Activity Recommendation System

Avaljot Khurana[1], Avikaran Singh Bedi[2], Navya Sai Paladugu[3], Neeharika Cheruku[4], Swaroop K Pydisetty[5]
Department of Computer Science, Erik Johnsson School of Engineering, UTD

Email: [1]ask150930@utdallas.edu, [2]asb150330@utdallas.edu, [3]nxp153030@utdallas.edu, [4]nxc151230@utdallas.edu, [5]skp150330@utdallas.edu

***Abstract*— A System that predicts the best leisure activity in your vicinity at a given place and date based on the weather conditions. With the help of state-of-the-art machine learning algorithms, we predicted weather for the next day. Based on the weather prediction, nearest activities are filtered out which favor weather conditions of that day.**

Keywords—Activity Recommendation System, Random Forrest Model, Weather Predictions, Scala, Apache Spark, Big Data, GHCN Dataset, OpenStreetMap Community

I. Introduction

The Internet has no dearth of content. The challenge is in finding the right content for yourself. Search engines help solve the former problem; particularly if you are looking for something specific that can be formulated as a keyword query. However, in many cases, a user may not even know what to look for [1]. Often this is the case with finding enjoyable activities where users end up browsing the things that might interest them without taking the weather conditions into account. This project solves the problem by recommending best activity in the vicinity taking weather conditions into account.

Temperature is predicted using Decision Tree Regression[5]. The Temperature data is non-linear in nature. Therefore, the best prediction can only be achieved through regression model. Whereas, pressure data is classified into five classes to get minimum error. Random Forest Classifier[6] is employed to predict the class for the next day. These machine learning tools are available in Apache Spark MLlib—Spark's distributed machine learning library. The code is written in Scala. MongoDB database is put is used to record the interesting points in the vicinity.

II. Background

The following concepts are pertinent to our system:

Apache Spark is an open-source distributed framework for data analytics. It avoids the I/O bottleneck of the conventional two-stage MapReduce programs by providing in-memory cluster computing. Also, it supports both batch processing and streaming data [2].

MLlib is Spark's open-source distributed machine learning library. MLlib provides efficient functionality for a wide range of learning settings and includes several underlying statistical, optimization, and linear algebra primitives [3].

MongoDB is a free and open-source cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schemas.

[5]**Decision Tree** is the most widely applied supervised classification technique. The learning and classification steps of decision tree induction are simple and fast and it can be applied to any domain [4].

[6]**Random forests** add an additional layer of randomness to bagging. In addition to constructing each tree using a different bootstrap sample of the data, random forests change how the classification or regression trees are constructed. In standard trees, each node is split using the best split among all variables. In a random forest, each node is split using the best among a subset of predictors randomly chosen at that node. This somewhat counterintuitive strategy turns out to perform very well compared to many other classifiers, including discriminant analysis, support vector machines and neural networks, and is robust against overfitting [5].

OpenStreetMap (OSM) is a collaborative project to create a free editable map of the world. The creation and growth of OSM has been motivated by restrictions on use or availability of map information across much of the world, and the advent of inexpensive portable satellite navigation devices [6].

III. Datasets

For the weather prediction two datasets from National Climatic Data Center were used:

A. GHCN Data Set

NOAA Global Historical Climatology Network (GHCN) is an integrated database of climate summaries from land surface stations across the globe that have been subjected to a common suite of quality assurance reviews. The data are obtained from more than 20 sources. Some data are more than 175 years old while others are less than an hour old [7].

1) Dataset Selection: Data for 2000-2016 years was downloaded from the NOAA ftp server to filesystem. The most recent file i.e. file for year 2016 was downloaded daily.

2) Dataset Cleaning: We filtered data for the Texas Stations and saved maximum and minimum temperature dataset in a separate file for each station.Further, the dataset was filtered based on the TMAX and TMIN tags. Two files were created for each station. One had the data for maximum temperature(TMAX) while other had for minimum temperature(TMIN).

