(trainingSet)  //Example
```

Reduce(key, values, context):

```
        presentProbability // constant to record positive outcome i.e. bird sighting
       absentProbability // constant to record negative outcome i.e. bird not sighted

    for(i=0 to values):
        get Instance
        for(j=0 to classifiers):
            get probabilities in a list
            increment the count of presentProbability by fetching 0th element from
list
            increment the count of absentProbability by fetching 1st element from
        list
              end
      end
    //Make Prediction
    if presentProbability > absentProbability:
        assign "1" as prediction
      else
        assign "0" as prediction
    end
    String output = sampl_id + prediction //prediction is either 0 or 1
    emit(output, empty String)
```

// End of Job-2

6.***<u>Driver Program</u>***

classifyData(): //Used for Validation of Test Data

```
    setMapper(TestingMapper)
    setReducer(TestingReducer)
    set NumofPartitioner(n) //where n is fetched from args
```

<u>trainData():</u> //Job for creating model and training it

```
    setMapper(TrainingMapper)
    setReducer(TrainingReducer)
    set NumofPartitioner(n) //where n is fetched from args
```

<u>Run():</u>

    Call trainData() and classifyData()

<u>Main();</u>

    Call run()

IV. Why & How Pre-Processing

Pre-Processing is done to fetch the desired fields of interest and also to eliminate the missing values or malformed values like “?”. From the research I did I found that in case of Classification the records that have missing values are ignored. But in case of our data some attributes like ***flowing brackish (from)*** are 85% missing, so ignoring those records altogether are not feasible as they can contain other valuable information that can help in training the model and getting more accurate prediction results. We replaced the missing values with -999 just an encoding value.

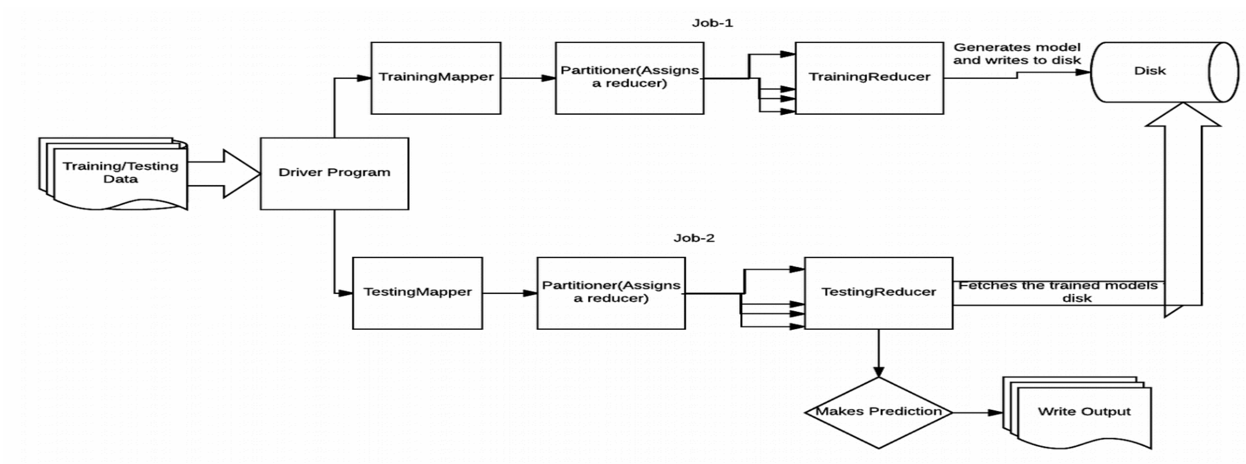


Diagram Explaining the Flow of Data

<u>Summary of models used and Prediction Accuracy:-</u>

Sr. No.	Model	Accuracy for Sample Data	Attributes used
1	NaiveBayes	71 %	month, time, caus_temp_avg, housing density, housing vacant, population persq mile, distance from flowing and standing fresh and brackish water
2	RandomTree	75 %	month, time, caus_temp_avg, housing density, housing vacant, population persq mile, distance from flowing and standing fresh and brackish water
3	Random Forest [depth=15,	76 %	month, time, caus_temp_avg,

	trees=15, features = 19]		housing density, housing vacant, population persq mile, distance from flowing and standing fresh and brackish water
4	All three combined i.e. NB, RandomTree and Random Forest. [depth=15, trees=15, features = 19]	75%	month, time, caus_temp_avg, housing density, housing vacant, population persq mile, distance from flowing and standing fresh and brackish water

References:-

a. https://www.udacity.com/course/classification-models—ud978
b. http://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/
c. **http://eol.org/pages/1052017/details**
d. **https://www.allaboutbirds.org/guide/Red-winged_Blackbird/lifehistory**
e. **http://www.ccis.northeastern.edu/home/yzhao/slides/Intro_no_pause.pdf**
f. **http://www.cs.waikato.ac.nz/ml/weka/**