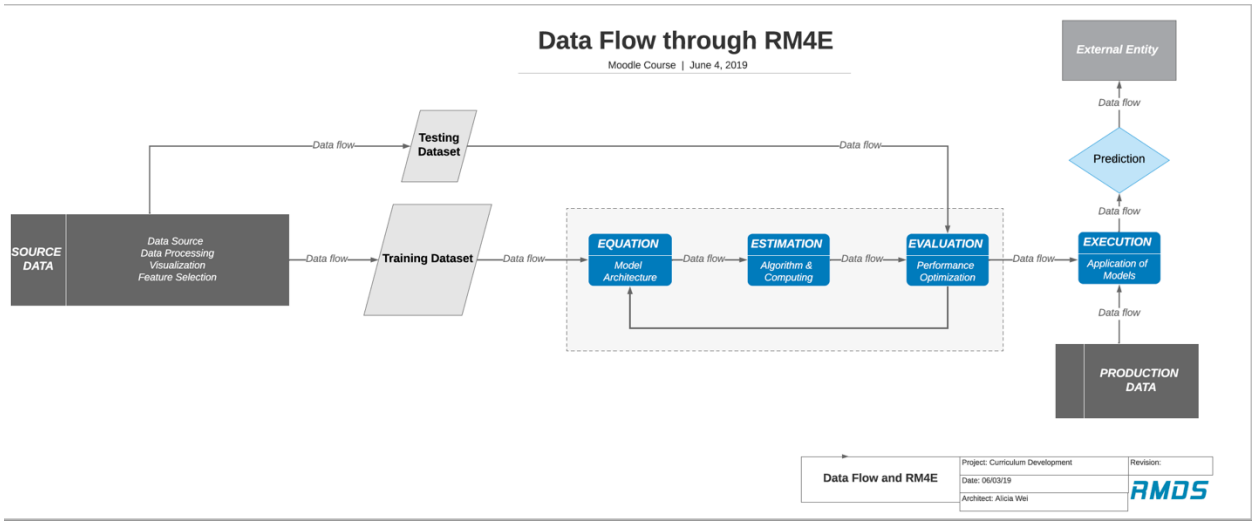
The Workflow Development GuideBook

The RMDS Lab

www.grmds.org

www.rmdslab.com



I. Introduction

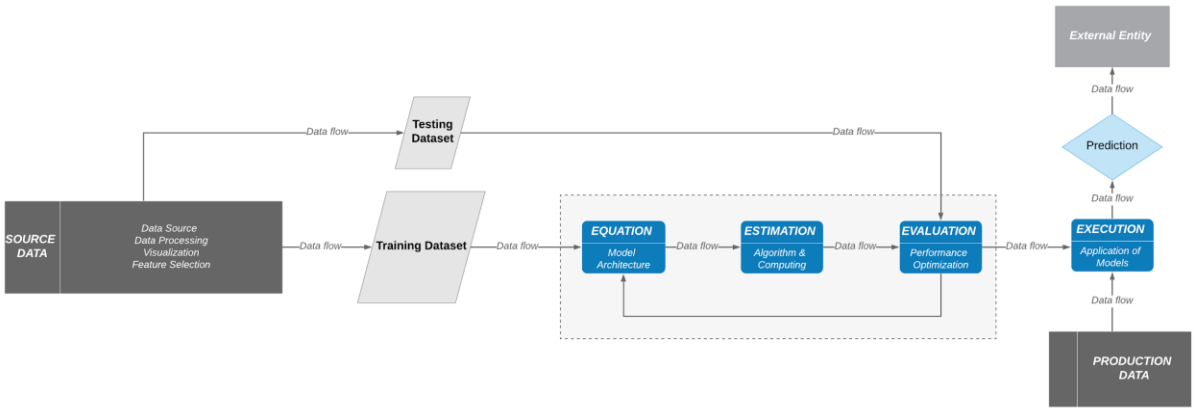
What is a data science workflow?

