239 report

Jing Ma

I named my solution as “Solution of LinearRegression”

**Chapter 2**

* Algorithm(s) considered/selected (and why)

For solution of LinearRegression:

The algorithm I chose is RidgeRegressionWithSGD (L2- regularization) in Spark’s library.

There are three reasons why I use this:

* The library of Spark is limited.
* The model which is I need should be a numeric output machine learning algorithm for creating probabilities for each record.
* This one is the fastest and lowest cost one in the library they provided.
* In all Linear Regressions in Spark’s Lib, this one has the best accuracy. (Please refer part of evaluation)

How it works: in this solution, we transfer prediction to a problem as: “considering the customer will buy the insurance which in the best quotation. The best quotation means the one which has the highest probability.” In the training dataset, we put the purchase record’s probability as 1 and the quote records’ probability as 0.

Raw Dataset for training (quote records and purchase records)

**Manually add**

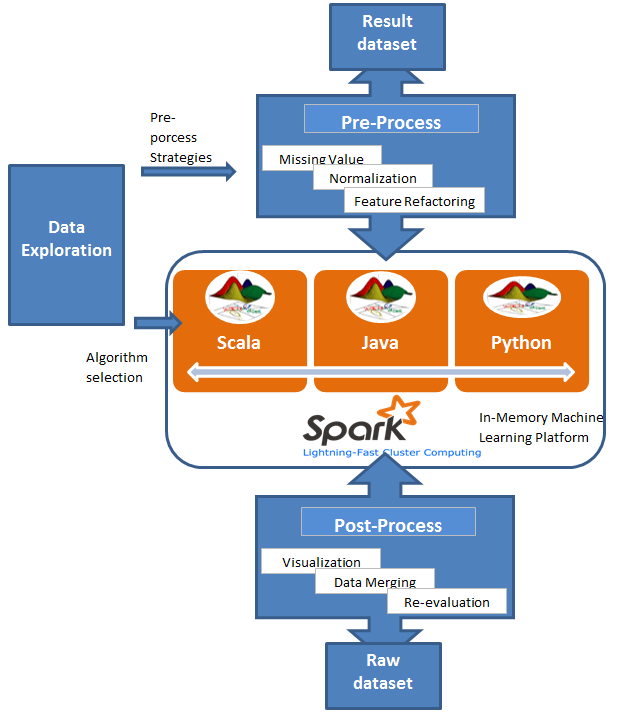
Proba-bilities

* Architecture-related decisions:

In this project, we choose Spark as our primary Machine Learning Platform, because it has several advantages:

* It is in-memory, which is very suitable for iterations.
* It can run as stand-alone, and it also can run in paralleled way on top of HDFS which is from Hadoop.
* Even it is new, it already become the top level project of Apache. Spark will have a great future in this area.
* System design/architecture/data flow

System Design:



**Ch3. Implementation Details**

* 4.Preprocessing performed (add two features:, delete one feature)

Training dataset Preprocess for solution of LinearRegression:

* Load file (raw dataset from Kaggle);
* Normalization: for all numeric features normalize them to range from 0 to 1;
* Take Boolean type to be 0 or 1, and then transfer it to be numeric;
* Put all categorized feature like states to be numeric (using linear regression, all input should be number): use a vector for describing 32 state, for example New York:

[1, 0, .... , 0]. (The first value in this vector =1 means it is New York.);

* Our prediction is for getting the largest Purchase probability: definition: this record is quote: Purchased=0; if this is purchased record: Purchased=1;
* Handle missing value: already transferred all data to be numeric, then the missing values can be filled by average value of all records.
* Feature refactoring:
  + Delete customer\_id from feature list. They are big numbers with no meaning for quotations. add features:
  + Add feature insurance combination number per customer

Assumption: more quote more engagement:

Overall dataset, how many quotes appear as this option combination of this customer has;

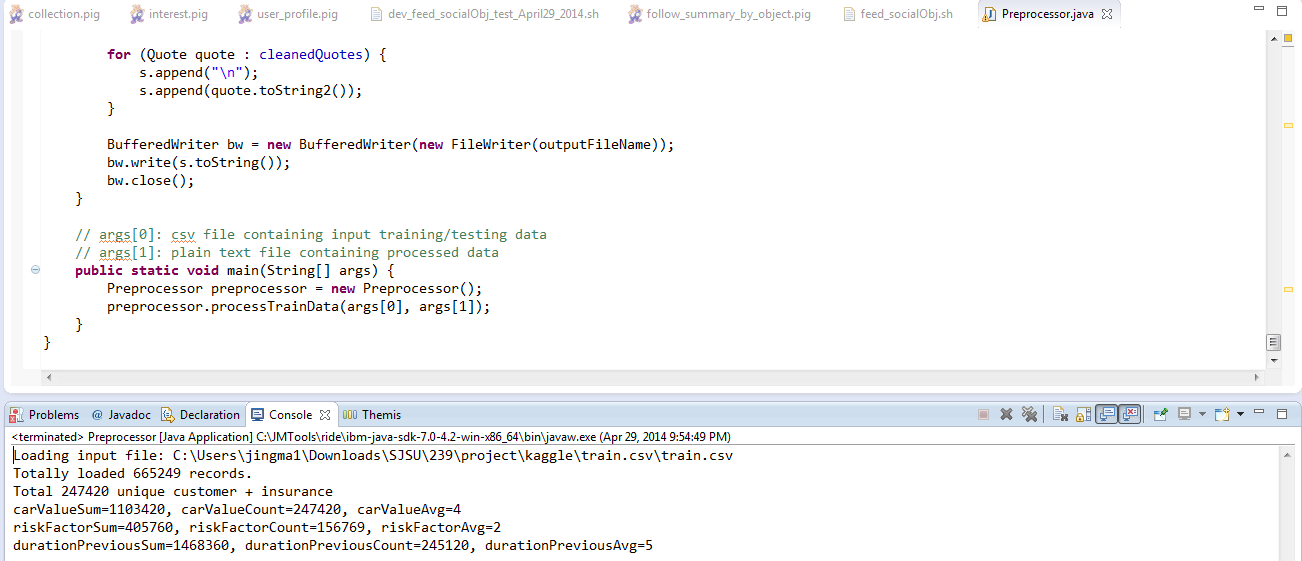
* + Add feature Feature: isLastQuote( Boolean)

