# Pragmatic Programming Techniques

**Monday, May 14, 2012**

## Predictive Analytics: Overview and Data visualization

I plan to start a series of blog post on predictive analytics as there is an increasing demand on applying machine learning technique to analyze large amount of raw data. This set of technique is very useful to me and I think they should be useful to other people as well. I will also going through some coding example in R. R is a statistical programming language that is very useful for performing predictive analytic tasks. In case you are not familiar with R, here is a very useful link to get some familiarity in R.

Predictive Analytics is a specialize data processing techniques focusing in solving the problem of predicting future outcome based on analyzing previous collected data. The processing cycle typically involves two phases of processing:

1. Training phase: Learn a model from training data
2. Predicting phase: Deploy the model to production and use that to predict the unknown or future outcome

The whole lifecycle of training involve the following steps.

#### Determine the input and output

At this step, we define the output (what we predict) as well as the input data (what we use) in this exercise. If the output is predicting a continuous numeric quantity, we call this exercise a "regression". If the output is predicting a discrete category, we call it a "classification". Similarly, input can also be a number, a category, or a mix of both.

Determine the ultimate output is largely a business requirement and usually well-defined (e.g. predicting the quarterly revenue). However, there are many intermediate outputs that are related (in fact they are be input) to the final output. In my experience, determining these set of intermediate outputs usually go through an back-tracking exploratory process as follows.

- Starting at the final output that we aim to predict (e.g. quarterly revenue).
- Identify all input variables that is useful to predict the output. Look into the source of getting these input data. If there is no data source corresponding to the input variable, that input variable will become the candidate of the intermediate output.
- We repeat this process to learn about these intermediate outputs. Effectively we build multiple layers of predictive analytics such that we can move from raw input data to intermediate output and eventually to the final output.

#### Feature engineering

At this step, we determine how to extract useful input information from the raw data that will be influential to the output. This is an exploratory exercise guided by domain experts. Finally a set of input feature (derived from raw input data) will be defined.

Visualizing existing data is a very useful way to come up with ideas about what features should be included. "Dataframe" in R is a common way where data samples are organized in a tabular structure. And we'll be using some dataframe that comes with the R package. Specifically the dataframe "iris" represents different types of iris and their measures in different lengths.

```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> nrow(iris)
[1] 150
```

**About Me**

Ricky Ho

I am a software architect and consultant passionate in Distributed and parallel computing, Machine learning and Data mining, SaaS and Cloud computing.

View my complete profile

**Popular Posts**

MongoDb Architecture
NOSQL has become a very heated topic for large web-scale deployment where scalability and semi-structured data driven the DB requirement tow...

Designing algorithms for Map Reduce
Since the emerging of Hadoop implementation, I have been trying to morph existing algorithms from various areas into the map/reduce model. ...

NOSQL Patterns
Over the last couple years, we see an emerging data storage mechanism for storing large scale of data. These storage solution differs quite...

Couchbase Architecture
After receiving a lot of good feedback and comment on my last blog on MongoDb , I was encouraged to do another deep dive on another popular ...

Predictive Analytics: Overview and Data visualization
I plan to start a series of blog post on predictive analytics as there is an increasing demand on applying machine learning technique to ana...

Predictive Analytics: Generalized Linear Regression
In the previous 2 posts, we have covered how to visualize input data to explore strong signals as well as how to prepare input data to a fo...

BigTable Model with Cassandra and HBase
Recently in a number of "scalability discussion meeting", I've seen the following pattern coming up repeatedly ... To make you...