The data science workflow is a completed procedure of data analyzation. A workflow usually follows these steps: data collection, data cleaning, data processing, data visualization and analysis, modeling, model application. Modeling can also be divided into several parts covering supervised and unsupervised machine learning, regression and classification, etc. RMDS data science workflow specifically follows an ecosystem approach with the RM4Es approach: equation, estimation, evaluation, and execution.

While the detailed process depends on specific occasions, there are indeed some theoretical ideas and guidance we should follow in general. This guidebook aims to be a useful tool for people who have data analysis experience and to set up a few standards for each step when constructing an ideal workflow.

This guidebook contains five parts, part 1 is the introduction of RMDS workflow and the purpose of this guidebook, part 2 illustrates the RM4Es under our ecosystem approach, part 3 states some general standards for workflow, while part 4 explains the standards in practical examples, and part 5 is about the review and approval process for a workflow to be published in our RMDS community.

II. 4E



a. Equation - equations represent the models and frameworks for our research. They serve as a link between data and research ideas or designs.

b. Estimation - estimation is the link between equations (models) and the data used for our research. They are the algorithms used to compute the parameters of our models.
c. Evaluation - errors are used to evaluate the fit between models and data. These metrics evaluate the performance of our estimation methods and the produced models.
d. Execution or explanation – the explanation is the link between equations (models) and our intended research purposes. How we explain our results depends on our research purposes and also on the subject we are studying. Execution is the step that takes the created model for production, which is then utilized for decision making.

III. Standards for Workflow

This section will state the steps a RM4E workflow should follow. The following will first demonstrate the workflow building process. Our expert committee will then judge your workflow based on the following RM4Es standard.

We initially built our RM4E idea based on the CRISP-DM model. However, we then refined it with our 4E and ResearchMap frameworks. CRISP-DM is abbreviation for Cross Industry Standard Process for Data Mining. The 6 steps CRISP-DM follows are: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment (Quote from **https://www.datasciencecentral.com/profiles/blogs/crisp-dm-a-standard-methodology-to-ensure-a-good-outcome**).

In the following demonstrations for building the standard workflow, we will use a data analytics case study on Apple's App Store Applications to offer a real-life illustration. The example here will be shown using Python. Extra examples in R will be displayed in the later sections.

Define Questions

- What business question do we want to solve?
- What are the specific tasks?
- What would be success metrics?

One of the widest uses for workflow is to make predictions on. The applications are widespread and can be used for a variety of purposes. For example, the applications could be to predict a movie's box office profit or the score of sports games. Different types of predictions can be driven from one data set based on what it is you're trying to predict. Therefore, before stepping into analytics of the dataset, be sure to clarify the goal for the modelling or analysis. For example, what business value you want to drive from the data, or what value you want to predict for your project. It can be predicting a specific numerical result (regression),or forecasting the success of one event

(classification). Always have your objective in mind throughout the process of building workflow on your data.

Case Study

Our goal is to predict the popularity for each app based on its features offered in App Store.

Data Sources

Having the goal settled, searching for resources is the next key step. There are many general data storage websites we can refer to, such as Kaggle, Google Public Data Explorer, Open Data for each city (in the US), and Census Bureau. Besides these websites, we can also scrape data from websites based on our needs, using API, json, or other web scraping tools. For example, to get information from Google Maps, we need to apply for its API on Google Cloud Platform. To scrape from websites in certain disciplines such as the IMDB website, we need to look at its web structure through 'inspect', scrape relevant data and build up the database on our own.

The dataset can also be stored in different ways. The common way to store it is in Microsoft Excel. It's usually in a '.xlsx' or '.csv' format. This type of data is also rather easy to handle, using 'read.csv' in R or 'pd.read_csv' in Python from Pandas Package. One drawback for Excel is that it takes longer to process data when you a large amount. Some other database storages could be SQL, BigQuery, etc. It's easy to connect to these databases both on the local machines or cloud platform, such as Google Cloud Platform and AWS. Both SQL and BigQuery perform under SQL command.

