CS 6240: Final Project

Goal: Use data mining to predict sightings of the Red-winged Blackbird in birding checklists.

After learning about various classification and prediction techniques, we will now apply our expertise to a real data mining challenge. This challenge is organized as a friendly competition between teams, each having two members. Talk to your class mates and use the online forum to form teams. To avoid difficulties when scheduling the final presentations, both team members either have to be in the same section, or have to agree to be flexible to present their work in either section (as determined by the instructor).

Each team has to create all deliverables from scratch. In particular, it is not allowed to copy another team's code or text and modify it. (If you use publicly available code/text, you need to cite the source in your code and report!) You also cannot just replicate another team's approach. Every team has to develop their own solution independently.

The project has two separate assignments with different due dates: a report and the presentation slides. **Make sure you submit each to the appropriate assignment.**

Project report: Please submit your solution through Blackboard by the due date shown online. For late submissions you will lose one percentage point per hour after the deadline. This HW is worth 100 points and accounts for 15% of your overall homework score. To encourage early work, you will receive a 10-point bonus if you submit your solution on or before the early submission deadline stated on Blackboard. (Notice that your total score cannot exceed 100 points, but the extra points would compensate for any deductions.) Always package all your solution files, including the report, into a single standard **ZIP** file. Make sure your report is a **PDF** file.

Please name your submitted ZIP file "**CS6240_first_last_Report**", where "first" and "last" are your first and last name, respectively. For each program submission, include complete source code, build scripts, and small output files. Do not include input data, output data over 1 MB, or any sort of binaries such as JAR or class files.

To enable the graders to run your solution, make sure you include a standard Makefile with the same top-level targets (e.g., *alone* and *cloud*) as the one Joe presented in class (see the Extra Material folder in the Syllabus and Course Resources section). You may simply copy Joe's Makefile and modify the variable settings in the beginning as necessary. For this Makefile to work on your machine, you need Maven and make sure that the Maven plugins and dependencies in the pom.xml file are correct. Notice that in order to use the Makefile to execute your job elegantly on the cloud as shown by Joe, you also need to set up the AWS CLI on your machine. (If you are familiar with Gradle, you may also use it instead. However, we do not provide examples for Gradle.)

As with all software projects, you must include a README file briefly describing all of the steps necessary to build and execute both the standalone and AWS Elastic MapReduce (EMR) versions of your program.

This description should start with unzipping the original submission, include the build commands, and fully describe the execution steps. This README will also be graded.

Presentation slides: Please submit your solution through Blackboard by the due date shown online. For late submissions you will lose one percentage point per hour after the deadline. This HW is worth 100 points and accounts for 10% of your overall homework score. Make sure your presentation is a **PDF** file.

Data Set and Analysis Task

You will find two data sets (**labeled.csv.bz2** and **unlabeled.csv.bz2**), together with a reference document (**ebird-ref-data_rev2014.pdf**) explaining the data, here:
https://drive.google.com/drive/folders/0BxLUDit8x6n1VXRTOC1TdEpCNEE

Note that the reference document describes the data in its original file structure, where the different variables (called covariates in the document) of an observation record were distributed over three files:

1. Checklist—16 columns describing the sampling event (date, time, lat, long, duration, distance traveled, protocol, observer id, etc.), plus variables for all the species in the data set. There is one column per species, with each column listing the number of birds observed.
2. Core covariates—were suggested by domain experts as most important for most of the species.
3. Extended covariates—include extensive statistics about habitat configuration, fine-grained climate measurements, and observer expertise.

This partitioning, and also the partitioning by year, was an artifact of the limitations of the computation environment used by many scientists. (E.g., they like to open a file using Excel to take a look at the data.) Since this course covers big-data processing, we combined the information across all the individual files to let the data appear in all its intended un-partitioned beauty. In particular, all covariates appear together in a single line for each record.

The goal is to predict the presence or absence of the Red-winged Blackbird (*Agelaius phoeniceus*) in each birding session as accurately as possible. This bird is coded in the 27th column of the data sets; its name is in the header for that column. You can use the *labeled* data in any way you like for model training, parameter tuning, and testing. For final evaluation, the labels predicted for all records in the *unlabeled* data set have to be submitted. (There the value for the bird of interest contains only "?")

