For office use only	Team Control Number	For office use only
T1 _____	8700	F1 _____
T2 _____		F2 _____
T3 _____	Problem Chosen	F3 _____
T4 _____	A	F4 _____

2018
HiMCM
Summary Sheet

From Rollercoaster's Indicators to "RollerOutstander"

Roller Coaster rankings are typically linked to solely subjective factors, such as excitement and adventure. Thus, building an objective ranking for the roller coaster requires in-depth evaluation of determinants of a roller coaster's quality. To solve this problem, we build a comprehensive evaluation model and apply entropy weight method to calculate the ranking of the roller coasters. We also develop concept and design of an app targeting potential roller coaster players, "RollerOutstander". We propose its user-friendly features including interactive filtering and recommendation system, and select algorithms for these features.

In problem one, we first analyze and preprocess the data, which includes dimensionless treatment, handling the outliers, and others. We also use a neural network to fill the missing values of the data. The testified error of the filled data value is 3%. Next, we analyze the data from the given data table and the Internet. We screen the indicators based on the principle of objectiveness and quantifiability and use grey correlation analysis to screen the factors with high correlation. Though the screening of indicators might limit the range of indicators we use, it achieves our goal of objectiveness. Last but not least, we use entropy weight method to assign objective weights to the indicators and obtain the ultimate comprehensive model for roller coasters.

In problem 2, we calculate the top 10 ranking of roller coasters (Fig8.) using our comprehensive evaluation model in problem 1. We test the stability of our model using sensitivity analysis. Compared with the online rankings, our own roller coaster ranking system would give a more comprehensive review of the factors, including the basic technology of the roller coasters and excluding impact of the tourism industry and economy.

In problem 3, we design an app for the potential riders. At the model level, we establish a more comprehensive indicator evaluation system and use fuzzy evaluation method to quantify the descriptive indicators. At the algorithm level, we use the collaborative filtering algorithm, and tag-based recommendation algorithm to generate accurate and customized recommendation for our users. On the user experience level, we design the GUI of the app, which helps the users take initiative action of filtering, or recommend roller coasters to the users based on the system. We also designed other user-friendly features including ticket booking information querying system.

Keywords: Neural network, Entropy weight method, Fuzzy evaluation, Grey relational analysis, Collaborative filtering, Tag-based recommendation

Team#8700

RollerOutstander, your personal recommender of the roller coasters

Nov.19th, 2018

Also, the most objective ranking is available at the bottom.

Heartbeat, screaming, trembling, free of gravity… Roller coasters are all about the excitement. Like many items, people have different taste of it. It used to be a mystery to find the excellent roller coaster that meets your expectation, now RollerOutstander makes it all within a download of app.

The most objective ranking of roller coasters has been developed as well. We look through all the factors that increase the experience of excitement and provides an exceptional reference to all the people who love the stimulation. Check the list at the bottom if you love roller coasters to find if you have taken any one of them:

Ranking	Name
1	Kingda Ka
2	Top Thrill Dragster
3	Superman: Escaper from Krypton
4	Steel Dragon 2000
5	Red Force
6	Tower of Terror II
7	Millennium Force
8	Formula Rossa
9	Fury 325
10	Leviathan

Covering more than 300 most advanced roller coasters and a great many common ones all over the world, RollerOutstander automatically analyzes your personal profile you develop by yourself and recommend the best roller coasters that suite you well. Whichever type you may enjoy, whether you are into speed, height or simply the scenery, there will be a type for you on RollerOutstander. The app will know your interest better than anyone else while you may not even notice. A customized recommendation list is created as soon as you register.

The app is much more than that. We also build a community for those who are thrilled by roller coasters. You will be able to comment on the roller coasters you take, share experiences and look for fun facts. The pricing and special account are also available. Basically, all the things you may want to know are on the app.

It's definitely the app you would need to look for thrilling experience.

Unlock your
Outstanding
Rollercoaster Experience.

Anastarica National #1
Roller coaster advising app

G#8700

RollerOutstander

Recommendation
Exclusively
For You

Contents

1 Introduction

1.1 Problem Background

The roller coaster is an exciting recreational facility. The thrill of achieving such a high speed fascinates a lot of people. The wonderful experience includes not only the adventure of an unexpected journey but also the acceleration and gravitational force intertwined.

With the development of technology and discovery of new materials, the techniques of constructing the roller coasters' loopbacks, slopes and reversions have been more advanced. The types of roller coasters have been more abundant, which bring the rollers even more attractive and adventurous.

1.2 Literary Review

The previous roller coaster rankings are basically in two types: entirely quantitative ones and the ones based on the subjective judgments.

An example of the quantitative one is extracted from the *Roller Coaster Database,* which was created in 1996 by Duane Marden. It outlines the rankings based on different factors: speed, height, length, drop, inversions and angle. ("Record Holder")

The subjective one such as the one released by ranker describes the top roller coasters as "the tallest roller coasters, the fastest roller coasters, and the most thrilling roller coasters". (The Best Roller Coasters in the World)

1.3 Our Work

Our aim is to screen out reasonable indicator and establish a comprehensive evaluation model, which can practically judge the roller coasters in an objective way. In particular, we will include an improved evaluation model to support the design of an app with an algorithm, which can satisfy the demand of customers, and provide a subjective ranking of roller coasters.

Our work includes the following:

- Analyze and process various types of data.
- Establish a complete indicator evaluation system.
- Establish an objective comprehensive evaluation model and test the stability of the model.
- Analyze the differences and causes between our rankings and online website rankings.
- Improve our model and design app algorithm.
- Design the app interface and give a news release.

2 Defining the Variables

Symbol	Description
u	User ID
i, j	Item ID

U	Set of all users
I	Set of all items
$R(u)$	Set of items recommended to user u
$T(u)$	Set of items user u interacted with in the test
w_{ij}	Similarity index of item i and j

This is just a few of the variable descriptions, and others will be explained in the article.

3 General Assumption

To simplify the problem, we make the following basic assumptions, each of which is properly justified.

- The data we collect is reliable and valuable because it comes from the authoritative web-site related to roller coasters.
- We don't consider the defects in the roller coaster design process, so we assume that the design parameters are reasonable.
- We do not consider the psychological impact of height or rollover on passengers, so we assume that the various actions of the roller coaster are within the passenger's tolerance.

4 Data analysis and preprocessing

4.1 Data Analyzation

In this section, we will analyze the data in the appendix, which will help us know the methods of data preprocessing.

- To begin with, row one and row three in the data sheet provided (COMAP_RollerCoasterData_2018.xlsx) mainly describes the name and park the roller coaster is in. It is also recognized that there exists an overlap in naming, and more than one roller coaster in the same park.
- Row 3 to 6 describes the **Geographical location** of the roller coaster, which are descriptive indicators. In question 3 we will utilize these indicators, yet we will not use these indicators in the objective ranking system in question one.
- The **Construction** and **Type** will be explained in 5.2.1&5.2.2
- The row "Status" is the type of operating of the roller coaster, which will not affect the ranking. The data will not be included in our evaluation system.
- The row **"Year/Date Opened**" provides the data of age, we regard the roller coaster that is constructed near the present to have more advanced technology. It is thus a maximal type of data. Also, by assuming that the design of roller coasters is safe, data in the row "Height", "Speed", and "Length" can be utilized directly, and the total score is higher if the data is larger. They are great type indices as well.
- Row "**Inversion (YES or NO)**" and "**Number of Inversions**" can be combined together. The influence can be reflected by only the data in "**Number of Inversions**". The higher the value, the higher the score the rollercoaster will get through our model. This part is detailed explained in 5.2.3.

- The row "**Drop (feet)**", "**Duration(min:sec**)", **"G-Force**", and "**Vertical Angle** (degrees)" include a large amount of missing value, which are very essential factors that influence the ranking. Thus, we fill the missing value in 4.5 "filling missing data"

4.2 Dimensionless treatment

In general, each evaluation index is not commensurable because of the difference between the unit of measure and the order of magnitude. In order to eliminate such influence and correctly reflect the actual situation, we need to make indexes (e.g. height, speed, length, etc.) being dimensionless, which is the standardized treatment of index data.

We adopt the min-max normalization method to process the data of the indicators through the dimensionless treatment:

$$F_i' = \frac{F_i - F_{i\min}}{F_{i\max} - F_{i\min}}$$

F_i' is the data which is dimensionless data of each indicator. $F_{i\max}$ and $F_{i\min}$ are the maximum and minimum value of the indicators respectively.

4.3 Handling the outliers

To avoid the influence of outliers in the original dataset, we use Z-score to identify them. The calculation of Z-score is by using the following formula (OSWEGO)

$$\mathrm{Z} = \frac{\mathrm{Value} - \ \mathrm{Mean\ \ of\ \ Values}}{\mathrm{Standard\ \ Deviation\ \ of\ \ Values}}$$

It determines how many standard deviations the value is from the mean. If the Z-score of the data falls outside 3 and -3, it will be considered as an outlier. An example of an outlier is the number of inversions of a roller coaster called Smiler which is 14. The Z-score is 4.52. To handle this kind of outliers, we delete them and then process them in missing value imputation.

4.4 Data consistency

In a complex comprehensive evaluation problem, it may contain many types of indicators such as maximal index, minimal index. In order to facilitate the Advantages and Disadvantages of its index, we changed minimal indexes into maximal indexes.

For a minimal index x_j, the smaller value is better. To turn it into a minimal index, we only need to convert the index to the reciprocal transformation or translation $x_j' = M_j - x_j$ where $M_j = \max_{1\le i\le n}\{x_{ij}\}$.

4.5 Fill missing data

Because of missing information or other reasons, the data obtained from the website database includes a few missing data in the height、speed、length three indicators; the data of the four indicators of the drop, duration, G-force, and vertical angle show a large scale of missing data.

As the degree of data missing is different, we use different processing methods.

Method 1: For a small amount of missing data, we use the average for substitution.

Method 2: To cope with large-scale missing data, the mean substitution method does not apply. A relationship between existing data and missing ones should be found to calculate and substitute the missing data. After comparing several different methods, we finally decide to use BP neural networks to fill the missing data.

4.5.1 BP Neural Network

The classic BP Neural Network usually consists of three layers: input layer, hidden layer, and output layer. The number of neurons in the input layer is usually relative to the eigenvalues; the number of outputs is consistent with the number of categories; the number of layers in the hidden layer and the number of its neurons can be customized.

We use a trained BP Neural Network to deal with regression. Each sample obtains outputs, and the network obtains input neuron and one output neuron respectively.

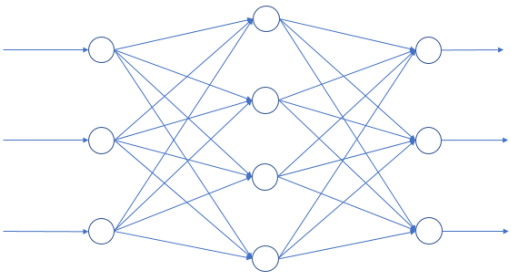


Fig1. BP Neural Network

4.5.2 Feed-Forward process

The relation between the number of neurons in each hidden layer and input layer can be described in the function as follows

$$I_j = \sum_i W_{ij} O_i$$

$$O_j = sig\,\mathrm{mod}\left(I_l\right) = \frac{1}{1+e^{I_l}}$$

W_{ij} represents the connection weight between neuron i and neuron j , O_j represents the output of neuron j . $sig\,\mathrm{mod}$ is a special function which maps the arbitrary real number into the (0,1)interval.

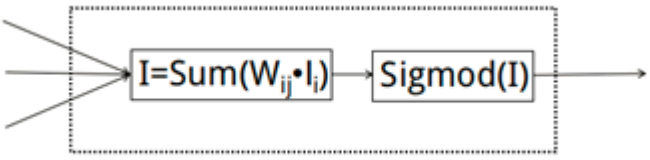


Fig2. Function graph

The $sig\,\mathrm{mod}$ function is called an activation function. In the practical application, we usually place a biased neuron as an adjustable input correction, or an offset parameter for each neuron in the hidden layer. We place n eigenvalues in the input neuron, the output layer calculates the regressed value according to the processed value by the hidden layer.

The steps above are called Feed-Forward process, where the inputs and outputs are nearly the same with the ones in multivariable functions.

4.5.3 Training process

The current problem is how to train the neural network.

BP neural network training is to compare the prediction value in the feedforward as well as the reference value. It also can adjust the connection weight W_{ij}. The process is called Back Propagation; the data stream is the inverse of the feedforward.