**Blog Archive**

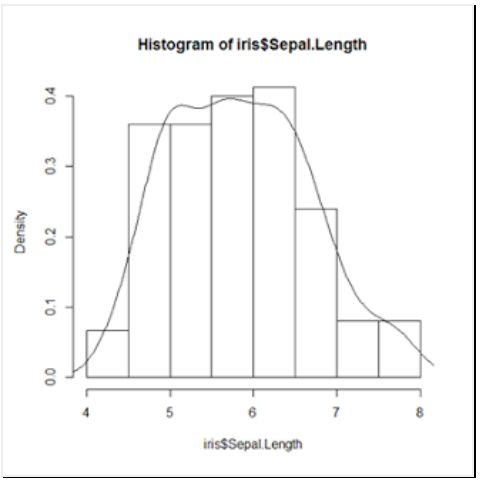
```
> table(iris$Species)

    setosa versicolor  virginica
        50         50         50
>
```

### Single Variable Frequency Plot

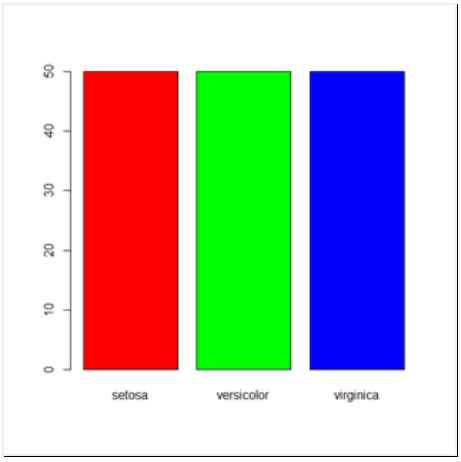
For numeric data, it is good to get some idea about their frequency distribution. Histogram and a smoother density plot will give a good idea.

```
> # Plot the histogram
> hist(iris$Sepal.Length, breaks=10, prob=T)
> # Plot the density curve
> lines(density(iris$Sepal.Length))
>
```




For category data, bar plot is a good choice.

```
> categories <- table(iris$Species)
> barplot(categories, col=c('red', 'green', 'blue'))
>
```




### Two variable Plot

Box plot can be used to visualize the distribution of a numeric value across different categories.

```
> boxplot(Sepal.Length~Species, data=iris)
>
```


- ► September (1)
- ► August (2)
- ► July (1)
- ► June (3)
- ▼ May (3)
  - Predictive Analytics: Generalized Linear Regressio...
  - Predictive Analytics: Data Preparation
  - Predictive Analytics: Overview and Data visualizat...
- ► April (3)
- ► March (1)
- ► February (1)
- ► January (1)

► 2011 (6)
► 2010 (18)
► 2009 (31)
► 2008 (22)
► 2007 (11)

**Search This Blog**

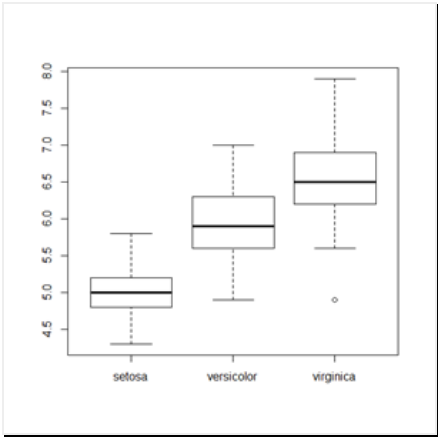
Search

**Labels**

machine learning data mining map reduce Architecture Design Cloud computing algorithm NOSQL Hadoop scalability Distributed system parallel processing big data predictive analytics Design patterns SOA performance REST ensemble method recommendation engine

**Pages**

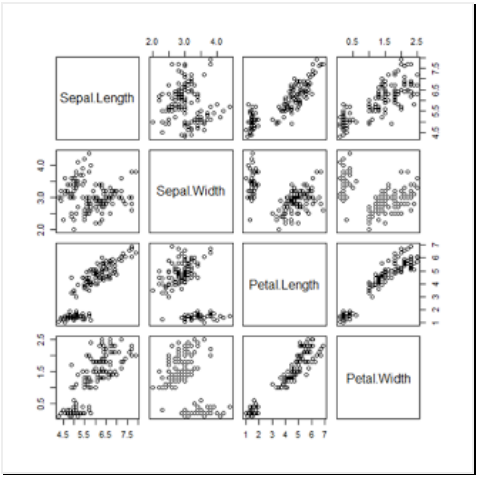
- Home




When there are multiple numeric input variables, it is useful to get an idea of their correlation to identify if there are any redundancy in our input. Certain model is sensitive to such redundancy and won't give accurate prediction if their input has high redundancy. This problem is known as the multicollinearity problem.