Case Study

In our case, we find that each app in App Store has a specific json file that contains description including name, category, language of the app, etc. By accessing URLs with different keywords, the website will return relevant json files. To make the files return automatically instead of running the code manually, we also applied Selenium to the system. This makes the code keep running by itself and the json files will be saved into a local file. Here is an example.

```
import selenium
from selenium import webdriver
options = webdriver.ChromeOptions()
options.add_argument('--headless')

```

```
import time
term = 'abc'
link = 'https://itunes.apple.com/search?country=us&entity=software&term='+term+'&limit=100'
browser = webdriver.Chrome(executable_path='chromedriver')
browser.get(link)
time.sleep(2)
browser.page_source
```

In these json files, as mentioned before, each node represents one feature, such as ‘game_center’, ’genre’, ‘price’, ‘language’, etc. Therefore, we extract data from the json file in a for loop, and rearrange them into dataframe. We get about 35k rows of data in total.

```
import pandas as pd
import json
def file_to_df(file):
    t = open(file, "r")
    text=t.read()
    try:
        jsontext=json.loads(text)
        name,ID,release,game,device,current_release,size,cnt_shot,rate_ver,rate_cnt_ver,genre,primary,currency,pri
        if jsontext["resultCount"] != 0:
            for i in range(len(jsontext['results'])):
                nm=jsontext['results'][i]['trackCensoredName']
                name.append(nm)
                id_=jsontext['results'][i]['trackId']
                ID.append(id_)

                if 'sellerName' in jsontext['results'][i]:
                    sel=jsontext['results'][i]['sellerName']
                    seller.append(sel)
                else:
                    seller.append(None)
```

And the data frame looks like:

Out[7]:

	GameCenter	ID	cur_release	currency	description	genre	language	name	num_device	num_display
0	False	1211999956	2017-03-26T20:22:24Z	USD	Kalender GKY disediakan oleh Sinode GKY, didal...	[Business]	[EN]	Kalender GKY	58	3
1	False	1390457257	2018-12-04T03:56:55Z	USD	GKY Puri Mobile App provides News, Magazines, ...	[Books, Lifestyle]	[EN]	GKY Puri	58	8
2	False	492008270	2017-01-19T02:58:38Z	USD	Read through the Bible in 3 years using GEMA. ...	[Books, Reference]	[ZH, EN, ID]	GEMA	58	5
3	False	998550591	2016-02-26T22:53:56Z	USD	WLR UK is a 4x4 off-road racing game with real...	[Games, Business, Racing, Simulation]	[EN]	WLR UK	63	4
4	False	390426794	2018-06-15T17:21:54Z	USD	WLR is Waterford's favourite radio station. We...	[Music, News]	[EN]	WLR FM	58	3
5	False	1078012508	2018-07-02T20:53:15Z	USD	This app provides you with a free, simple and ...	[Music, Entertainment]	[AR, HY, CA, CS, DA, NL, EN, FI, FR, DE, EL, H...	Radios Ireland FM Irish Radio	57	5

Cleaning Data

Fill/Omit missing values:

Having the raw data in hand, it's normal that we see missing values or outliers in a dataset. However, always remember, do not clean them up directly! Look into the data first, you may actually find the 'NA' or outliers valuable to you. (This depends on your objectives and data in specific cases as well.)

To detect missing values, you can use '.isna()' or '.isnull()' in Python Pandas Package, and it will return boolean values with 'TRUE' or 'FALSE'. To drop all 'NA' values in the dataframe, use 'pd.dropna()'. You can also add a condition to the drop null value statement: 'df.dropna(axis='columns')' means to drop columns with missing values, etc. Instead of dropping off the missing values, you may also want to fill them in. 'df.fillna(0)' means to fill all null values with 0.

Fix data inconsistency

If there is any inconsistency in the data, for instance, a data in different units of measurement while some are in centimeters and some others in inches, be sure to check and fix the inconsistency to avoid error and bias.

Data Integration

Besides dealing with the missing value and inconsistency issue, you may find the data you have right now does not contain enough information you need to reach your goal. In this case, you may need to add extra information to your data by further looking at other resources. For example, suppose you are working on a movie's Box Office prediction based on IMDB data, and you figure having rates from different film websites may be helpful in accuracy, you would need to scratch more information from sites like Rotten Tomatoes.