Step1: At first, we randomly initialize the connection weight W_{ij}, and obtain the output of each neuron from one feedforward of one certain training sample.

Step2: To begin with, calculate the error of output layer:

$$E_j = sig\bmod'\left(O_j\right)*\left(T_j - O_j\right) = O_j\left(1-O_j\right)\left(T_j - O_j\right)$$

E_j is the error of neuron j, O_j represents the output of neuron j, T_j refers to the output of the current training sample, $sig\bmod'\left(x\right)$ is the first derivative of the $sig\bmod$ function.

Step3: Calculate the error in hidden layers:

$$E_j = sig\bmod'\left(O_j\right)*\left(E_k W_{jk}\right) = O_j\left(1-O_j\right)\sum_l E_k W_{jk}$$

If there is no reference value in the hidden layer, we use the weight in the next layer $\left(T_j - O_j\right)$ as the weight and substitution.

After calculating the error we adjust W_{ij} and θ_j to:

$$W_{ij} = W_{ij} + \lambda E_j O_i$$

Where λ is a parameter called learning rate, which usually falls in interval (0, 0.1).

To accelerate the learning, we use the correction matrix, which records the value in the previous propagation process. The function is thus adjusted as:

$$W_{ij} = W_{ij} + \lambda E_j O_i + \mu C_{ij}$$

μ is a parameter called correction rate. The correction matrix is regenerated:

$$C_{ij} = E_j O_i$$

4.5.4 Convergence process

Each sample will renew the parameters in the whole network. We need additional conditions to stop the test. The simplest way is to set the maximum number of irritations. However, it can't ensure the accuracy of the results. One of the better methods is to use loss function as the condition to stop training.

Loss function can choose the variance of each node:

$$L = \sum_j \left(T_j - O_j\right)^2$$

We usually extract a part of the results for testing to avoid meaningless iteration. When the prediction error is higher than the threshold, the testing is stopped.

4.5.5 Error analysis about filling Data

In this problem, we set the inputs as Height, Speed, Length, and Number of inversions which creates the.

We use the drop indicator as an example to demonstrate the process of verification. Part of the data is extracted from the sample as the training set, and the remaining part is used as testing set.

Step 1: Error analysis of the set as the number of iteration increases.

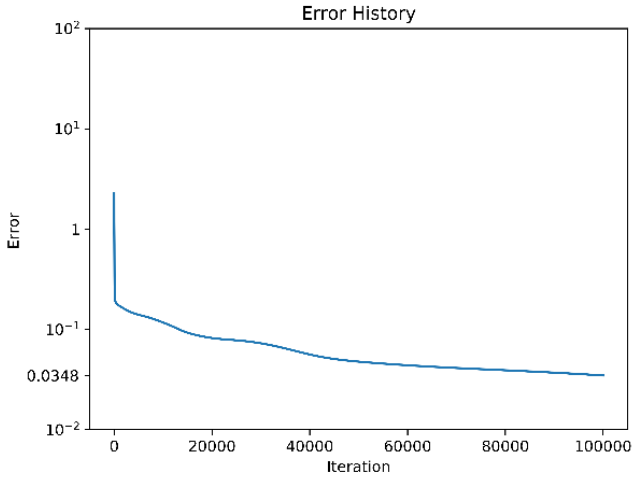


Fig3. Drop indicator neural network error analysis

It can be inferred from the figure that the error between the calculated value and practical value will be reduced greatly as the number of iteration increases. When the iteration reaches more than 100 thousand, the error is reduced to 3.48%, which falls within the acceptable range, and we will consider this trained neural network to be accurate, which is close to the practical results.

Step 2: Our goal is to obtain a viable neural network through effective iterations of the training set. To achieve the goal, we use the test suite to testify if the difference of calculated value and practical value is among the acceptable range.

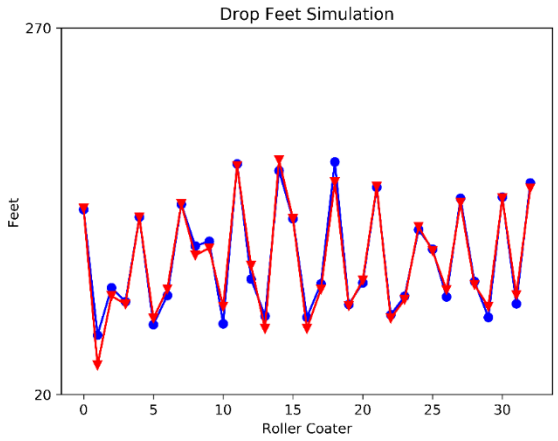


Fig4. Neural network error comparison

The comparison of the calculated value and test suite is demonstrated in the figure. The red broken line represents the test set result, and the blue broken line represents the training result. Obviously, the data calculated by the neural network is highly fitted to the actual data, which further indicates that our neural network is mature, so we can use this neural network to get large areas of missing data.

5 Roller coaster comprehensive evaluation model

5.1 Preliminary selection of indicators

The rational indicator system is an important step in the establishment of our roller coaster comprehensive evaluation model. Therefore, we need to consider whether the given indicators are available, whether or not there are better and more reference indicators that need to be included in the indicator system. We will do a detailed analysis of the selection of indicators and the establishment of the indicator system below.

We use the indicator given in the problem and the ones we search on our own as the preliminarily selected indicators.

Table1.All indicators

City/Region	Geographic Region	Construction	Type
Status	Year/Date Opened	Height	Speed
Length	Inversions (YES or NO)	Number of Inversions	Drop
Duration	G Force	Vertical Angle (degrees)	Park
Sales volume	Price	Favorable comment	Ride limit

5.2 Indicator analysis and screening

5.2.1 Analysis of the 'Construction' indicator

We analyze the three indicators of height, length and speed of wood and steel respectively, trying to find the correlation between construction and roller coaster parameters.

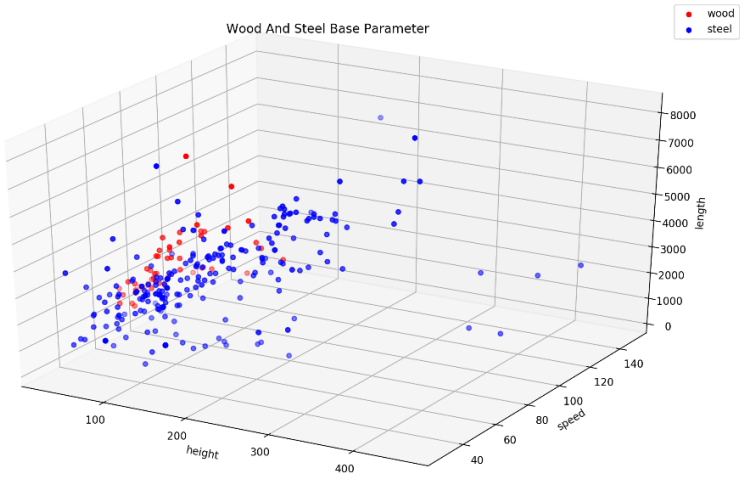


Fig5. Data distribution analysis

After plotting the data in three dimensions, we find that the steel roller coasters have comparative advantage against the wood one, which means the design parameters can affect user experience. Furthermore, we can infer from the graph that construction factor has a significant impact on the parameters of roller coasters.

Considering the strong correlation between "construction" and "design parameters" of rollercoasters, we treat wood and steel rollercoasters separately in following evaluation.

5.2.2 Analysis of the 'Type' indicator

There are many types of roller coasters bringing different sensations, such as "Sit Down", "Inverted", "Stand Up", "Suspended", and "Flying"

For example, Wing Coaster's design is different from many traditional steel roller coasters because its train is not located above or below the track, but on the side. This design can make the players feel as they are flying. Regardless the construction of rollercoasters, Inverted and Suspended design each gives passengers a sense of overturning.

Therefore, we conclude that the "type" of rollercoasters affects its ranking in two ways. For one, customers have inclination or preference toward certain types. For the other, the type of rollercoasters affects the stimulation of the riding experience. To take this into account in our evaluation model, we construct a weight vector to express the importance of type in the ranking of rollercoasters, noted as:

$$T_{ir} = [1.00 \quad 1.15 \quad 1.10 \quad 1.10 \quad 1.20]$$

Weight of the i^{th} roller coaster's type recorded in the weight vector. The weights are respectively correspondent to "Sit Down", "Inverted", "Stand Up", "Suspended" and "Flying".

In the calculation of final result, we will consider the effect of "type" factor of rollercoasters by applying weighted summation.

5.2.3 Inversions (YES or NO)& Number of Inversions

One of these two indicators is descriptive, and the other is numerical. They describe the same situation. "Inversions (YES or NO)", "Number of Inversion", are representative of whether there are inversion 0-1 variables, "Number of Inversion" is the number of inversions of the track of the roller coaster, so the data in these two columns can be aggregated into one numerical column.

5.2.4 Screening of indicators from feasibility

The feasibility requirements for indicators mainly include two aspects:

- Indicator data is easy to collect and quantify.
- In line with the current statistical methods, adopt sophisticated and general acknowledged indicators.

Based on the above feasibility requirements, we consider eliminating the following indexes:

Table2. Rejected Index

Number	Rejected Index	Reason for Rejection
1	Sales volume , Price	Difficult to obtain credible data
2	Ride limit , Favorable comment	Indicator data is difficult to quantify

5.2.5 Indicator screening based on Grey Correlation Analysis

In this essay, we choose grey correlation to screen the indicators, which can avoid duplication , to ensure the importance of our selected indicators to decision variables.

To begin with, the comparison object (evaluation object) and the reference number column (evaluation criteria) are determined. Then we construct the standard sequence and the comparison sequence for analysis.

Table 3 Reference number column & Comparison object

Category	Indicator	Mathematical representation
Reference sequence	i	$x_0^{(i)} = \left\{ x_0^{(1)}, x_0^{(2)}, \cdots, x_0^{(k)} \right\}$
Comparison sequence	j	$x_0^{(j)} = \left\{ x_1^{(1)}, x_1^{(2)}, \cdots, x_1^{(k)} \right\}$

The absolute difference between reference number column and comparison object is:

$$\Delta_i(k) = \left| x_0(k) - x_i(k) \right|, \left(k = 1, 2, \cdots n; 1 \le i \le m \right)$$

$\Delta_i(k) = \left(\Delta_i(1), \Delta_i(2), \ldots, \Delta_i(n) \right)$ is Differential sequence

According to the definition above, the correlation between comparison object x_i and the reference number x_0 at point k is:

$$\xi(x_0(k), x_i(k)) = \frac{\min\limits_i \min\limits_k \left| x_0(k) - x_i(k) \right| + \rho \max\limits_i \max\limits_k \left| x_0(k) - x_i(k) \right|}{\left| x_0(k) - x_i(k) \right| + \rho \max\limits_i \max\limits_k \left| x_0(k) - x_i(k) \right|}$$

The respective correlation R is:

$$R\left(x_0, x_k \right) = \frac{1}{n} \xi(x_0(k), x_i(k))$$

According to the model above to solve the problem:

Step 1: Solve the initial value, let

$$x_i' = \frac{x_i}{x_i(1)} = \left(x_i'(1), x_i'(2), \cdots, \ x_i'(n) \right), i = 0, 1, \cdots, n$$

Step 2: Calculate the grey correlation coefficient:

$$\xi_{ij}(t) = \frac{\Delta_{\min} + \rho \Delta_{\max}}{\Delta_{ij}(t) + \rho \Delta_{\max}}$$

which $\Delta_{\min} = \min\limits_i \min\limits_k \left| x_0(k) - x_i(k) \right|$ is the minimum value of the difference sequence.

$\Delta_{\max} = \max\limits_i \max\limits_k \left| x_0(k) - x_i(k) \right|$ is the maximum value of the difference sequence.

$\Delta_{ij}(t) = \left| x_0(k) - x_i(k) \right|$ is the value of difference.

ρ is the resolution coefficient which falls within [0,1]. In this essay, it is 0.5. When $\Delta_{ij}(t)$ increases, $\xi_{ij}(t)$ decreases, and vice versa. The value of $\xi_{ij}(t)$ can thus demonstrates the correlation between two sets of data.

