Classifying Brazilian Cities Based on Weather Data Using Apache Spark

Benjamin Khuong, Paul Kim, and Tomohiko Ishihara
University of San Francisco, San Francisco CA 94105, USA
bkhuong, ykim57, ttishihara@usfca.edu

1.0 Introduction

1.1 Data Description

Our dataset is comprised of hourly observations of various weather indicators at 122 weather stations across south-east Brazil from the years 2000 to 2016. The states included are Rio de Janeiro, Sao Paulo, Minas Gerais, and Espirito Santo. Some major cities included are Sao Paulo, Rio de Janeiro, and Belo Horizonte. In this dataset, some typical weather indicators can be found such as air temperature, dew point, and wind speed, in addition to indicators such as solar radiation and wind gust intensity. A data appendix can be found below.

1.2 Data Pre-Processing Goal

Our goal for this dataset was to build a distributed framework and predict Brazilian cities based on existing weather data using a variety of algorithms from the Spark ML library (Multinomial Logistic Regression, Random Forest Classifier and Support Vector Machines). We then compared the F1-scores, training, and prediction speeds of these algorithms to determine which performed best in a parallelized system.

The features we used to predict the city includes date-time, atmospheric (air pressure, solar radiation, humidity, etc.), temperature and wind data. We have removed all features that contain information about location (latitude, longitude, elevation, etc.). We also dropped features for precipitation amount and wind speed as around 10% of all rows were missing these features. In the end, we had a dataframe with 9,507,884 observations and 28 features.

Each label was converted to a number value ranging from 0 - 122 based on the 122 different cities so that they could be used by the machine learning algorithms. The hour and month features were converted to a sparse vector of 24 and 12 elements respectively. This helped to eliminate any ordinal relationships, treating each data point as a binary feature. Afterwards, all the features were combined into a single feature vector so they could be fed into the machine learning algorithms (see figure 1 below). Finally, the dataset was divided into a

90/10 train test split with a training sample size of 8,558,132 and test sample size of 949,752 observations.

```
+--------------------+-----+
|            features|label|
+--------------------+-----+
|(51,[14,39],[1.0,...|  4.0|
|(51,[14,39],[1.0,...|  4.0|
|(51,[14,39],[1.0,...|  4.0|
|(51,[14,39],[1.0,...|  4.0|
|(51,[14,39],[1.0,...|  4.0|
+--------------------+-----+
```

Figure 1. Final Input for ML Algorithms

2.0 Setup

2.1 S3 Data Storage

Our data was stored on Amazon Web Services Simple Storage System (AWS S3).

2.2 MongoDB Setup

For our distributed data system, we set up 2 shards with 3 servers each (one primary and two secondary servers), 3 configuration servers, and one mongos server for a total of 10 t2.medium machines on AWS. Each t2.medium machine has 2 vCPUs and 4 GB of memory, for a total of 20 vCPUs and 40 GB of memory.

2.3 EMR Setup

For our distributed computing system, we set up 1 master m3.xlarge node and 9 worker m3.xlarge nodes, for a total of 10 m3.xlarge nodes. Each m3.xlarge server has 4 vCPUs and 15 GB of memory, for a total of 40 vCPUs and 150 GB of memory.

3.0 Modeling

We employed two machine learning algorithms to predict the cities: Logistic Regression and Random Forest Classifier.

3.1 Logistic Regression

Logistic Regression was our first pass at prediction due to its simplicity and utility as a baseline model. We used a regularization parameter of 0.1, maximum number of iterations of 100, and fit intercept of True. Our Logistic Regression model trained in 14 minutes and 23 seconds, with an accuracy of 5.52% and an f1 score of 2.31%. Both of these metrics are low, but outperform random guessing.

3.2 Random Forest

The next model we chose was Random Forest, due to its predilection towards parallelization. We used a maximum depth of 10 nodes and 10 trees. The run time of Random Forest was 1 minute and 28 seconds, much faster than that of Logistic Regression. Random Forest also vastly outperformed Logistic Regression, with an accuracy of 26.41% and and F1 score of 23.33%.