Requirements

Model Training Program

1. Use the labeled training data set in any way you like to train, tune, and test your model.
2. Use a MapReduce or Spark parallel processing program upon distributed data. This should be a single program composed of one or more jobs. Make sure your program clearly improves over the single-processor solution by overcoming a computation or memory bottleneck. To evaluate your design, ask yourself: "Does it achieve non-trivial speedup?" and/or "Is it able to use more

of the training data (rows and/or columns) compared to an approach that has to fit all training data into memory on a single machine before being able to train a model?"

3. Choose any classification or prediction technique, including ensemble models, to build a prediction model. Note that we are solving a classification problem, hence the labels predicted by your model should be either 0 or 1. If you use a technique that returns probabilities, make sure you threshold them accordingly when producing the final predictions.
4. You may use any data-mining related open source libraries, languages, or environments as long as they are called from your single program.
5. You must write the program so that it is out-of-scope to use a pre-packaged solution (e.g., H2O). If you are uncertain whether an external component is allowed, ask on the online forum. A good test is to ask yourself: "Did I solve any non-trivial distributed computation problem myself, e.g., determine how input is partitioned or how load is assigned?"
6. The resulting model must be persisted for (possibly repeated) use by the prediction program.

Note about using Spark's MLlib: You are allowed to use MLlib. However, the project cannot just be "I loaded the data, trained model XYZ from MLlib, and here is my result." If you use MLlib, you need to understand and discuss in your report, how MLlib implements the training process. In particular, how many partitions are created? Is data being shuffled? How many iterations are executed (for methods that have multiple iterations)? You also need to find out, how to affect performance. In particular, change the number of partitions or the actual partitioning itself and report how this affects running time.

Note about using Weka: You are allowed to use Weka libraries in your program. For example, you can use Weka to train individual models of an ensemble in the different Mappers or Reducers of your program.

Prediction Program

1. Use MapReduce or Spark to implement a parallel prediction program that uses the previously generated, persisted model to predict for each record in the unlabeled data whether or not the bird of interest will be reported in a checklist. Stated differently, this program should replace the "?" label in the unlabeled data by either "0" or "1", based on the model's prediction.

Validation

1. As part of the model training process, most data mining techniques require tuning of parameters. Use the given labeled data in any way you deem appropriate to do this. A standard approach is to partition the labeled data set randomly into 3 sub-sets: training data to build the model, validation data to tune its parameters, and test data to determine it accuracy. Often about 50-70% is used for training and 15-25% each for validation and testing. However, these are not fixed rules. Also, since we evaluate on the unlabeled data, you do not need a separate test set and hence can split the labeled data into training and validation data only, e.g., around 80% for training and around 20% for validation. Again, these are just generic recommendations.
2. We will use accuracy (ACC) for the final evaluation.

Testing

1. Once you found a satisfactory model, execute your prediction program on the unlabeled test data (i.e., unlabeled.csv.bz2) to produce your final predictions.
2. Your output file must be named as specified in the deliverables and contain the following header line:
   SAMPLING_EVENT_ID, SAW_AGELAIUS_PHOENICEUS
   followed by lines with this format:
   sampling-event-id, prediction
   where prediction is either 1 or 0. Value 1 indicates that the bird sighting was predicted for that session and 0 indicates the opposite. There must be exactly one line for each and every respective line in the unlabeled input file. Make sure that the predictions are saved in the same order as the lines in the unlabeled input file.

These formatting rules are strict for both grading your homework and determining the competition winner. To assist you in validating and tuning your model and also making sure your prediction file format is correct, a simplified version of the scoring program is available in Blackboard. You may use and modify this code but, as written, it represents the de facto scoring algorithm.

Final Project Report

The final report should fully describe your prediction methodology as well as the design of your programs. This should be concise (see page limit in deliverables section), but comprehensive, as it serves as project documentation to assist graders in evaluating its quality. Also report on your validation results using (and describing) your quality measurement. Make sure your report covers the following points:

- Type of model used and parameters explored for it.
- Pseudo-code, possibly accompanied by examples and images, illustrating the steps of the distributed computation. It should make clear how data and computation are partitioned and assigned to the workers. This needs to be done for both model training and prediction.
- If you performed any pre-processing, briefly summarize what you did and why you did it.
- Accuracy numbers for different parameter settings you explored. Use your own judgement in choosing, which results to include.
- Running time and speedup results for the model training and the prediction phase. You do not need to include numbers for every model or model instance you tried. Use good judgement to select a few numbers.