Step 3: Criteria for correlation (Levels of Correlation Degree (in Appendix table4))

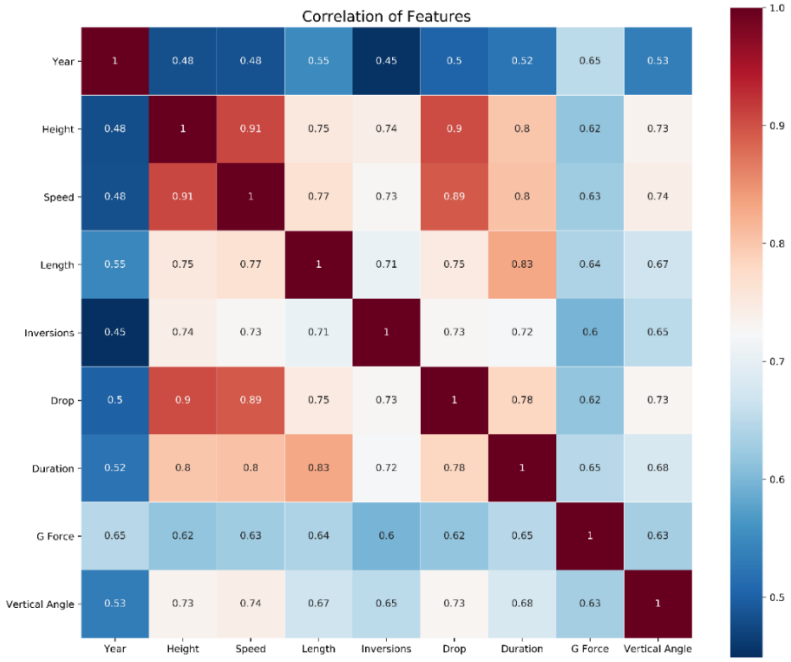


Fig6. Index correlation analysis

Through grey correlation, we reject the following indicator which has high correlations: speed, length, and drop.

5.2.6 Comprehensive Evaluation Index System

Our aim is to find some indicators which are measurable and suitable. We use them to judge good or bad of roller coaster.

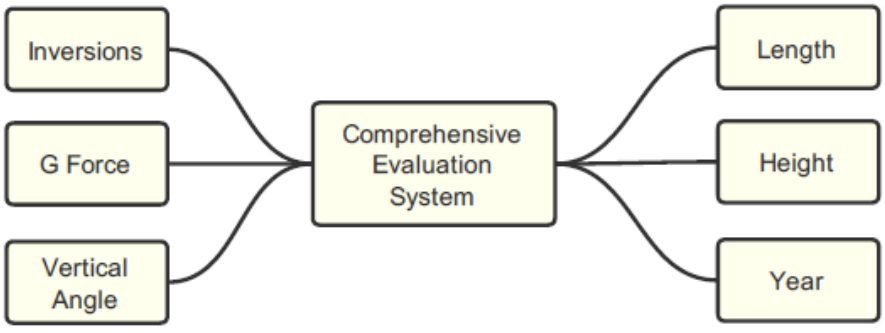


Fig7. Comprehensive evaluation system

This comprehensive evaluation system is obtained after screening and is important for our subsequent solution.

5.3 Comprehensive Evaluation Model based on Entropy method

We consider several weighting methods which include Analytic Hierarchy Process (AHP), yet the method refers to the experts 'opinion and reduces the objectivity of the evaluation system. So we ultimately choose Entropy Method.

The principle of entropy weight method is to quantize and synthesize the information of each unit to be evaluated. Entropy weight method can simplify the evaluation process

Step 1: Construct the primary data matrix

Firstly, by the above related evaluation indexes, the original data matrix can be obtained through the data of each index of roller coaster

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1j} \\ x_{21} & x_{22} & \cdots & x_{2j} \\ \vdots & \vdots & \vdots & \vdots \\ x_{i1} & x_{i2} & \cdots & x_{ij} \end{pmatrix}_{i\times j}$$

The matrix is normalized, i.e. the ratio of the column vectors in the matrix to the sum of all elements in the matrix is taken as the normalization result. The formula is as follows:

$$z_{ij} = \frac{x_{ij}}{\sum_{i=1}^{n} X_{ij}}, \left(j = 1, 2, \cdots, m\right)$$

where， z_{ij} is the element after normalization

Step 2: Determine the entropy weight of the evaluation index

$$H\left(x_j\right) = -k\sum_{i=1}^{n} z_{ij} \ln z_{ij}, \left(j = 1, 2, \cdots, m\right)$$

k is poundage factor which $k = 1/\ln n$ ； z_{ij} is the value of standardized indicator j in the i evaluation unit.

Step 3: Convert the entropy value of evaluation index to the weight value.

$$d_j = \frac{1 - H\left(x_j\right)}{m - \sum_{j=1}^{m} H\left(x_j\right)}, \left(j = 1, 2, \cdots, m\right)$$

where $0 \le d_j \le 1$， $\sum_{j=1}^{m} d_j = 1$.The weight value is then obtained.

Step 4: Add the weight value T and calculate the total score

$$Q_i = \sum_{j=1}^{6} d_{ij} z_{ij} T_i$$

5.4 Data dimensionality reduction based on PCA Analysis

Principal Component Analysis (PCA) attempts to optimize the multivariate cross-section data table under the principle of ensuring the least loss of data information. It can reduce the dimensionality of high-dimensional variable space, and it is more objective and scientific. Therefore, this paper chooses the method of principal component analysis for dimensionality reduction.

Determining p PCs and perform statistical analysis

PCA converts numerous indices into several weakly correlated comprehensive indices. PCs and corresponding accumulative contribution rate are given below.

Table 4 PCs and Corresponding Contribution Rates

PC	Contribution Rate (%)	Accumulative (%)
1	35.617589	35.617589
2	31.459494	67.077083
3	12.448942	79.526025
4	11.050469	90.576494
5	4.946335	95.522829
6	4.477171	100

Under accumulative variance of 90.576% we obtain 4 PCs. These 4 PCs provide 90.576% information of the indices, which satisfies the principle of PCA. Hence we've realized the dimensionality reduction of indices and simplified further calculation followed.

6 Ranking Result Interpretation

6.1 Calculation of rank according to our model

We take the roller coasters with 10 highest score which we get from the comprehensive model in section 5 as our top-ranking list. The ranking list is in figure 8

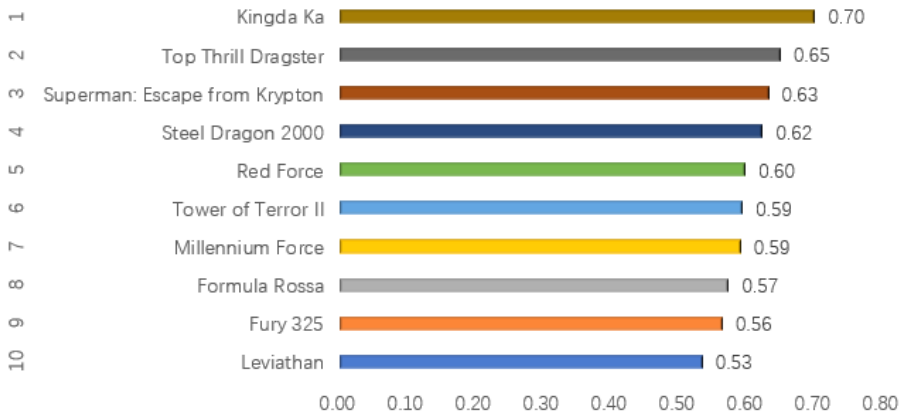


Fig8.Our rank

In our ranking, Kingda Ka is the NO.1 roller coaster, at the same time it is also the highest roller coaster.

So the main features of our own ranking will be analyzed in 6.3.

6.2 Evaluation Model Inspection

We need to determine whether the sensitivity test of our comprehensive evaluation model is qualified by studying whether the change of evaluation index has a significant impact on the ranking of roller coasters. We examine the sensitivity of the model from the following two methods.

Sensitivity analysis of index values can be carried out between two evaluation objects, or all index values can change at the same time according to a certain percentage; it can be either a single index change analysis or multiple index change analysis. In this paper, the overall synchronous change of the index value is taken as an example to analyze the sensitivity of the single index value and the combined index value.

Assume there are i number of roller coasters, j number of evaluation indicators, were $z_{i1}, z_{i2}, z_{i3} \cdots z_{ij}$ respectively. In application 2, the weight of each index is obtained by solving the result of each index are $d_1, d_2, d_3 \cdots d_j$ respectively，obviously $d_1, d_2, d_3 + \cdots + d_j = 1$，Evluation value Q_i is：

$$Q_i = (d_1 z_{i1} + d_2 z_{i2} + \cdots + d_j z_{ij}) \bullet T_{ir}$$

For evaluation value m，take into account the increase or decrease of all evaluation objects according to each percentage, assuming changes in range r_j and the new evaluation value would be:

$$Q_i' = (d_1 z_{i1} + d_2 z_{i2} + d_j z_{im}(1 + r_i) + \cdots + d_j z_{ij}) \bullet T_{ir}$$

To conduct sensitivity analysis for change of single indicator z_{ij}. First, the original evaluation results are sorted and the ordinal number of each evaluation object is marked. Then set up r's initial value r_0 and the final one, also set up step length r，and loop, always calculate the evaluation value after the varying of z_{ij}, and then sort them again, and compare the consistency between the new sort and the original one, stop the cycle once it is inconsistent, the value r_j at this time is the threshold of the critical point. Since the variation of r_j can either be increasing or decreasing, so we should calculate separately.

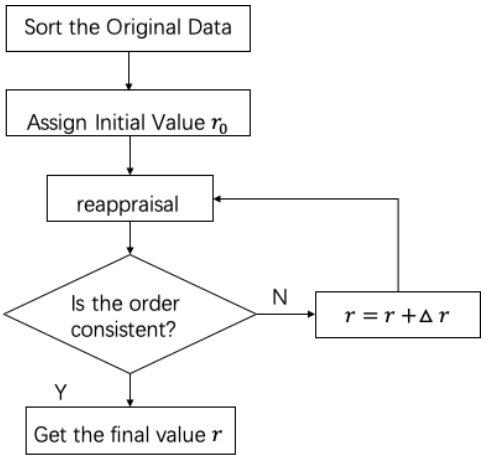


Fig9. Flowchart of the index sensitivity analysis

The flow chart of sensitivity analysis of index values is shown in the figure. To analyze the sensitivity of each indicator z_{ij}, we first sort the original results of evaluation, and number all the evaluation object. Then, we set the initial value of r as r_0 and the terminal value, then set the step r_j and circulate the ranking. Each time calculate the value of z_{ij} and reorder the ranking.

6.2.1 Sensitivity Model Solving and result analysis

The maximum standardized data in this essay is 100, when doing the indicator sensitivity analysis. As there are increases and decreases in the value of the indicators, there are two sets of parameter-setting methods: when the data increases, the initial value is 0.01, the step is also 0.01, and the terminal value is 100; when the data is decreasing, the initial value is -0.01, the step is also -0.01, and the terminal value is -100.

Table5. Single indicator sensitivity result

Indicator variable	Value Fluctuation hreshold（%）		Range（%）
1	-4	2	6
2	-14	10	24
3	-8	11	19

From the above table, we can see that the sensitivity range of the roller coaster height is -4%-2%, which have a higher sensitivity index, while the threshold of the built year and length is -14%-10% and -8%-11 respectively, which have a lower sensitivity.

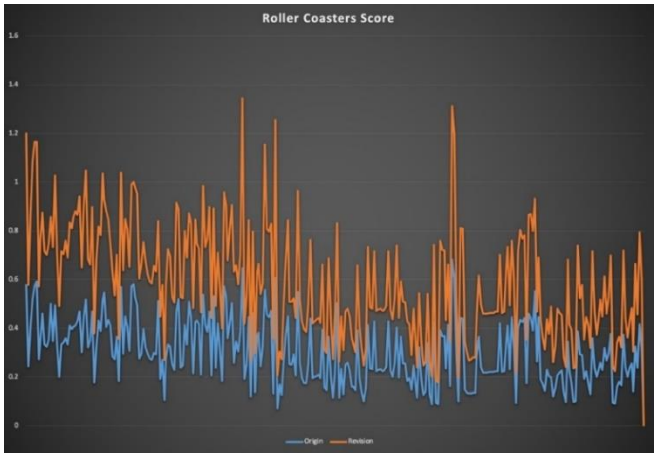


Fig10. Stability analysis

In order to observe the impact of adjusting the impact of the roller coaster height indicator on the overall score, the original score and the adjusted score map in the sensitivity test were compared. The blue line is the original score and the red curve is the processed score curve. From the figure, we can clearly observe that although the curve has improved overall, the trend has not changed significantly. This shows that our ranking system is stable and will not overly affect the overall ranking because of an indicator.