4.0 Results

The results of our models can be found in the table below:

Model	Accuracy	F1 Score	Time
Logistic Regression	0.055	0.023	14 min 23s
Random Forest	0.264	0.233	1 min 23s

Table 1. Comparing Model Metrics

A confusion matrix for our Random Forest model can be found below:

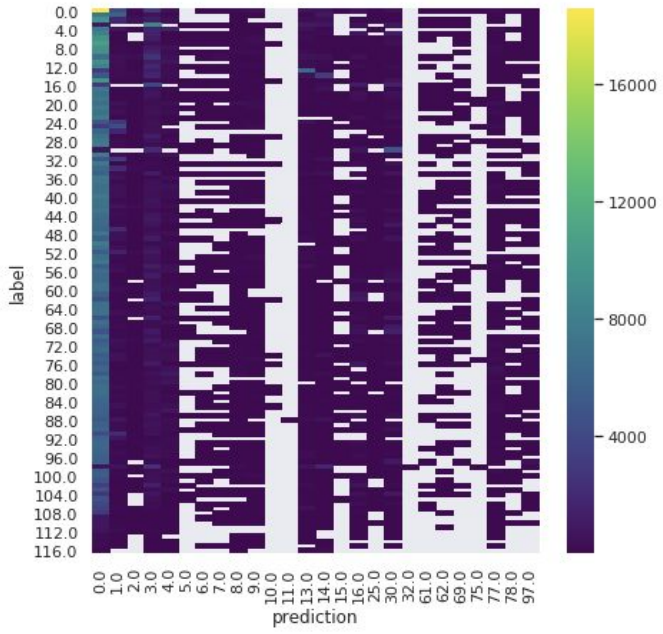

Figure 2. Confusion Matrix for Logistic Regression

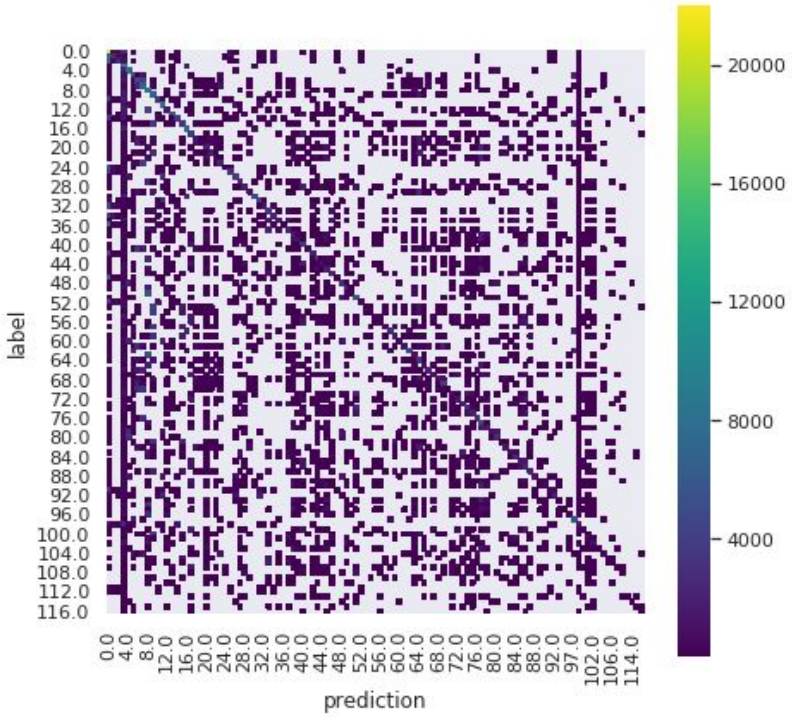

Figure 3. Confusion Matrix for Random Forest

5.0 Conclusion

Most of the takeaways from this project were with regards to the setup process rather than the machine learning itself. Our data was relatively clean, but we still ran into issues when reading the data and casting the data types. One crucial breakthrough was the realization that our data collection included empty strings that were not recognized as nulls in Spark SQL that had to be removed within a mongo shell. Once we were able to remove these documents/fields that had empty strings, we were finally able to run the machine learning algorithms.

Overall, setting up our distributed data system, distributed computing network, and connecting the two was the most challenging part of this project. Most of the time spent on this project was during the setup process; however, considering our relatively short run times, we clearly saw the benefits of using distributed systems.