Presentation

Each team will present highlights of their solution to the class during final exam week. The allotted time is 6 minutes with another minute for questions. Make sure you submitted a PDF version of your slides by the deadline. (See also the file naming convention in the deliverables section.) This will enable the instructors to download all of them to a designated presentation laptop, avoiding time loss due to transition problems. For compatibility, all presentations have to be in PDF format. Note that this might prevent certain fancy animations. Consider the following recommendations when preparing your presentation:

- Think of this as a hiring talk. You are interviewing for a high-paying job with your favorite Big Data company and the class is a group of experts from this company who will decide if they should hire you or not. How do you show this technically versed audience in a few minutes that you are capable of solving a Big Data problem? However, be careful and avoid over-selling!
- Since everybody knows the data and the problem, you do not need to talk about it. Dive right into the *solution* you found.
- Focus on a few carefully selected results that best showcase your expertise as a MapReduce or Spark programmer. The presentation should give an overview of your approach, but you want to spend most time on the interesting aspects that you think make your solution special or highlight your analytical and creative thinking.
- Present graphs or tables showing concrete measurements. Think about which ones you need in order to make a case that you (1) solved the problem well and (2) your code is scalable or in other ways improves over the single-processor solution.
- Very briefly present your validation accuracy results for comparison to the test results, which will be published by the instructors.

Points will be awarded based on slide design and quality of oral presentation. <u>Exceeding the 6-minute time limit will result in automatic deduction.</u> It is important that you practice presentation timing.

Competition (extra credit)

To raise the stakes and add some fun, there will be a competition for the best predictions across both sections of this course. The members of the winning team will receive a 20-point credit on the presentation assignment. Second to eighth place will receive 15, 10, 7, 5, 3, 2, and 1 points in credit. Total score is not capped, i.e., the credit might result in some teams with more than 100% on the presentation—with the corresponding impact on the final course grade. Team ranking will be determined based on prediction **accuracy** on the unlabeled data:

Accuracy = (CorrectPositives + CorrectNegatives) / TotalObservations.

We would like to see the top 4 winners demonstrate their project programs performing the model building and prediction. This will take place outside of class in a scheduled 10-15 minute meeting during finals week. At least one team member must attend and demonstrate the project from source code build through predicting the unlabeled data. In the unfortunate case that a top team is unable to demonstrate their project, they will vacate their position and all lower scoring teams will rise one place. Being unable to demonstrate your project does not affect your presentation score, just your qualification to receive extra credit points.

Deliverables for Project Report

1. The project report as discussed above. The report cannot have more than 6 pages in the font size and layout of this assignment. The shorter, the better—just make sure all the important information is included in a concise manner. (1 PDF file; 40 points)
2. The source code of your programs, including an easily-configurable Makefile that builds your programs for local execution. **Make sure your code is clean and well-documented. Messy and hard-to-read code will result in point loss.** In addition to correctness, efficiency is also a criterion for the code grade. However, *low prediction accuracy* does not result in any deduction. (40 points)
3. The syslog (or similar) files for a successful run of your model training program on the full labeled data set. (5 points)
4. The syslog (or similar) files for a successful run of your prediction program on the full unlabeled data set. (5 points)
5. Final prediction output file in CSV format for the full unlabeled data, as specified in the requirements. **Make sure you name the result file according to your team members as follows**: use the last names of all team members, sorted alphabetically, as the file name. For instance, the two-person team Joe Smith and Mary Miller would use file name MillerSmith.csv. (10 points)

Deliverables for Presentation

1. The presentation slides as discussed above. **Make sure you name the file according to your team members as follows**: use the last names of all team members, sorted alphabetically, as the file name. For instance, the two-person team Joe Smith and Mary Miller would use file name MillerSmith.pdf. (1 PDF file)