6.3 Comparison to existing ranks of rollercoaster

We found two rankings online, the first one is a ranking created by college students at University of Buffalo, the second is from ranker.com, where the public can set approve or

disapprove of the ranking. We compare the results to the results we get from the objective model. Table 6 shows the original rankings and the rankings on our own list.

Table6 Online Ranking 1& Online Ranking 2

Name	Online Ranking	Our Ranking	Name	Online Ranking	Our Ranking
Millennium Force	1	7	Millennium Force	1	7
Top Thrill Dragster	2	2	Steel Vengence	2	34
Superman - Ride of Steel	3	77	Top Thrill Dragster	3	2
The Beast	4	133	Maverick	4	103
Oblivion	5	172	El Toro	5	244
Superman - the Escape	6	3	Fury 325	6	9
Griffon	7	43	Intimidator 305	7	11
Goliath	8	65	The Voyage	8	95
Son of Beast	9	unknown	Kingda Ka	9	1
Superman - Ride of Steel	10	72	Apollo's Chariot	10	76

There are 3 roller coasters on the list of first ranking that also exists on our own list. This includes Millennium Force, Top Thrill Dragster, and Superman - the Escape. Top Thrill Dragster both becomes the second place of the list. There are 4 roller coasters, Millennium Force, Top Thriller Dragster, Fury 325 and Kingda Ka, exist on both the second online ranking and our own list. The well-recognized top roller coaster from the 3 rankings are Millennium Force and Top Thriller Dragster. Their information is on table 1 in the appendix. While their overall score is high in our own ranking, they process impressive context which makes them stand out.

Millennium Force was the world's first giga coaster, which exceeds 300 feet in height and complete a full circuit. The ride is also the third roller coaster in North America. Kingda Ka is the world's tallest and second fastest roller coaster.

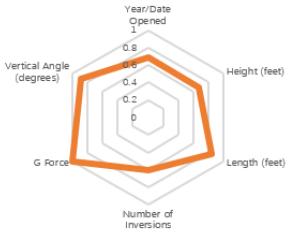
The tendency to put the roller coasters with the special figure (highest, longest, etc.) on the top makes this occur.

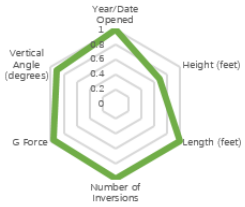
The rankings share different characteristics as well. Based on the data table 2 in the appendix, we create radar charts to visualize the characteristics of mean data in different rankings. We will analyze their similarities and differences.

Table7.Mean Data for the rankings

Name	Online Rank 1	Online Rank 2	Our Ranking
Year/Date Opened	1999.4	2005.85	2005.6
Height (feet)	235.7	238.2	346.6
Length (feet)	3808.1	4550.8	4465.3
Number of Inversions	0	0.75	0
G Force	3.994	3.855	3.983
Vertical Angle (degrees)	79.9	81.51	89.822
Score	0.433	0.446	0.604

Table 7 shows the quantitative data of the top roller coasters from two online ranking and the one we adapt from our own. To discover the different characteristics, we use radar charts to visualize the difference. (Figure 11-1,2,3)





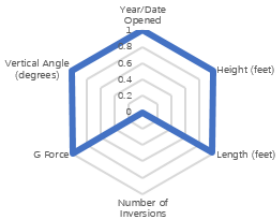


Fig11-1. online rank 1 Figure11-2. online rank 2 Figure11-3.our rank

The average age of the roller coasters in the first rankings is the largest among the three rankings, which will result in a lower score in the evaluation system we created.

In fig11-3 , we can see that the main difference is in the number of inversions. The top roller coasters in our own ranking contain no inversions, while some in the first and second online ranking have inversions. The correlation of inversions and height is 0.7 as mentioned in the (5.2.5). Even though in our own evaluation system, the data of inversions are great type index, meaning that the larger the value, the higher the score, the construction of inversions may influence the other key factors (height, for instance) and lower the overall score for the roller coaster. The top roller coasters in our own evaluation system score the highest in the other 5 criteria – age, vertical angle, height, G force, and length. Though the number of inversions influence the key factors from the objective view, it increases the excitement of the ride of roller coaster greatly from the subjective view. This may result in the variation of our objective ranking and the subjective online rankings.

Apart from the quantitative data above, we can also see that the top roller coasters on the online rankings are mainly from United States (In Appendix table2). We didn't consider the effect of region on the ranking of the roller coaster as it doesn't influence the actual function directly. Still, the most roller coaster from the data set are from United States. Figure 12 shows the distribution of roller coasters directly based on the dataset given. The more roller coasters that exists, the redder the value. The roller coasters from the data set, as noted, are the ones which data exceeds the average level, and can be regarded as the top roller coasters in the world.

Fig12. Roller coaster distribution in world region

Even though the U.S roller coasters have a higher possibility to dominate the top rankings, the roller coasters in the other countries which process advanced structure are excluded for other reasons. To begin with, people would tend to give higher rankings for the ones that are familiar to the majority people. As the roller coasters are distributed all over

the U.S nations, tourists have easy access to the high-performance ones, making them famous among a lot of people. Also, the people who do the rankings are Americans – the first one is created by U.S university students and the second one is created by a website which is based in Los Angeles. They might process natural bias towards the roller coasters in the other countries, or lack of information as they are not familiar to them. Last but not least, new roller coasters are produced each year, yet the old ones are not replaced respectively, the rankings that are not so up-to-date would reserve the roller coasters that used to be the most advanced ones. As we can see from the table 3 (In Appendix), most of the roller coasters are opened before the year 2000, near 20 years ago from now. Only one of them is constructed in 2018.

Fig13-1

Fig13-2

Visual impact is usually the biggest subjective determinant that influence the degree of thrilling when people are evaluating the roller coaster. Complex structure and interweave lines make a roller coaster more spectacular than others. Due to this reason, people are more likely to treat the wooden structure as a more thrilling design (Figure 13-1,2). Hence wooden structure occupied more spaces than using steel which brings people a misconception that it will be more dangerous.

However, since our model only determine the thrilling index from objective perspectives, meaning that it won't be influenced by a few criteria even though some of the roller coaster are exceptional in these parts, especially factors like visual impact which containing subjective factors. So, the ranking from our model are more accurate and objective than online rankings and shows more scientific result.

6.4 Conclusion

Our ranking is partially different and partially overlap with the rankings online. This is mainly because of the different criteria we hold. In addition to the parameters that cause the overlap, factors like economy, tourism industry, geographical location, advertising, and so on would also drive the deviation in the result. These factors are not reflected in the evaluation system which is based on the completely objective performance of the roller coasters.

7 Designing a User-friendly APP

7.1 Demand Analysis

In order to design a user-friendly app, it's crucial that we know what type of users we are designing for. Next, we should figure out their core pain-points and needs regarding rollercoasters. Therefore, this section will analyze of the portrait of our users as well as their demand that contributes to the figuration of our approach.

Portrait of Users

Because the App is relatively focused on one aspect, rollercoasters, users who initiatively download the App are possibly interested in roller coasters in the first place or have been recommended by their friends. Accordingly, we come to the assumption that majority of our users is the young generation that actively seek for a rollercoaster experience. Because the generality of target customers of rollercoasters, the age range and gender inclination are not very restricted for our App, therefore we should also consider a wider context of the users. Nevertheless, we can start from the typical user portrait of young customers shown in the figures below, including both genders:

Fig14.Typical User Portrait

Main Demand

To begin with, users may want to find the appropriate (accessible) roller coasters which they would most likely try.

Besides, they'd like to explore roller coasters with different features, for interest or potential future visits as well.

Finally, they're tending to share experiences and comments of rollercoasters via the platform of RollerOutstander.

Pain Points

Current platforms provide limited information source and experience about the roller coasters that users are not familiar with. They require users to initiatively locate rollercoasters instead of generating customized recommendations and thus create hindrance for unexperienced users, which harm the experience.

7.2 Key features

First, we include a **user-adaptive filtering algorithm** that allows the user to get customized ranking results of rollercoasters based on their own preferences of indices. This contributes to the solution for "vague meaning of ranking". (section 7)

Next, in order to reach proactively to our user, we include a **recommendation system**. This system is self-learning, and adjustable to the user's personal profile and preferences. (section 8)

Finally, we incorporated the key features into a **user-friendly interface** so as to enhance the **user experience** using RollerOutstander. To illustrate in detail, we included a **typical user case** and conducted **reflection** on our design of the App that can lead to future updates. (section 9)

8 User-Adaptive Filtering System

8.1 Establishing New Evaluation Model

Although our model in the first question is relatively objective, but because of the objective loss of many evaluation indicators, in a mature, intelligent roller coaster selection app, the ranking order recommendation must be combined with people's subjective will, geography Location and objective conditions, such rankings are valuable for people who want to ride a roller coaster. Therefore, we improved the evaluation model in the first question and added the fuzzy comprehensive evaluation model.

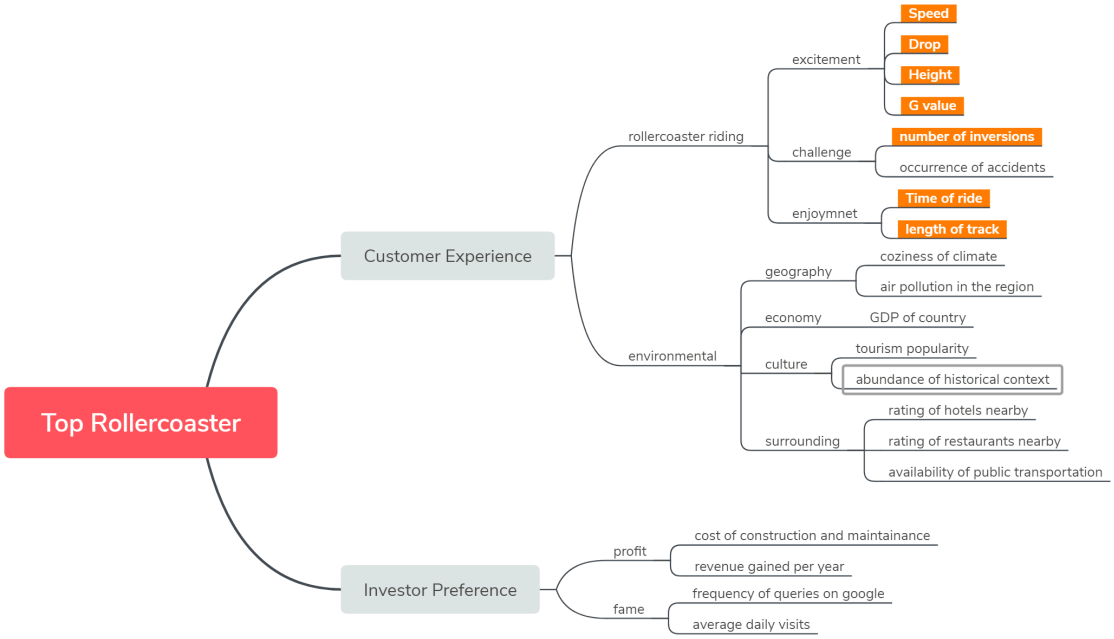


Fig15. New comprehensive evaluation system

Step1：Determining factor set

For example, type, construction, security, peripheral integrity, etc. All these factors constitute a collection of evaluation indicator systems, namely the set of factors, recorded as:

$$U=\{u_1,u_2,\cdots,u_n\}$$

Step2：Confirm the comment set

Due to the different evaluation values of each indicator, different levels are often formed. For example, the integrity of the surrounding facilities can be divided into fantastic, good, medium, poor, and poor. A collection of different decisions is called a collection of comments, which is recorded as:

$$V=\{v_1,v_2,\cdots,v_n\}$$

Step3：Determine the weight of each factor

Under normal circumstances, the factors in the concentration of factors play a different role in the comprehensive evaluation. The comprehensive evaluation results are not only related to the evaluation of various factors, but also to a large extent and the role of various factors in the comprehensive evaluation, this needs to determine the weight distribution between each factor, which is a fuzzy vector on it, recorded as:

$$A=[a_1,a_2,\cdots,a_n]$$

Step4：Determine the fuzzy comprehensive judgement matrix for the indicator

For the indicator u_i, the membership degree of each comment is the upper fuzzy subset of V. The evaluation of the indicators is recorded as:

$$R_i = \left[r_{i1}, r_{i2}, \cdots, r_{im}\right]$$

The fuzzy comprehensive judgement matrix of each indicator is as follows, which is a fuzzy relation matrix from U to V:

$$R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1m} \\ r_{21} & r_{22} & \cdots & r_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n1} & r_{n2} & \cdots & r_{nm} \end{bmatrix}$$

Step5：Comprehensive Evaluation

For a fuzzy relation $R = \left(r_{ij}\right)_{n\times m}$, we can obtain the fuzzy transformation $T_R : F(U) \to F(V)$ using R. Then the comprehensive evaluation result $B = A \bullet R$ is obtained. The comprehensive evaluation can be regarded as a fuzzy vector on V, noted as $B = \left[b_1, b_2, \cdots, b_n\right]$.