To get an idea of the correlation, we can use a scatter plot matrix for any two pairs of input variables. We can also use a correlation matrix for the same purpose.

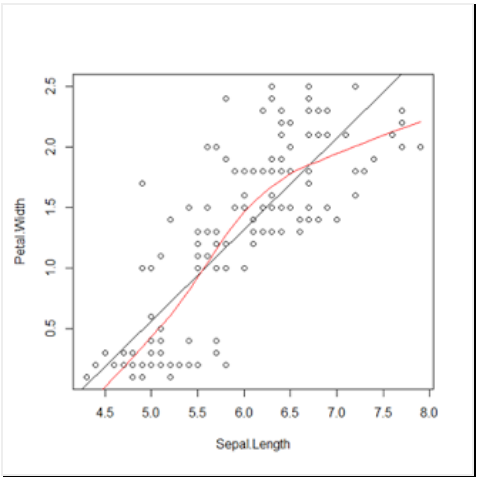
```
> # Scatter plot for all pairs
> pairs(iris[,c(1,2,3,4)])
> # Compute the correlation matrix
> cor(iris[,c(1,2,3,4)])
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    1.0000000  -0.1170695    0.8716902   0.8179410
Sepal.Width    -0.1170695   1.0000000   -0.4284401  -0.3661259
Petal.Length    0.8716902  -0.4284401    1.0000000   0.9628654
Petal.Width     0.8179410  -0.3661259    0.9628654   1.0000000
>
```




From the observation, we can see Petal Length is highly correlated to Petal Width, and also there are some correlation between Sepal Length and Petal Width as well.

We can further drill down into analyzing the relationship between two numeric value by fitting a regression line or a regression curve (by performing local neighbor linear regression).

```
> # First plot the 2 variables
> plot(Petal.Width~Sepal.Length, data=iris)
> # Learn the regression model
> model <- lm(Petal.Width~Sepal.Length, data=iris)
> # Plot the regression line
> abline(model)
> # Now learn the local linear model
> model2 <- lowess(iris$Petal.Width~iris$Sepal.Length)
> lines(model2, col="red")
>
```




### OLAP processing

If the data is in multi-dimensional form (has multiple categorical attributes), OLAP type manipulation can give a very good summary.

In this section, lets use a different data source CO2, which provides the carbon dioxide uptake in grass plants.

```
> head(CO2)
  Plant   Type  Treatment conc uptake
1   Qn1 Quebec nonchilled   95   16.0
2   Qn1 Quebec nonchilled  175   30.4
3   Qn1 Quebec nonchilled  250   34.8
4   Qn1 Quebec nonchilled  350   37.2
5   Qn1 Quebec nonchilled  500   35.3
6   Qn1 Quebec nonchilled  675   39.2
>
```

Lets look at the count summarized in different dimensions

```
> cube <- xtabs(~Plant+Type+Treatment, data=CO2)
> cube
, , Treatment = nonchilled

     Type
Plant Quebec Mississippi
  Qn1      7           0
  Qn2      7           0
  Qn3      7           0
  Qc1      0           0
  Qc3      0           0
  Qc2      0           0
  Mn3      0           7
  Mn2      0           7
  Mn1      0           7
  Mc2      0           0
  Mc3      0           0
  Mc1      0           0

, , Treatment = chilled

     Type
Plant Quebec Mississippi
  Qn1      0           0
  Qn2      0           0
  Qn3      0           0
  Qc1      7           0
  Qc3      7           0
  Qc2      7           0
  Mn3      0           0
  Mn2      0           0
  Mn1      0           0
  Mc2      0           7
  Mc3      0           7
  Mc1      0           7

> apply(cube, c("Plant"), sum)
Qn1 Qn2 Qn3 Qc1 Qc3 Qc2 Mn3 Mn2 Mn1 Mc2 Mc3 Mc1
  7   7   7   7   7   7   7   7   7   7   7   7
> apply(cube, c("Type"), sum)
     Quebec Mississippi
         42          42
> apply(cube, c("Plant", "Type"), mean)
```

```
     Type
Plant Quebec Mississippi
  Qn1    3.5          0.0
  Qn2    3.5          0.0
  Qn3    3.5          0.0
  Qc1    3.5          0.0
  Qc3    3.5          0.0
  Qc2    3.5          0.0
  Mn3    0.0          3.5
  Mn2    0.0          3.5
  Mn1    0.0          3.5
  Mc2    0.0          3.5
  Mc3    0.0          3.5
  Mc1    0.0          3.5
>
```