Case Study In the case study, we want to use as much observations as we can get from App Store, so we fulfill all the null values with 0 as the first step.

```
New_df.fillna(value=0)
New_df.reset_index(drop=True)
```

Data Transformation

Congratulations! Now you have your basic dataset built up, but we may still need to do some transformation to make the data a better fit.

Feature Selection

Data scientists are familiar with ‘overfitting’, meaning that the model describes the original data (usually training set) too well that it’s not doing a good job in making predictions (on test and validation set). Feature selection is one effective way to avoid overfitting.

There are also many algorithms for feature selection: subset selection, test-validation, k-fold, etc. While these algorithms can both help you with feature selection, there are pros and cons for each of them. Subset selection is more computationally complex, while k-fold seems easier to be used with big data. You can choose any algorithm based on your case.

Feature Extraction

Other than selecting features from the existing ones, you can also compile and extract your own features. PCA, short for principal component analysis, is one approach. The idea of PCA is to use less features to explain more in the dataset. You can generate PVE, percentage of value explained, with different number of principal components to examine how well the combinations of new components work in explaining the original data. You would want to choose a compromised one eventually, one with relatively high PVE and relatively low number of PC. Furthermore, you can build regression on the new component variables, named PLS (partial least squares regression) and PCR (principle component regression).

Normalization

Once the dataset is built up, you need to decide whether or not the data needs some transformation. The transformation includes but does not limit to data aggregation, generalization, normalization, etc. In many dataset, we apply feature engineering as transformation. Feature engineering here refers to transforming or even adding more variables based on one’s domain knowledge. Some examples can be turning variable x into its absolute value |x| or squares x^2. Building new interaction terms like x1*x2 is also a common method.

Other than feature engineering, you may also need to pay attention to normalization of the data. It may not be necessary for all the datasets to be standardized, as

standardization very much depends on the specific situation, but there are certain machine learning algorithms that require this process.

K-NN is one of these algorithms. By K-nearest Neighbor Theory, we make predictions on data based on their similarity between other observations, and ‘similarity’ here is determined by the Euclidean distance between two neighbors. When calculating the distance, we want to weight the importance of each features equally. There are indeed cases that you want to weigh one feature more important than the other, then you can add the coefficient in front of the equation. However, in any case, it’s a must to normalize data before applying the K-NN algorithm. Otherwise, data in larger absolute value will be automatically weighted more in the algorithm.

Case Study

Since we are getting data for apps randomly, we do get data from apps outside of US or from non-English speaking languages. To avoid bias, we decided to focus on data within the US App market. Therefore, we did text analytics from the title for each app and got rid of the non-english apps. Moreover, as we figure keywords in titles may also affect one app’s popularity, we did a keyword count and rank them by frequency. Then we take keywords with Top 30 frequencies as a dummy variable in our dataset. For example, ‘notes’ is a Top30 keyword, and we add a feature named ‘notes’. For each observation, if ‘notes’ appear in the title, then we mark a 1 under this feature. If not, we mark it as 0.

We also apply interaction terms between the dummy variables and all other features, like ‘study X notes’. We ended up with 90 features in total.

Data Analysis

Hooray, you finally get the data processing done when you get to this part! Welcome to the world of analysis.

For data scientists to understand a set of data, besides their domain knowledge, getting familiar with the data quickly will also make the modeling easier. Data visualization can be one way to help in analysis. You can see the general distribution of data through collective bar graphs, determine outliers on linear graphs, or see the correlation between two variables, etc. This is not much of a technical process, but this is how you make the data tell stories.

Case Study

Here we aggregate the amount of apps under each category, and see the correlation between each feature. We found that apps under the genre ‘game’ plays tend to be more popular than the others.

Statistical Modeling/Prediction

I believe this is the part you have been waiting for the whole time, it’s modeling time!

When talking about machine learning, we all know that it’s been divided into two categories: supervised and unsupervised learning. Supervised learning refers to observations with both input and output values (X and Y), while unsupervised learning typically organize data into clusters by their similarity.