Such a fuzzy evaluation system combined with the entropy weight method constitutes a comprehensive evaluation model including both subjective and objective factors, and is applied to the internal ranking calculation of the app.

8.2 Algorithm Procedure

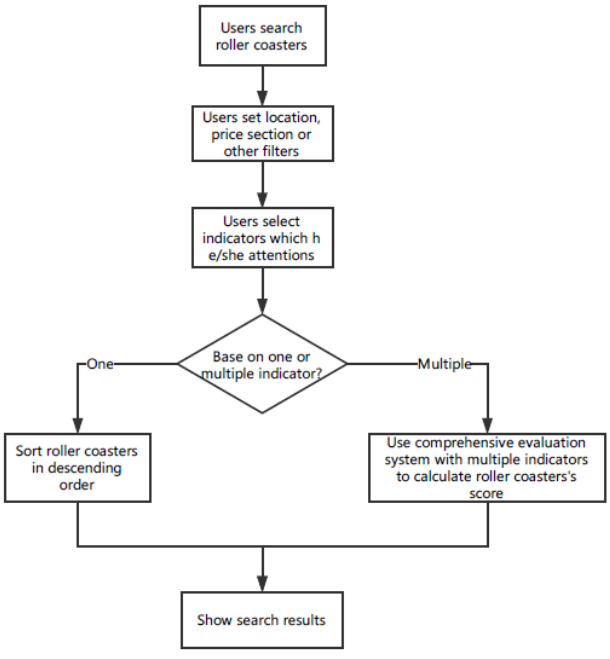


Fig16. Algorithm flowchart

The users take initiative to interact with the filter algorithm, and decide on whether filter the result with one or multiple factors. Based on user's specific demand and settings, we apply the AHP model described above to calculate customized rollercoaster ranking result and feed it to the user.

9 Recommendation System (RS)

9.1 User-friendly basis

The app needs to be user friendly to maintain effectiveness in its function. We define "user friendly" as efficient, comprehensive, and effective. Based on these 3 features, we develop an app called "RollerOutstander" which helps the users to find the roller coaster that they would be mostly likely to take a ride easily.

9.2 Concept and Structure

9.2.1 General Concept for RS

Recommendation system is a tool for automatically contacting users and items. It can help users find information that interests them in the environment of information overload, and push information to users who are interested in them.

There are mainly three strategies for RS. The first strategy is to locate the items that the user prefers, and then recommend similar items; the second strategy is to locate the user's similar users and then recommend the items preferred by these users; the third strategy is first to spot characteristics between users and items, and then recommend the items which have common characteristics with students. Various algorithms and approaches can be adopted to generate recommendations for target users via either of the three approaches.

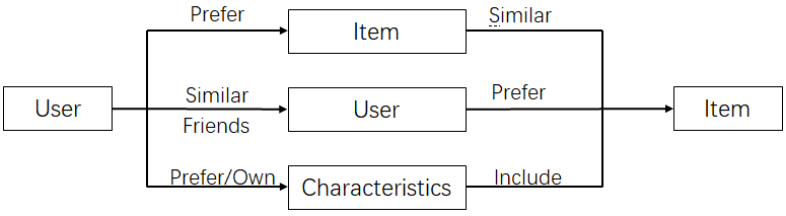


Fig17. Recommendation process

9.2.2 Recommendation System and Recommendation Engine Structure

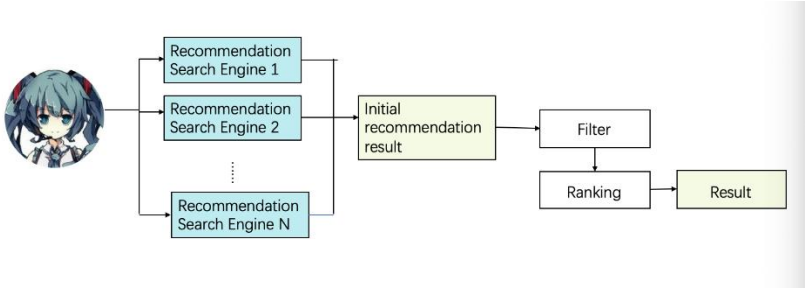


Fig18. Recommendation Engine Structure

Recommendation via single algorithm or approach is seldom optimal, therefore several recommendation engines are synthesized in one recommendation system in order to create comprehensive set of recommendations. A filter and ranker then refines the initial recommendation result set and feed final recommendation result to the users.

Based on consideration of our specified need regarding the area and scope of the App, we choose to adopt two recommendation engines – ItemCF and Tag-based Recommendation – to recommend rollercoasters to our users. Detailed explanation of these algorithms are given in section 9.3 and section 9.4.

9.2.3 Evaluation of RS

Local Notation 8.3

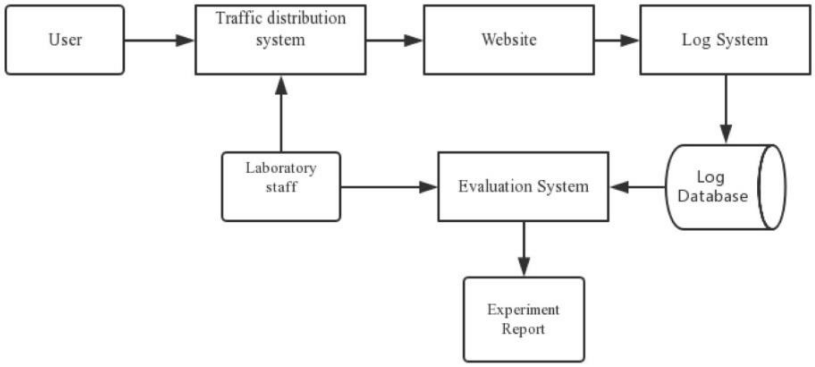


Fig19. Recommendation Engine Structure

When evaluating a RS, part of the users is selected to form an A/B test. Recommendation for each group are generated via different algorithms to test, and user behaviors are tracked for analyzation. The graph is a typical structure of a recommendation system testing.

To evaluate our rollercoaster RS, we must define what's a good RS. Merely the accuracy of prediction is not sufficient for a good RS, due to its purpose not only to predict user's action. More importantly, a qualified RS should provide information that user's not familiar but likely to try, which cause new interaction possibilities and potentially level up the business.

The **recall rate** of the recommendation is defined as:

$$Recall = \frac{\sum_{u \in U} \left| R(u) \cap T(u) \right|}{\sum_{u \in U} \left| T(u) \right|}$$

The **precision** of the recommended results is defined as:

$$Precision = \frac{\sum_{u \in U} \left| R(u) \cap T(u) \right|}{\sum_{u \in U} \left| R(n) \right|}$$

Coverage rate (coverage) describes the ability of a recommendation system to explore the long tail of items. The simplest definition is the proportion of items recommended by the recommendation system to the total set of items. Assuming that the user set of the system is U, the recommendation system recommends an item list R (u) with a length of N to each user. The **coverage** rate of recommendation system can be calculated through the following formula:

$$Coverage = \frac{\left| \bigcup_{n \in U} R(u) \right|}{\left| I \right|}$$

In order to satisfy the broad interests of users, recommendation lists need to be able to cover different areas of interest, that is, recommendation results need to be diverse. Covering most users' interest points will increase the probability of users finding interesting items. Diversity and similarity are corresponding. Assume the similarity between items i and j is defined, and the definition of **diversity** of the recommended list $R(u)$ of user u is as follows:

$$Diversity\left(R(u)\right)=1-\frac{\sum_{i,j\in R(u),i\neq j} w_{ij}}{\frac{1}{2}\left|R(u)\right|\left(\left|R(u)\right|-1\right)}$$

The **overall diversity** of recommender systems can be defined as the average of the recommended list diversity of all users:

$$Diversity\left(R(u)\right)=1-\frac{\sum_{i,j\in R(u),i\neq j} w_{ij}}{\frac{1}{2}\left|R(u)\right|\left(\left|R(u)\right|-1\right)}$$

9.3 Item-based Collaborative Filtering Recommendation (ItemCF)

Table8.Local Notation

$N(i)$	$\{u \mid r_{ui} > 0\}$	Set of users that treat item i positively
r_{ui}	$r \in [0,+\infty)$	User u' s preference of item i
p_{uj}	$p \in [0,+\infty)$	Prediction of user u' s preference of item j

9.3.1 General Concept for Collaborative Filtering (CF)

What is Collaborative Filtering

Collaborative filtering is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc. Collaboratively filtered social media can have a very large number of editors, and content improves as the number of participants increases. Services like Reddit, YouTube, and Last.fm are typical examples of CF-based media.

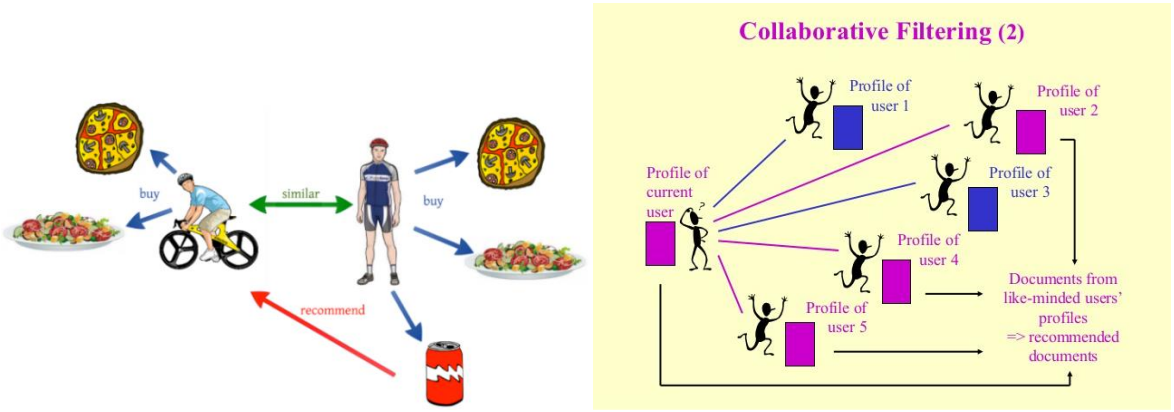


Fig20. Collaborative Filtering

There are two type of CF, which are User-based Collaborative Filtering (UserCF) and Item-based Collaborative Filtering (ItemCF). We choose ItemCF for our recommendation algorithm in consideration of their suitable applicable conditions and our specific need regarding the development and implement of RollerOutstander.

9.3.2 Why We Choose ItemCF

Our goal is to recommend rollercoasters for users. Obviously, rollercoasters are fixed structures that takes a long time to design and create. Therefore, the number of high-level

rollercoasters in the world must be a relatively stable number which only changes occasionally. Rollercoasters are huge constructions that one can serve numerous people over an extended length of time. After comparing to the features of UserCF and ItemCF that listed above, we decide to use ItemCF because it will theoretically **operate much more efficiently** due to the relatively small number of rollercoasters to compute about.

Other features of ItemCF also fit our need. The market of rollercoasters should be **highly customized** as most users want unexperienced, new adventures instead of boring, obsolescence repeats. ItemCF has the strong ability to bulldoze the long tail of popularity distribution, in other words, discover the proportion of rollercoasters that are not generally popular or known by most rollercoaster players, therefore provide them with more individualized and diverse recommendation.

For another feature, ItemCF is highly **timely** and can cold-start. It refines the recommendation after every behavior of the user, and can start generating recommendation since the first behavior is tracked. Typical users of RollerOutstander would not use it very frequently as rollercoaster-riding is a relatively occasional recreational activity. Therefore, it's important for us to be able to generate the recommendation with the least user behavior record.