B. ISD Data Set

NOAA Integrated Surface Database (ISD) consists of global hourly and synoptic observations compiled from numerous sources into a single common ASCII format and common data model [7].

1) Dataset Selection: The data for 2000-2016 years was downloaded from the NOAA ftp server to filesystem.

2) Dataset Cleaning: For the weather prediction algorithm the data was cleaned and reformed. The data was filtered for Texas stations only.We find that not all stations measures the pressure, so stations were filtered out by this criteria.

C. OpenStreetMap Dataset

We used Openstreetmap dataset because it provides available places for activity and the dataset is actively refreshed by Openstreetmap community. On OpenStreetMap, each place is characterized by some characteristic fields (i.e., sport, tourism, etc.) that contains a list of tag.

1) Data Selection: The dataset for the Texas State was downloaded from the OpenStreetMap database.

2) Data Cleaning: The tools provided by OpenStreetMap were run to keep only a given set of required characteristics. We also ran a Spark-Job to keep only the information of the places that were useful and we merged all the tags of a different characteristics on a unique tag list. The merging was done as some of the characteristics on OpenstreetMap doesn't fit our goals (for example the tag in amenity can be related to almost anything).

IV. Building Machine Learning Model

A. Introduction

In order to predict the weather for the next day, we are interested in the temperature (expressed in degree Celsius), and the pressure (expressed in Hecto-pascal).

B. Temperature

Considering the temperature, the values that we used for training come from the first dataset GHCN. From that dataset we only use the values for the maximal temperature and the minimal temperature. This will allow us to easily determine the temperature range for the day. In order to predict the values for the minimal and maximal temperature we separated them and ran twice the same following approaches:

1) Second degree kernelized Linear Regression:

In this approach, we modeled the temperature only by the current date. Since the dataset contained temperature values for every day for multiple years, we wanted to see if we could only use the date as an input to train a model that predicts the temperature for the following date. We thus trained a regression model with a second degree kernel. Unfortunately, this model did not give a good result at all.

2) Decision tree regression model:

One of the reasons that we chose to use a Decision tree is the fact that it can capture non-linearity, which is very useful in our case because the response variable (temperature) is continuous. Due to the fact that a Decision tree partitions the feature space (in our case - date series), the number of bins / leafs will determine how many consecutive days will fall in the same bin. We empirically tested three different number of bins of our Decision tree in order to find the optimal amount of bins that give the best prediction. In our case the best number of bins was 50. From the graph (Figure d) we see that 50 bins are sufficient to predict with a pretty good accuracy.

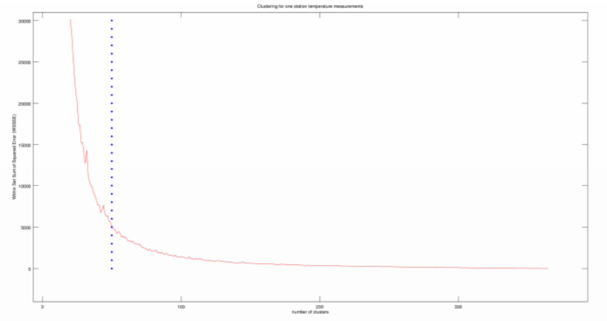

Figure (d): Error vs the number of bins in our Decision Tree model

We analyzed that the temperature can be predicted by looking at the previous measurements. The idea behind this is that the temperature should have a high inertia and that the temperature of today may not differ a lot for the one of tomorrow. However, the problem is that we cannot predict rapid changes in the temperature due to some extreme conditions (storms, hurricanes, etc.), but for cases where the weather does not change, it should work well. The next big question is how many days do we need to take into account? In order to solve this question, we used a moving window of size *n* to get the *n* previous temperature values and use them as features to predict the temperature for day *n + 1*. We tried different values for n and used cross-validation over multiple splits to determine the performance for each value of n. The final value we empirically selected is 7. Therefore, we used the seven previous days to determine the eighth one.