#### Prepare training data

At this step, the purpose is to transform the raw data in a form that can fit into the machine learning model. Some common steps including ...

- Data sampling
- Data validation and handle missing data
- Normalize numeric value into a uniform range
- Compute aggregated value (a special case is to compute frequency counts)
- Expand categorical field to binary fields
- Discretize numeric value into categories
- Create derived fields from existing fields
- Reduce dimensionality
- Power and Log transformation

Data preparation is the bulk of effort for most predictive analytic tasks and usually consumes 60 to 80% of the time. This is an important area that deserve a dedicated post. I'll cover data preparation in my next blog post.

#### Train a model

At this step, we pick a machine learning model based on the characteristics of the input and output features. Then we feed the training data into the learning algorithm which produce a set of parameters to explain the occurrence of training data.

There are many machine learning models that we can choose from, some common ones including ...

- Linear regression
- Logistic regression
- Regression with regularization
- Neural Network
- Support Vector Machine
- Naïve Bayes
- Nearest Neighbors
- Decision Tree
- Ensemble Methods (Random Forest, Gradient Boosted Tree)

Each of the above models has its own characteristics and fit better in certain types of problems. I'll cover each of them more detail in future posts.

#### Validate the model

After we train the model, how do we validate the model do a good job in prediction. Typically, we withhold a subset of training data for testing purpose. Through the testing, we quantitatively measure the performance of our model prediction ability. I'll cover this topic in more detail in future post.

#### Deploy the model

When we are satisfied with the model, we'll deploy the model in production environment and use that to predict the real-life events. We should also closely monitor if the real-life accuracy aligns with our testing result.

Usually, the quality of the model prediction degrades over time due to the stationary assumption (the data pattern in future is similar to the data pattern at training time) not longer holds as time passes. On the other hand, we may acquire additional input signal that can improve the prediction accuracy. Therefore, model should have an expiration time and need to be re-train after that.

Hopefully, this post gave a high level overview on the predictive analytics process. I'll drill down to each items in more detail in my future posts.

Posted by Ricky Ho at 11:38 PM

Labels: data mining, data visualization, machine learning, predictive analytics

## 5 comments:

**Jos Verwoerd said...**

Hi Ricky, thanks for this post. It's a great idea to start a series on predictive analytics. At BigML's we fully recognize the increasing demand for machine learning technology to analyze large dta. Recently we launched our machine learning as a service. It'll enable the user to simply do most of the tasks you've described here by a simple mouse click or API call. It is great that through posts like yours and services like BigMLs many businesses are able to benefit from predictive analytics.

May 15, 2012 at 12:19 AM

**nimrodpriell said...**

This is a very good introduction into exploratory techniques. I'm looking forward to your post about the data preparation. While many people talk about the machine learning aspects (different algorithms, etc.), like you say a lot of work goes into cleaning and preparing the data.

In terms of exploratory techniques I would also stress out the importance of finding outliers using maybe QQ plots or other techniques and using different summarizing measures (looking at medians and MADM, not just averages and standard deviation).

May 15, 2012 at 5:37 AM

**CHIT said...**

clear and nice article with code examples.

May 16, 2012 at 12:48 PM

**Dylan said...**

Thanks Ricky, you've put together a great series of posts here. Nice to see attention to data preparation/feature engineering and also evaluation after the model has been trained.

A terrific resource particularly for people getting started with Kaggle competitions!

June 15, 2012 at 5:43 PM

**Prabhanjana Guttal said...**

Excellent article, very useful.

August 27, 2013 at 4:45 AM

Post a Comment

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)