In following sections 9.3.2 – 9.3.5 we'll provide detailed explanation of realization of ItemCF algorithm.

9.3.3 Preference Index

Table9.Local Notation

x_{uiz}	$u \in [0,+\infty)$	User u' s amount of behavior z related to item i
y_z	$y \in [0,+\infty)$	The weight of behavior z on user's preference

User u's preference of item i can be calculated by the product of amount of behavior and weight of behavior, then add all the behaviors:

$$r_{ui} = x_{ui} y^T = \sum_z x_{uiz} y_z$$

Similarity Index

Then we calculate the similarity w_{ij} applying equation below

This equation is edited from the original simple equation

$$w_{ij} = \frac{|N(i) \cap N(j)|}{|N(i)|}$$

We can punish the popularity of item j to ensure that periodical trends don't cause too much correlation between the popular item and all other items, thus obtain

$$w_{ij} = \frac{|N(i) \cap N(j)|}{\sqrt{|N(i)||N(j)|}}$$

9.3.4 Preference Prediction

Then we can obtain the prediction of user u's preference of item j by the scale product of correlation vector and preference vector:

$$p_{uj} = \sum_{i \in N(u) \cap S(j,k)} w_{ij} r_{ui}$$

Where function $S(j,k)$ is defined as the set of K items that are most similar to item j.

9.3.5 Improvements

According to conclusion of prior studies, we can increase the diversity of recommendation. This method eliminates the impact of differentiated average similarity between different categories of goods.

$$w'_{ij} = \frac{w_{ij}}{\max_j w_{ij}}$$

Another subject is the selection of constant K. By looking at prior work we can estimate an empirical value $K \cong 20$.

9.4 Tag-based Recommendation

9.4.1 Concepts of Tags and Our Rationale

Tags are a kind of keywords without hierarchical structure to describe information, which can be used to describe the semantics of objects. (Wikipedia) Tags are generally from two sources: one is from authors or experts; the other is to let ordinary users tag items, that is, UGC (User Generated Content) tagging applications. UGC's tag system is an important way to express user interest and goods semantics. When a user tags an item, the tag describes the user's interest on the one hand, on the other hand, it expresses the semantics of the item, thus connecting the user and the item.

Based on the given characteristics of tags, we find it suitable for recommendation of our roller-coaster recommendation. First, tag is a user-friendly method to let users leave their opinion and express their feeling of a roller coaster experience. Second, because the App is for single type of items, rollercoasters, the variety of tags are controllable and therefore probably converge to highlight characteristics of each rollercoaster after some accumulation of user-generated tags.

In following sections, we explain how the Tag-based recommendation algorithm works.

9.4.2 Simple Algorithm

Table10.Local Notation

b	$b \in Z^+$	Label ID
$B(u)$	B={b | lable b is used by user u}	
$C(i)$	C={b | lable b is used on item i}	
n_{ub}	$n \in Z$	Times label b used by user u
m_{bi}	$m \in Z$	Times label b used on item i
p_{ui}	$p \in [0, +\infty)$	Prediction of user u's preference of item i

To come up with a simple algorithm generating recommendation for users based on their behaviors regarding tags, we can start from considering only user's most frequently used tags, and most popular items labeled this tag.

$$p_{ui} = \sum_{b \in B(u) \cap C(i)} n_{ub} m_{bi}$$

This formula gives the quantitative prediction of user u's preference of item i.

9.4.3 Further Improvement

Table11.Local Notation

n^*_b	$n^* \in Z$	Number of users that have used label b
n^*_i	$n^* \in Z$	Number of users that have put label on item i

Punishing Hot Items

The prior formula tends to put excessive weight on popular labels and items, which affects the diversity and novelty of recommendation. To cope with this we punish the popularity of label b and item i by dividing the number of user associated with b and i. log function is used to treat user scale difference on exponential level.

$$p_{ui} = \sum_{b \in B(u) \cap C(i)} \frac{n_{ub}}{\log\left(1+n^*_b\right)} \frac{m_{bi}}{\log\left(1+n^*_i\right)}$$

Data Sparsity

New users or items have relatively limited number of labels and are consequently unlikely to be recommended by the algorithm above. To deal with the data sparsity we can add extra labels to the set $C(i)$ that are relevant, in order to provide more precise recommendation.

The core strategy is to calculate the similarity between labels and add the labels that are most similar to labels in $C(i)$ and then add them to the set.

We can define similarity as cosine of angle between two vectors representing the relationship between labels and items, which are

$$n_b = \begin{bmatrix} n_{b1} \\ n_{b2} \\ n_{b3} \\ \vdots \\ n_{bi} \end{bmatrix}, n'_b = \begin{bmatrix} n'_{b1} \\ n'_{b2} \\ n'_{b3} \\ \vdots \\ n'_{bi} \end{bmatrix} where\ i \in N(b) \cup N(b')$$

By applying Euclidean Point Product Formula we obtain

$$\vec{a} \cdot \vec{b} = \|a\| \|b\| \cos\theta$$

$$sim(b, b') = \cos\theta = \frac{n_b \cdot n'_b}{\|n_b\| \|n'_b\|}$$

$$sim(b,b') = \frac{\sum_{i \in N(b) \cap N(b')} n_{bi} n_{b'i}}{\sqrt{\sum_{i \in N(b)} n^2_{bi} \sum_{i \in N(b')} n^2_{b'i}}}$$

Tidying of Tags

Labels have their user-generated nature of randomness and inaccuracy. To maintain a high quality of label set for the items, labels have to be self-tidied according to several strategies.

First, synonyms should be combined. For example, "iron structure" and "iron made" are synonym comments left by users. A dictionary of synonyms can be used to combine synonyms. Simple differences such as different root in one word, or occurrence of separator (for example "recommender" and "recommended") can be detected and processed first using simpler algorithms.

Second, meaningless words, such as high-frequency stop word or conjunctions, should be deleted. This can be done using a preset dictionary of low-value words. **Third**, tags which are too subjective are not valuable when generating recommendations. **Forth**, some specific words, including abusive language or discriminative content, should be filtered out of the tag set.

10 User Experience

10.1 User Interface Prototype

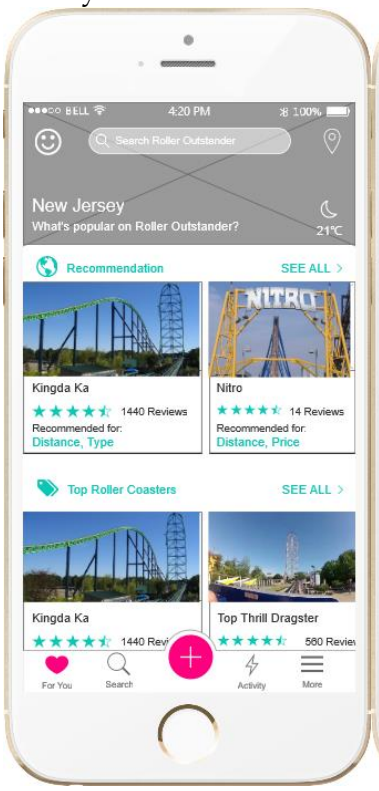
Homepage The home page is the most basic and important of this app it can quickly guide you to access to other pages and exploring the roller coaster you like. The element "recommendation" can directly show the user a list of rollercoasters that most fit to them. In most cases, the content of the recommendation will be varying from people to people, since the previous algorithm can according to users' preference and taste after they use the app for a period of time. At the same time, the "ranking" part below will show the contemporary best rollercoaster among all the app users. The filter downward can help the user to manually filter out the type of roller coaster that they required or want.

Preference In the setting page of preferences, users can change their information of basic preferences and "likes" manually according to what kind of roller coaster they really like. The reason why this page needs to be established is because that algorithm sometimes will begot inaccurate result in some extreme condition (the possibility is quite low). In those cases, users can change their own preference by using this page if they want to regulate the result of our algorithm. Moreover, this tool might often be using by people who want to recommend roller coast to their friends.

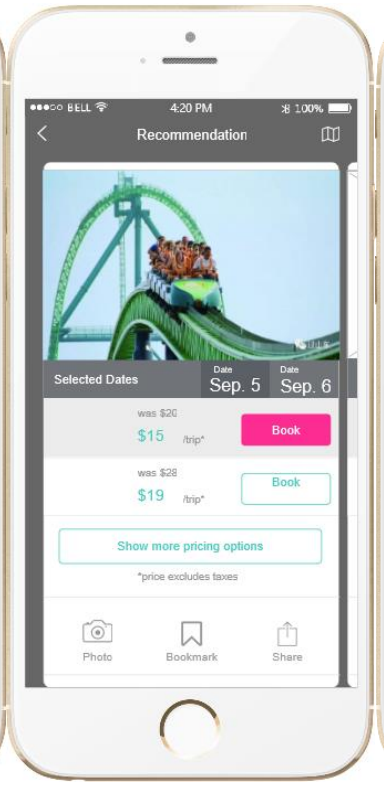
Search The option search can be found easily at the bottom of the app. It helps the users find the roller coasters that they might be interested in taking a ride. To achieve this goal, we include the "recommendation" button which links to the results calculated by recommendation system. Another button which includes the users' "favorite" roller coasters is placed next to "recommendation". They are both placed on the most obvious place of the interface which help the users use them easily.

A "filter" is also available in the searching process. It mainly includes the information that the user might want to limit the results. Among them are the geographical region, range of price, and public open time. In this way, the users can find the roller coasters that fit their expectation better while helping the system study the habits of the users.

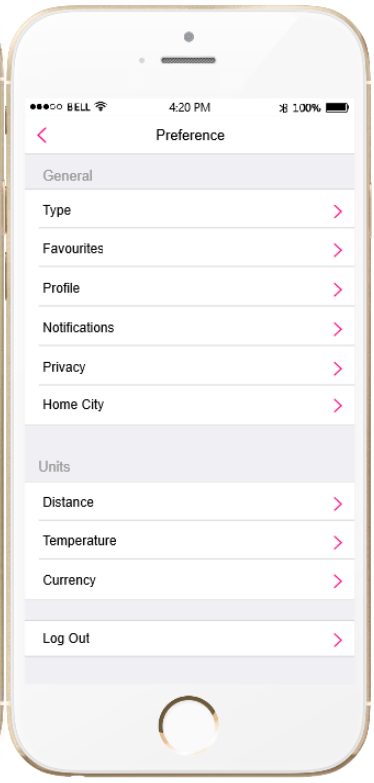
The historical research is included at the bottom of the page. This help the users find the information of the roller coasters that they have reviewed more quickly. The function is mainly for convenience.



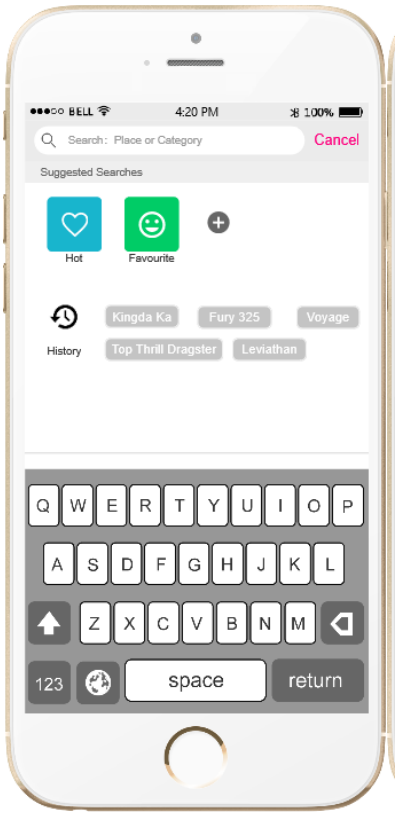
Homepage



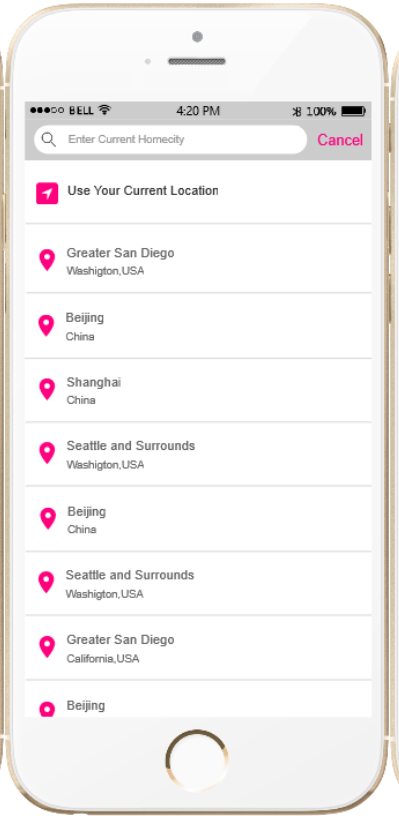
Preference



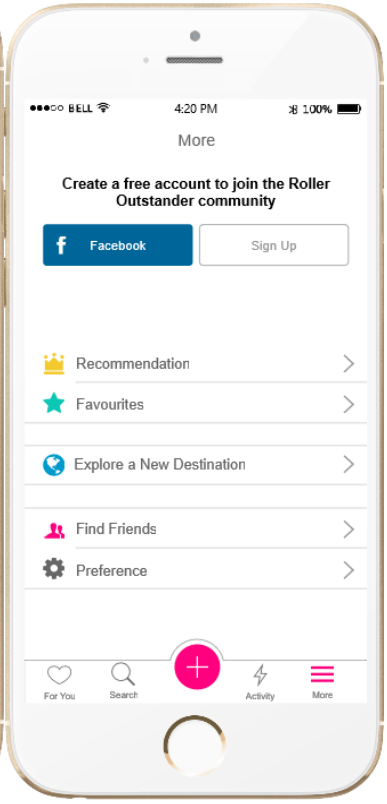
Search



The historical research



Location



More

Location The location of the users is required for a more advanced search; thus, the system will ask the users about their usual geographical location. The interface for the location is very concise, the only thing the users have to do is filling the blank with their home city. The information can be utilized to identify the similar users (as it is also one of the characteristics) and improve the recommendation system.