C. Pressure prediction

1) First approach: consider the moving-average model

In our database the pressure is ranging from 960 to 1070. Our initial idea was to predict the pressure for the day and use it to predict the weather afterward. We used the same moving-window model as employed for the temperature prediction. But this method gave an error (RMSE) of 40hPa, which represents around 0.4% of magnitude error. Taking into account, that only values between 960 and 1070 occur in real life, the error represents 80% of the size of this smaller interval. This is a huge value and it means that whatever the predicted pressure value is, we are unable to correctly predict the pressure. We changed the approach to initially classify the pressure data and then predict the class for the next day.

2) Second approach: Random Forest Classifier

We thought about a barometer in order to classify the weather depending on the predicted pressure.

With this method we get the following classification :

Pressure	Weather
Smaller than 980	Storm
Between 980 and 1000	Rain
Between 1000 and 1025	Change
Between 1025 and 1050	Fair
Greater than 1050	Very Dry

We employed Random Forest Classifier and fed it the pressure classes. The model would return the class instead of the pressure. By applying cross-validation and testing for different range of previous day we found that by only using the weather class of just the previous day, the model could perform well. Indeed the model would predict the correct class 75% of the time.

D. Machine Learning Conclusion

The main impact of weather prediction is to choose between indoors and outdoors activities. A stormy, rainy or changing weather will select indoors activities and fair and very dry weather will select outdoors activities.

V. Developing Recommendation System

A. Introduction

Recommender systems or recommendation systems are a subclass of information filtering system. Recommender systems have become extremely common in recent years, and are utilized in a variety of areas: some popular applications include movies, music, news, books, research articles, search queries, social tags, and products in general.

Our Recommendation System is responsible for retrieving activities based on two criteria:

- The activity should be in the vicinity of the user;
- The proposed interesting points should be filtered according to the weather conditions.

This module relies on the weather predictor and the MongoDB that contains the geo-localized activities. We coupled the weather prediction to filter out the irrelevant activities. For example, it is not interesting to spend your Sunday afternoon at a park near to your location if it is raining.

B. Search for nearest activities

For searching the nearest stations from the user location the NoSQL database MongoDB is used. The main reason for using MongoDB system is built-in support work with geospatial indexes. The two databases collections (one for GHCN and one for ISD dataset) with station coordinates and elevation were stored in MongoDB. They are queried on user's demand for searching the nearest *n* stations and then for the temperature prediction.

C. Inside Activities

We simply output the interesting point to the user, as an inside activity is almost never influenced by weather conditions.

D. Outside Activities

For outside activities, we need to query the weather recommendation system to get:

- The predicted temperature,
- The weather predicted class (from STORMY to VERY DRY).

Activities are classified as follows:

- Winter sports to lower temperatures (below 1 degree Celsius)
- Normal activities to normal temperatures (from 10 to 20 degree)
- Extreme or water activities to highest temperatures (from 20 degrees and above)

Then we took the weather prediction into account to filter out non-suitable activities:

- If the weather will be stormy: we let no outside activities be recommended.
- If it is raining: only outside water activities can be proposed to the user.
- Else we filter nothing out.

References

[1] Abhinandan Das, Mayur Datar, Ashutosh Garg, Shyam Rajaram, "Google News Personalization: Scalable Online Collaborative Filtering"

[2] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. Mccauley, M. Franklin, S. Shenker, and I. Stoica, "Fast and interactive analytics over Hadoop data with Spark."

[3] "MLlib: Machine Learning in Apache Spark", Journal of Machine Learning Research 17 (2016) 1-7

[4] "An Analysis on Performance of Decision Tree Algorithms using Student's Qualitative Data" Published Online June 2013 in MECS

[5] Andy Liaw and Matthew Wiener, "Classification and Regression by randomForest".

[6] Anderson, Mark (18 October 2006). "Global Positioning Tech Inspires Do-It-Yourself Mapping Project". National Geographic News. Retrieved 25 February 2012.

[7] "NOAA's National Centers for Environmental Information (NCEI) " , https://www.ncdc.noaa.gov/