There are packages for modeling in both R and Python. In Python, install ‘sklearn’ package. Notice that this is not a built-in package, so you need to install it by using ‘pip install’. The first step for all modeling process is to split the original data into train, test and validation test. It’s usually divided into the portion of 5:2.5:2.5.

When building up models for a dataset, it’s always good to have a baseline model built up first before getting deep into the more complex machine learning models. Usually, we use linear or logistic regression as our very first step since these are the most straightforward ones. Building upon the baseline, we improve the models step by step. Some of the criteria we use to judge the fitness of one model would be Adjusted R Square, MSE (mean square error), deviance, accuracy, etc.

Case Study

In our example, we set linear regression as our base line, and we also applied logistic regression, decision tree, random forest, K-NN, neural network. We take accuracy rate as our standard of measure in this case. The following chart is showing the result for all the models mentioned:

	Train (accuracy)	Valid (accuracy)	Test (accuracy)	Train (precision)	Valid (precision)	Test (precision)
Linear regression	86.87%	86.77%	86.75%	34.36%	33.90%	33.77%
Logistic regression	88.20%	88.20%	88.09%	40.99%	41.04%	40.45%
ANN (all features)	90.11%	90.10%	90.09%	36.52%	35.71%	38.75%
ANN (no “music”)	87.83%	87.44%	87.69%	44.58%	43.49%	44.05%
ANN (final - no “music” and “t30editor”)	90.23%	90.25%	90.04%	58.14%	54.69%	51.79%
k-NN	/	90.58%	90.58%	/	60.66%	60.87%
Decision Tree	90.62%	77.42%	76.12%	85.06%	67.81%	68.46%
Random Forest	98.35%	84.81%	84.02%	95.64%	60.87%	58.33%

Interpretation of results

Be careful about the wording in the interpretation or stating results in regards of your primary objective or business question. Pay attention to the interpretation of some tricky terms like ‘p-value’, ‘Adjusted R Square’, ‘confidence interval’, etc. It’s great to have the professional statistical interpretation included, and it may also be necessary to have interpretation in plain English for your non-statistical audience in some cases.

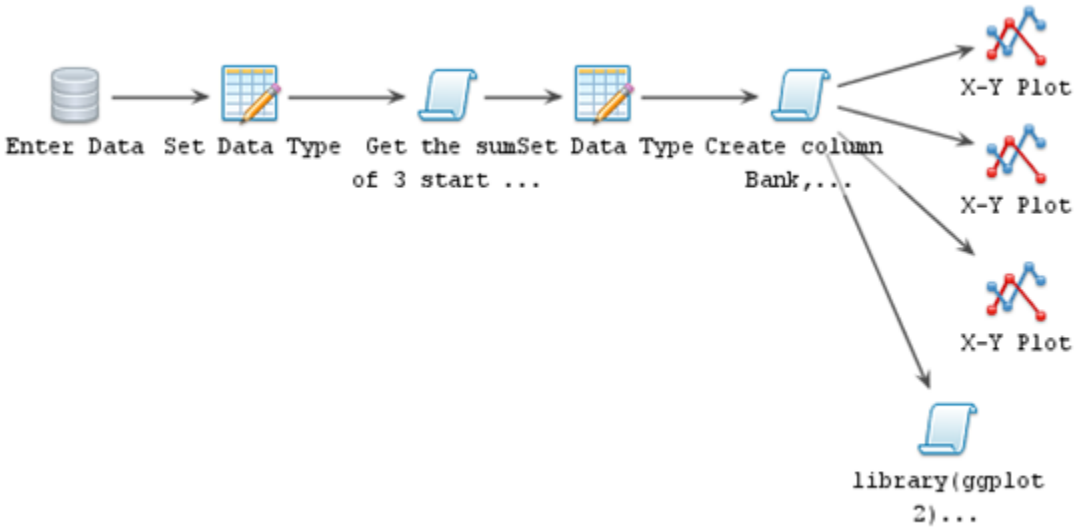
Synthesis and write up

Once the workflow is done, you will need to write a short report or a ‘readme’ file to concisely describe and summarize your workflow, as our purpose is to build up reproducible and reusable workflows for the community. In this way, people with similar dataset in hand can refer to your workflow to either customize their own workflows, or even reuse yours directly. This will make the analytics in the future more efficient. As open sources, the ‘readme’ texts also allow data scientists in our RMDS community to better understand your project and improve in the future if necessary.