More The "more" interface is a combination of the previous interfaces (which help guide the users to enter the section they want easily without memorizing the exact locations for each), and the features that really build an online community of roller coasters. The users can choose to link their Facebook account to the app to share more information of roller coasters easily. They can also find friends on the app. This two options satisfy users' needs for online social interactions, and provide data for the recommendation system to analyze. Usually users will share similar characteristics with the friends they have. This help pinpoint the preference and taste from the view of the system, and improve the user experience.

10.2 Typical User Case

One of the users, Allen want to find a roller coaster which he might enjoy near to the destination of his trip to Valencia, California in advance. He can find the result and the information easily by clicking the "search "button and use the filter to find the roller coasters which are open during the time he enters Valencia. He doesn't set a budget for himself so he just ignores the option of the range of pricing. This is all the things Allen needs to do to find the answer. The system analyzes the data of Allen and his friends online, and identifies Allen as the user who enjoys the thrills of roller coasters. Based on the information above, the system would recommend Allen with the results including Superman: Escape from Krypton, one of the top- ranking roller coasters located in Valencia, California. The result also provides him with the basic information about the public open time, the reflections and comments of the users on RollerOutstander, and the pricing. The whole process would finish within a few minutes as the system is really efficient.

10.3 Evaluation and Reflection

The benefits of the recommendation system we used is that it contains the result of three different method to find what the user are interested in which make the result more reliable than just evaluating users' behavior and give recommendation from one perspective. However, the drawback of system will be the buffer period for the user means that the system need time to study user's preference. When there is a new user register for the application, it can only give the user what most people are interested in instead of providing customized recommendations.

10.3.1 Another disadvantages

One of the drawbacks is that the users sometimes make random choices which are not correlated to their preference. This is hard to be identified by the system, which can lead to the inaccurate prediction of the user's preference, resulting in the unsatisfied recommendation to the users and have a negative effect. The process of making the adjustments in the identification of the users' taste requires large amount of calculations.

10.3.2 Reflection

The quality of the data is essential in the prediction of the users. Thus, the evaluation model mentioned in 8.2.5 might need adjustment to ensure the quality of data the system utilizes.

11 Advantages and Disadvantages

We reflected upon our modeling process and discover certain strengths and limitations

Strength:

- We build a model which is objective enough based on entropy method. We reduce the dimension brought by the multiple indicators.
- We test the sensitivity of our evaluation model to ensure its stability.
- We design abundant interfaces for our app and match it with more comprehensive evaluation, filtering, and recommendation mechanism. Users would have a good experience in using it.

Possible improvements:

- If the data is more complete, the results of our ranking might be more accurate.
- The time for completion is limited, and our app is not able to realize all the functions that we designed. If there is sufficient time, it would definitely be an excellent recommendation app for roller coasters.

References

1. Sean M. McNee, John Riedl, and Joseph A. Konstan. 2006. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In CHI '06 Extended Abstracts on Human Factors in Computing Systems (CHI EA '06). ACM, New York, NY, USA, 1097-1101. DOI=http://dx.doi.org/10.1145/1125451.1125659

2. "The Best Roller Coasters in the World." *ranker*, www.ranker.com/crowdranked-list/ best-roller-coasters.

3. "In-depth recommendation engine related algorithms - collaborative filtering." IBM, 21 3 2011,www.ibm.com/developerworks/cn/web/ 1103_zhaoct_recommstudy2/index.html?ca=drs-. Karypis, George.

4. "Evaluation of Item- based Top- N Recommendation Algorithms." University of Minnesota

5. "List of roller coaster rankings." Wikiwand, www.wikiwand.com/en/

List_of_roller_coaster_rankings.

Record Holders." Roller Coaster DataBase, rcdb.com/rhr.htm?m=3.

6. Wikipedia. en.wikipedia.org/wiki/Millennium_Force.

7. Xiang, Liang. Recommended System Practice (Turing Original 5). People Post Press.

Tools and Software

Python, free distribution.
Paper written and generated via Microsoft© Office™ 365 Word, institution certification.
Graph generated via Microsoft© Office™ 365 Excel, institution certification and Python.
Xmind Zen, free trial license.
Calculation using Python.

Appendix

Appendix Contains:

Processed Raw Data

Our ranking and online rankings

Data dimensionality reduction based on PCA Analysis

Table1

The Data for well-recognized Top Roller Coaster

Name	Kingda Ka	Millennium Force
City/State/Region	New Jersey	Ohio
Country/Region	United States	United States
Geographic Region	North America	North America
Construction	Steel	Steel
Type	Sit Down	Sit Down
Status	Operating	Operating
Year/Date Opened	2005	2000
Height (feet)	456.0	310.0

Length (feet)	3118.0	6595.0
Inversions	NO	NO
G Force	3.000197052	5.140847449
Vertical Angle (degrees)	90	80

Table2

Online Ranking 1		Our Ranking	
Name	**Country/Region**	**Name**	**Country/Region**
Superman - Ride Of Steel	United States	Kingda Ka	United States
Beast	United States	Top Thrill Dragster	United States
Oblivion	United Kingdom	Superman: Escape from Krypton	United States
Griffon	United States	Steel Dragon 2000	Japan
Goliath	United States	Red Force	Spain
Top Thrill Dragster	United States	Tower of Terror II	Australia
Millennium Force	United States	Millennium Force	United States
Superman: Escape from Krypton	United States	Formula Rossa	United Arab Emirates
Ride of Steel	United States	Fury 325	United States
		Leviathan	Canada

Online Ranking 2	
Name	**Country/Region**
Steel Vengeance	United States
Maverick	United States
El Toro	Germany
Voyage	United States
Top Thrill Dragster	United States
Millennium Force	United States
Kingda Ka	United States
Apollo's Chariot	United States

Table3

Name	Year/Date Opened	Height (feet)	Length (feet)	Number of Inversions	G Force	Vertical Angle (degrees)
Superman - Ride Of Steel	2000	197.0	5350.0	0	3.250908033	68
Beast	1979	110.0	7359.0	0	3.873252967	45
Oblivion	1998	65.0	1222.0	0	4.5	87
Griffon	2007	205.0	3108.0	2	4	90
Goliath	2012	191.6	1204.0	3	4.5	90
Ride of Steel	1999	208.0	5400.0	0	3.186758316	68
Steel Vengeance	2018	205.0	5740.0	4	3.28727346	90
Maverick	2007	105.0	4450.0	2	4.88133368	95
El Toro	2009	80.5	2379.0	0	3.461245	76.11497
Voyage	2006	159.0	6442.0	0	3.975952321	66
Apollo's Chariot	1999	170.0	4882.0	0	4.1	65

Table4

Table 4 Levels of Correlation Degree

levels	Extremely lost	Serious disconnection	Moderate loss of association	On the verge of losing	Primary association	Intermediate association	Intermediate association
Correlation Degree	(0，0.3)	(0.3，0.5)	(0.5，0.7)	(0.7，0.8)	(0.8，0.85)	(0.85，0.9)	(0.9，1.0)

Data dimensionality reduction based on PCA Analysis

● Standardization and Normalization of Raw Data

Assume there are m indexes and n objects to be evaluated. The value of index i attained by object j is expressed as a_{ij}. We convert the index value a_{ij} into standardized value a_{ij}, thus we obtain

$$a_{ij} = \frac{a_{ij} - \mu_j}{s_j}, i = 1, 2, \cdots, n; j = 1, 2, \cdots, m$$

Where $\mu_j = \frac{1}{n}\sum_{i=1}^{n} a_{ij}$； $s_j = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(a_{ij} - \mu_j\right)^2}$, $j = 1, 2, \cdots, m$， μ_j is the mean of index j and s_j is the standard deviation of index j.

$$x_j = \frac{x_j - \mu_j}{s_j}, j = 1, 2, \cdots, m$$

● Calculating Correlation Coefficient Matrix R

For the Correlation Coefficient Matrix $R = \left(r_{ij}\right)_{m \times m}$,

$$r_{ij} = \frac{\sum_{k=1}^{n} a_{ki} \cdot a_{kj}}{n-1}, i, j = 1, 2, \cdots, m$$

Where: $r_{ii} = 1$， $r_{ij} = r_{ji}$， r_{ij} is the correlation index between index i and j.

● Calculating Eigenvalue and Eigenvector

From the correlation coefficient matrix R we calculate its eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m \geq 0$ and corresponding eigenvector $u_1, u_2, \cdots, u_m$, where $u_j = \left[u_{1j}, u_{2j}, \cdots, u_{mj}\right]^T$. Eigenvectors composite m index variables:

$$y_1 = u_{11}x_1 + u_{21}x_2 + \cdots + u_{m1}x_m,$$
$$y_1 = u_{12}x_1 + u_{22}x_2 + \cdots + u_{m2}x_m,$$
$$\vdots$$
$$y_m = u_{1m}x_1 + u_{2m}x_2 + \cdots + u_{mm}x_m$$

Where: y_1 is first PCA, y_1 is the second PC, $\cdots$; y_m is the m^{th} PC.

- **Determining p PCs and perform statistical analysis**

PCA converts numerous indices into several weakly correlated comprehensive indices. PCs and corresponding accumulative contribution rate are given below.