Assumption and observation: We found around 70% option combination is the same with the last quote;

Test dataset: Almost every step is the same as above, except one: we delete the customer information from training dataset. Then we got the model (trained algorithms). We won’t combine customer id back to dataset in training process. However, we will combine customer id back to test dataset, since we should output a prediction for “who will buy what”.

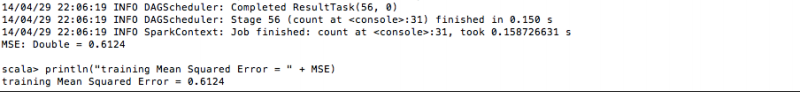
* 6. Technologies used (and why)
* I use Java for pre-processing, Python for post-processing and Scala for Machine Learning programming. We want to explore which language is more powerful and flexible for multiplying dataset. (The conclusion is Scala. It is concise, functional and platform irrelevant. )
* Spark as the primary platform for Machine Learning Platform. (Sparks’ strength please refers “Architecture-related decisions”).
* For data exploration, we use R which is good for visualization. Then we can get a quick idea of the data distribution.
* Screenshot:

Pre-process for training dataset creation:

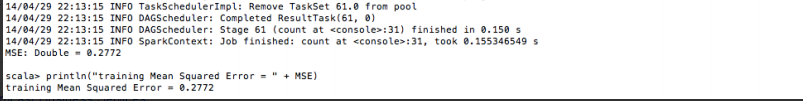


* For LinearRegression Algorithm evaluation and selection:

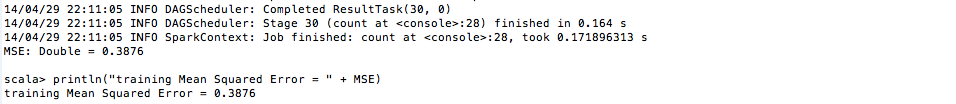
 LinearRegressionWithSGD; error Rate: 0.6124



RidgeRegressionWithSGD(L2- regularization) error Rate: 0.2772(Best one, I choose this as our machine learning algorithm.)



LassoWithSGD; error Rate: 0.3876



**Chapter 4 Experiment:**

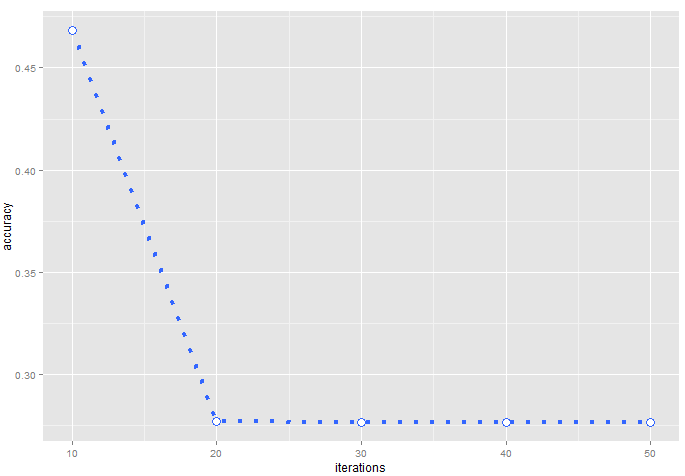
* + **Holdout set:**

I sampled 20,000 records from training dataset. Then I split it into two parts. One is for training my model, another one is for doing the evaluation.

* + **Graphs showing different parameters/algorithms evaluated in a comparative manner:**

Comparison between different Linear Regression: From this result, we choose RidgeRegressionWithSGD.

The best iteration number is 20



* + Analysis of results:

We submitted our prediction from this algorithm. The result is 53.793%. (Comparing to the top 1 result which is 54.4%, there is only less than 0.7% difference.)

From three Linear Regressions, we choose RidgeRegressionWithSGD. This one has L2-regularization which will be helpful in reducing over-fitting. We also tried iteration times as 10, 20, 30, 40, 50 and 60. We found that since it was more than 20, the accuracy stopped increasing.

That is why we choose RidgeRegressionWithSGD, with iteration number as 30. After training, we got a 53.793% accurate rate on test dataset.

Ch.5 Discussion (bullet points as applicable)

* Decisions made:

For my solution, I need to decide which kind of features should be kept and which ones should be excluding from training and test dataset. We also need to decide which algorithm and iteration number is good.

* Difficulties faced:

Building the model is the most challenging thing for us. We can understand this case as Clustering and try to find similarity between purchase records and quote. We also can think this one as a recommendation case. On this regression solution, we also can think it as probability predition.

* Project Plan / Task Distribution

Jing Ma: implement a whole solution including pre-processing, machine learning (Linear Regression: L2- regularization) and post-processing (clean and combine all results). My solution is based on Spark.

* Conclusion:

From our experiments, we have some findings:

1. Linear regression is efficient: quick and workable;

2. Algorithms choice might beat parameter optimization;

3. Good programming languages for data multiplying, from the best to worst:

Scala > R > Python >Java;

4. Spark is really fast, the only problem is the environment setting is too complex.