IV. Tools With Examples

Example with R using CDI (Chinese Data Institute) data

Here is what workflow should look like in visualization:



#This data set contains the startup financial institutes in different Chinese provinces from 1949 to 2004

1. We first need to load the data

#read the excel file (By changing the name of Province to get the data from different Province, I use Beijing as an example)

```
Beijing<read_excel("E:/RMDS/CHINA_DATA_LAB/Final_Reports/Alteryx_input/2004 Finance_by_Province_1.xlsx",sheet="Beijing")
```

2. After loading the data, we check the data type. (Make sure the variable type is numeric for further calculation.)

```
sapply(Beijing,class)
```

3. After we changed our data type to numeric, we need to have a new column to calculate the total number of startup institutions.

```
Beijing$Beijing_Total<- Beijing$`Banking startup total (I_6810, I_6820,I_6890)` +
Beijing$`Securities startup total(I_6910, I_6920, I_6930, I_6940)`+Beijing$`Insurance startup
total (I_7010, I_7020, I_7030)`
```

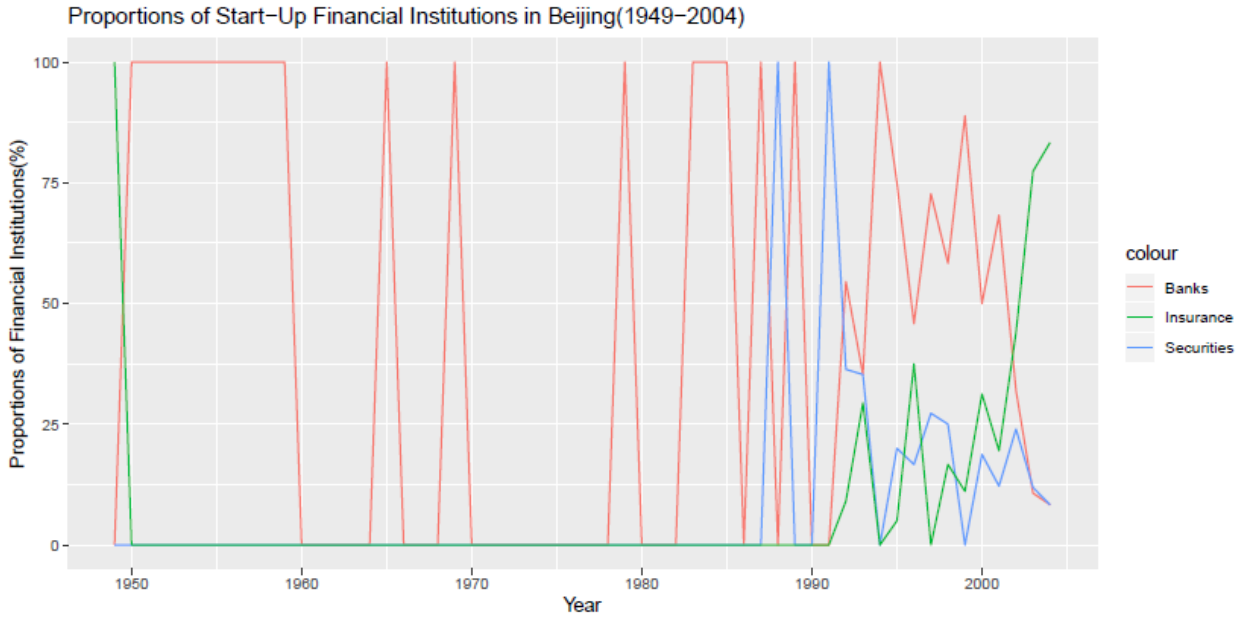
4. After we have the total column, we need to calculate the proportion of each institution have built in each year (Here is one example for banks)

```
Beijing$Beijing_Banks<-ifelse(Beijing$Beijing_Total== 0,0,Beijing$`Banking_startup_total
(I_6810,I_6820, I_6890)`*100/Beijing$Beijing_Total)
```