Table 5 PCs and Corresponding Contribution Rates

PC	Contribution Rate (%)	Accumulative (%)
1	35.617589	35.617589
2	31.459494	67.077083
3	12.448942	79.526025
4	11.050469	90.576494
5	4.946335	95.522829
6	4.477171	100

Under accumulative variance of 90.576% we obtain 4 PCs. These 4 PCs provide 90.576% information of the indices, which satisfies the principle of PCA. Hence we've realized the dimensionality reduction of indices and simplified further calculation followed

Codes

BP Neural Network

```
import math
import random
import datetime
import pyexcel_xls as px
import numpy as np
import matplotlib.pyplot as plt
from collections import OrderedDict


random.seed(0)


def rand(a, b):
    return (b - a) * random.random() + a
```

```
def make_matrix(m, n, fill=0.0):
    mat = []
    for i in range(m):
        mat.append([fill] * n)
    return mat


def sigmoid(x):
    try:
        return 1.0 / (1.0 + math.exp(-x))
    except OverflowError:
        return 0.0


def sigmoid_derivative(x):
    return x * (1 - x)


def normalization(x, Max, Min):
    x = (x - Min) / (Max - Min)
    return x


class BPNeuralNetwork:
    def __init__(self):
        self.input_n = 0
        self.hidden_n = 0
        self.output_n = 0
        self.input_cells = []
        self.hidden_cells = []
        self.output_cells = []
        self.input_weights = []
        self.output_weights = []
        self.input_correction = []
        self.output_correction = []

    def errorpolyline(self, limit, errhistory):
        errhistory10 = np.log10(errhistory)
        minerr = min(errhistory10)
        plt.plot(errhistory10)
```

```
        # plt.plot(range(0,limit+1000,1000),[minerr]*len(range(0,limit+1000,1000)))

        ax=plt.gca()
        ax.set_yticks([-2,-1,0,1,2,minerr])
        ax.set_yticklabels([u'$10^{-2}$',u'$10^{-1}$',u'$1$',u'$10^{1}$',u'$10^{2}$',str(('%.4f'%np.power(10,min-
err)))])
        ax.set_xlabel('Iteration')
        ax.set_ylabel('Error')
        ax.set_title('Error History')
        plt.savefig('errorhistory.png',dpi=700)
        plt.close()

    def setup(self, ni, nh, no):
        self.input_n = ni + 1
        self.hidden_n = nh
        self.output_n = no
        # init cells
        self.input_cells = [1.0] * self.input_n
        self.hidden_cells = [1.0] * self.hidden_n
        self.output_cells = [1.0] * self.output_n
        # init weights
        self.input_weights = make_matrix(self.input_n, self.hidden_n)
        self.output_weights = make_matrix(self.hidden_n, self.output_n)
        # random activate
        for i in range(self.input_n):
            for h in range(self.hidden_n):
                self.input_weights[i][h] = rand(-0.2, 0.2)
        for h in range(self.hidden_n):
            for o in range(self.output_n):
                self.output_weights[h][o] = rand(-2.0, 2.0)
        # init correction matrix
        self.input_correction = make_matrix(self.input_n, self.hidden_n)
        self.output_correction = make_matrix(self.hidden_n, self.output_n)

    def predict(self, inputs):
        # activate input layer
        for i in range(self.input_n - 1):
            self.input_cells[i] = inputs[i]
        # activate hidden layer
        for j in range(self.hidden_n):
```

```
            total = 0.0
            for i in range(self.input_n):
                total += self.input_cells[i] * self.input_weights[i][j]
            self.hidden_cells[j] = sigmoid(total)
        # activate output layer
        for k in range(self.output_n):
            total = 0.0
            for j in range(self.hidden_n):
                total += self.hidden_cells[j] * self.output_weights[j][k]
            self.output_cells[k] = sigmoid(total)
        return self.output_cells[:]

    def back_propagate(self, case, label, learn, correct):
        # feed forward
        self.predict(case)
        # get output layer error
        output_deltas = [0.0] * self.output_n
        for o in range(self.output_n):
            error = label[o] - self.output_cells[o]
            output_deltas[o] = sigmoid_derivative(self.output_cells[o]) * error
        # get hidden layer error
        hidden_deltas = [0.0] * self.hidden_n
        for h in range(self.hidden_n):
            error = 0.0
            for o in range(self.output_n):
                error += output_deltas[o] * self.output_weights[h][o]
            hidden_deltas[h] = sigmoid_derivative(self.hidden_cells[h]) * error
        # update output weights
        for h in range(self.hidden_n):
            for o in range(self.output_n):
                change = output_deltas[o] * self.hidden_cells[h]
                self.output_weights[h][o] += learn * change + correct * self.output_correction[h][o]
                self.output_correction[h][o] = change
        # update input weights
        for i in range(self.input_n):
            for h in range(self.hidden_n):
                change = hidden_deltas[h] * self.input_cells[i]
                self.input_weights[i][h] += learn * change + correct * self.input_correction[i][h]
                self.input_correction[i][h] = change
        # get global error
```

```
        error = 0.0
        for o in range(len(label)):
            error += 0.5 * (label[o] - self.output_cells[o]) ** 2
        return error

    def train(self, cases, labels, limit=100000, learn=0.05, correct=0.1):
        errhistory = []
        for j in range(limit):
            error = 0.0
            for i in range(len(cases)):
                label = labels[i]
                case = cases[i]
                error += self.back_propagate(case, label, learn, correct)
            print(j,":",error)
            errhistory.append(error)
        self.errorpolyline(limit,errhistory)

    def test(self):
        xls_data = px.get_data(r"./wood.xlsx")
        # xls_data = px.get_data(r"./steel.xlsx")
        origin_data = []
        train_cases = []
        train_labels = []
        check_cases = []
        res_start = 16
        res_end = 17
        year_max = float("-inf")
        year_min = float("inf")
        height_max = float("-inf")
        height_min = float("inf")
        speed_max = float("-inf")
        speed_min = float("inf")
        length_max = float("-inf")
        length_min = float("inf")
        inversion_max = float("-inf")
        inversion_min = float("inf")
        res_max = float("-inf")
        res_min = float("inf")

        for sheet in xls_data:
```

```
        origin_data = xls_data[sheet]
        break

    for line in origin_data:
        if line[9] > year_max:
            year_max = line[9]
        if line[9] < year_min:
            year_min = line[9]
        if line[10] > height_max:
            height_max = line[10]
        if line[10] < height_min:
            height_min = line[10]
        if line[11] > speed_max:
            speed_max = line[11]
        if line[11] < speed_min:
            speed_min = line[11]
        if line[12] > length_max:
            length_max = line[12]
        if line[12] < length_min:
            length_min = line[12]
        if line[14] > inversion_max:
            inversion_max = line[14]
        if line[14] < inversion_min:
            inversion_min = line[14]
        if(line[res_start:res_end]):
            if(not isinstance(line[res_start],str)):
                if line[res_start] > res_max:
                    res_max = line[res_start]
                if line[res_start] < res_min:
                    res_min = line[res_start]

    for line in origin_data:
        tmp = []
        tmp.append(normalization(line[9],year_max,year_min))
        tmp.append(normalization(line[10],height_max,height_min))
        tmp.append(normalization(line[11],speed_max,speed_min))
        tmp.append(normalization(line[12],length_max,length_min))
        tmp.append(normalization(line[14],inversion_max,inversion_min))
        if(line[res_start]==0):
            check_cases.append(tmp)
```

```
                elif(line[res_start:res_end]):
                    if(isinstance(line[res_start],str)):
                        check_cases.append(tmp)
                    else:
                        train_cases.append(tmp)
                        train_labels.append([normalization(line[res_start],res_max,res_min)])
                else:
                    check_cases.append(tmp)

        self.setup(5, 10, 1)
        self.train(train_cases, train_labels, 1000, 0.05, 0.1)

        prediect_res = []
        real_res = []
        for case in check_cases:
            tmp = self.predict(case)
            prediect_res.append(tmp[0]*(res_max-res_min)+res_min)
            print(tmp[0]*(res_max-res_min)+res_min)
        # print("#########################")
        # for item in train_labels:
        #     real_res.append(item[0]*(res_max-res_min)+res_min)
        #     print(item[0]*(res_max-res_min)+res_min)

        # minres = min(real_res)
        # maxres = max(real_res)
        # plt.plot(prediect_res,c="blue",marker="o")
        # plt.plot(real_res,c="red",marker="v")

        # ax=plt.gca()
        # ax.set_yticks([maxres*1.5,minres*0.5])
        # ax.set_xlabel('Roller Coater')
        # ax.set_ylabel('Feet')
        # ax.set_title('Drop Feet Simulation')
        # plt.savefig('predictresualt.png',dpi=700)
        # plt.close()

if __name__ == '__main__':
    nn = BPNeuralNetwork()
    nn.test()
```

Grey Correlation Analysis

```
import pandas as pd
import pyexcel_xls as px
import seaborn as sns
import matplotlib.pyplot as plt
from numpy import *

data = px.get_data(r"./data.xlsx")
df = pd.DataFrame(data['all'])

# drop useless data column
for i in range(9):
    df = df.drop(i,1)
df = df.drop(13,1)

def GRA_ONE(DataFrame,m=0):
    gray= DataFrame
    #read as dataframe format
    gray=(gray - gray.min()) / (gray.max() - gray.min())
    #normalize
    std=gray.iloc[:,m]# standard key element
    ce=gray.iloc[:,0:]# compare key element
    n=ce.shape[0]
    m=ce.shape[1]

    #compare with standard key element
    a=zeros([m,n])
    for i in range(m):
        for j in range(n):
            a[i,j]=abs(ce.iloc[j,i]-std[j])

    #get the max and min data in martix
    c=amax(a)
    d=amin(a)

    #calculate value
    result=zeros([m,n])
    for i in range(m):
        for j in range(n):
            result[i,j]=(d+0.5*c)/(a[i,j]+0.5*c)
```

```
    #calculate mean and gray correlation
    result2=zeros(m)
    for i in range(m):
            result2[i]=mean(result[i,:])
    RT=pd.DataFrame(result2)
    return RT

def GRA(DataFrame):
    list_columns = [str(s) for s in range(len(DataFrame.columns)) if s not in [None]]
    df_local = pd.DataFrame(columns=list_columns)
    for i in range(len(DataFrame.columns)):
        df_local.iloc[:,i] = GRA_ONE(DataFrame,m=i)[0]
    return df_local

def ShowGRAHeatMap(DataFrame):
    labels = ['Year','Height','Speed','Length','Inversions','Drop','Duration','G Force','Vertical Angle']
    colormap = plt.cm.get_cmap('RdBu_r')
    f, ax = plt.subplots(figsize = (14, 12))
    # plt.figure(figsize=(14,12))
    sns.heatmap(DataFrame.astype(float),ax=ax, linewidths=0.1,vmax=1.0, square=True, cmap=colormap, linecolor='white', annot=True, xticklabels=True)
    ax.set_title('Correlation of Features', y=1.05, size=15)
    ax.set_xticklabels(labels,rotation=360)
    ax.set_yticklabels(labels,rotation=0)
    f.savefig('correlation_of_features.png', bbox_inches='tight', dpi=700)

df.columns = ['Year','Height','Speed','Length','Inversions','Drop','Duration','G Force','Vertical Angle']
# calculate
data_wine_gra = GRA(df)
# show resualts
ShowGRAHeatMap(data_wine_gra)
```

Principal Component Analysis& AHP

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pyexcel_xls as px
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
```

```
data = px.get_data(r"./data.xlsx")
df = pd.DataFrame(data['all'])

# drop useless data column
df = df.drop(0,1)
df = df.drop(1,1)
df = df.drop(2,1)
df = df.drop(3,1)
df = df.drop(4,1)
df = df.drop(5,1)
df = df.drop(6,1)
df = df.drop(7,1)
df = df.drop(8,1)
# df = df.drop(11,1)
df = df.drop(13,1)
# df = df.drop(15,1)
# df = df.drop(16,1)
df = df.drop(19,1)
df = df.drop(20,1)

origin_X = df.values
mm = MinMaxScaler()
X = mm.fit_transform(origin_X)
pca = PCA(n_components=6, whiten=False, copy=True)
pca.fit(X)
res = pca.transform(X)

print(pca.explained_variance_ratio_)
# print(pd.DataFrame(res))

li = np.array(X) # convert to martix
li=(li-li.min())/(li.max()-li.min())      #minmax normalize
m, n = li.shape
k = 1 / np.log(m)
yij = li.sum(axis=0)
pij = li / yij
test = pij * np.log(pij)
test = np.nan_to_num(test)
ej = -k * (test.sum(axis=0))
```

```
wi = (1 - ej) / np.sum(1 - ej)
print(wi)
tmp = wi[1]
wi[1] = wi[4]
wi[4] = tmp
tmp = wi[2]
wi[2] = wi[8]
wi[8] = tmp
print(wi)

for item in X:
    print(sum(wi*item))
```

Wood&Steel 3D- Comparation

```
import pandas as pd
import pyexcel_xls as px
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from numpy import *

data = px.get_data(r"./data.xlsx")
wood_data = pd.DataFrame(data['wood'])
steel_data = pd.DataFrame(data['steel'])

for i in range(10):
    wood_data = wood_data.drop(i,1)
    steel_data = steel_data.drop(i,1)
for i in range(13,19):
    wood_data = wood_data.drop(i,1)
    steel_data = steel_data.drop(i,1)

wood_data.columns = ['height', 'speed', 'length']
steel_data.columns = ['height', 'speed', 'length']
wood_data = wood_data.values
steel_data = steel_data.values
print(wood_data)
print(steel_data)

ax = plt.subplot(111, projection='3d')
ax.set_title('Wood And Steel Base Parameter')
ax.scatter(wood_data[:,0],wood_data[:,1],wood_data[:,2],c='red',label='wood')
```

```
ax.scatter(steel_data[:,0],steel_data[:,1],steel_data[:,2],c='blue',label='steel')
ax.set_xlabel("height")
ax.set_ylabel("speed")
ax.set_zlabel("length")
plt.legend()
plt.show()
```