5. Next, we create a plot of the trend for different startup institutions from 1949 to 2004 (Use ggplot package)

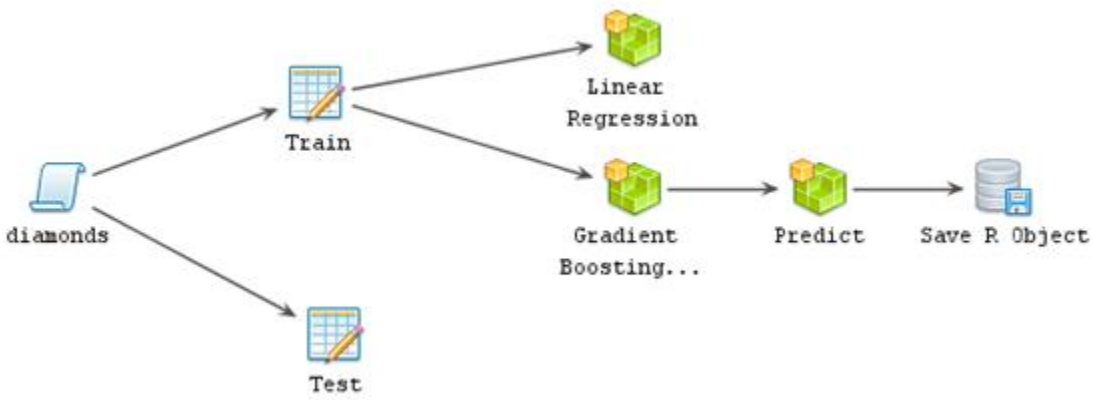
6. Finally, we export our plot to a pdf file. (Use ggsave function)

Here is output:



Example with R using data relating to diamonds:
Diamond dataset containing the prices and other attributes of almost 54,000 diamonds. It is a dataset inside of R package –ggplot2

Here is a simple workflow of diamonds:



1. Load the data, and check the missing value

```
#Load and look at the data diamonds
data("diamonds")
head(diamonds)

#Check missing value
sum( is.na( diamonds ) ) > 0 #is.na<=0 which means there are no missing value
```

2. Set the random seed for further model

```
set.seed()
```

3. Split the diamond into train and test data set

```
Sample.split()
#There many ways to split data, make sure how much percentage in each dataset
```

4. Build the first model (**Equation**)

```
#In this workflow, I use linear regression model for my first model
#Before I build the model, I use ggparis() to check the correlation #between different variables
to finalize which variable to use.
Example:
lin_regress<-lm(price~.-x-y-z,data = train)
```

5. Check the Adjusted R^2 and coefficient to get an idea about the model and relationship

```
summary(lin_regress)
```

6. Implement our test data to the linear regression model (**Estimation**)

predict(lin_regress,test)

7. After getting the prediction results, we can compare the actual result and calculate the Root Mean Square Error for my model (**Evaluation**)

#There are also other ways to measure the performance of a model, e.g. https://indatalabs.com/blog/predictive-models-performance-evaluation-important
sqrt(mean(Final_Data$error^2))

8. Next, we use Gradient Boosting Machines to build another model to check if there are any improvements (**Equation**)

I use package #library(gbm)
There are many other packages to use like xgboost, h2o, caret, etc.
I change the interaction.depth to 3 to have better perform.

9. Same as before, we need to implement our test dataset to GBM model

#Estimation our prediction (**Estimation**)

10. Calculate the Root Mean Square Error for this model

Compare the error with the previous model and find the better one

11. After the comparison, we find out the Gradient Boosting Machine Model has a smaller RMSE. We decided to use this one to predict the price (**Execution**)

V. Review & Approval Process

When your workflow is done, our expert committee will review your workflow following the standards as this guidebook suggests. Once approved, it will be uploaded to our community website as an open source, in order to let other experts discuss and improve upon.

Thanks for your participation and we warmly welcome you to join the RMDS community!