Capstone 1: Recommender System - Data Wrangling

The goal of this project is to predict the number of attempts a user will require to solve a practice problem that they've never seen before. I want to start with a logistic regression model as my baseline and then compare the performance to a collaborative filtering model of increasing complexity.

Collaborative filtering models that employ matrix factorization techniques are widely popular due to their scalability and ability to work on highly **sparse** data sets. For the collaborative filtering model, we only need the user_id and problem_id along with the problem attempt history between users and problems. This type of model needs no other information.

The regression model on the other hand will utilize features describing users and problems. Regression models require **dense** matrices as input. I will therefore focus my data munging on feature generation and imputing missing data for the regression model.

This project contains 3 datasets. The first, *train_submissions*, contains the 3 columns shown below. This training data does not contain any missing values and did not require any data cleaning.

	user_id	problem_id	attempts_range
0	user_232	prob_6507	1
1	user_3568	prob_2994	3
2	user_1600	prob_5071	1
3	user_2256	prob_703	1
4	user_2321	prob_356	1

The 2nd dataset is the user_data which contains various features describing each user. The feature are described below. The only feature with missing values was the country column. There were 1,153 missing country values out of 3571 (~32%). I decided to fill all missing country values with the string 'None'. This will allow me to retain the data for the regression model rather than discarding 30% of the user data. Categorical data were converted to dummy variables so that the entire dataset is numeric. This is required for the regression model.

1. **user_data.csv** - This is the file containing data of users. It contains the following features :-
    1. user_id - unique ID assigned to each user
    2. submission_count - total number of user submissions
    3. problem_solved - total number of accepted user submissions
    4. contribution - user contribution to the judge
    5. country - location of user
    6. follower_count - amount of users who have this user in followers
    7. last_online_time_seconds - time when user was last seen online
    8. max_rating - maximum rating of user
    9. rating - rating of user
    10. rank - can be one of 'beginner' ,'intermediate' , 'advanced', 'expert'
    11. registration_time_seconds - time when user was registered

The 3rd data set contains 3 feature columns describing each problem. This dataset had by far the most missing values, with 60% of the points column being null. Luckily, I found a pattern between the points and level_type (which describes the difficulty of the problem) columns. This allowed me to impute one column from the other.

1. **problem_data.csv** - This is the file containing data of the problems. It contains the following features
    1. problem_id - unique ID assigned to each problem
    2. level_id - the difficulty level of the problem between 'A' to 'N'
    3. points - amount of points for the problem
    4. tags - problem tag(s) like greedy, graphs, DFS etc.

level_type points	A	B	C	D	E	F	G	H
-1.0	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN
3.0	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5.0	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN
8.0	NaN	NaN	1.0	NaN	NaN	NaN	NaN	NaN
10.0	1.0	1.0	NaN	NaN	NaN	NaN	NaN	NaN
250.0	10.0	1.0	1.0	NaN	NaN	NaN	NaN	NaN
500.0	521.0	33.0	1.0	1.0	2.0	1.0	1.0	NaN
750.0	9.0	21.0	2.0	NaN	1.0	NaN	NaN	NaN
1000.0	14.0	483.0	37.0	4.0	2.0	1.0	NaN	NaN
1250.0	NaN	15.0	22.0	2.0	NaN	NaN	NaN	NaN
1500.0	1.0	19.0	375.0	32.0	3.0	1.0	1.0	1.0
1750.0	1.0	1.0	24.0	22.0	2.0	NaN	NaN	NaN
2000.0	NaN	1.0	27.0	319.0	24.0	1.0	NaN	NaN
2250.0	NaN	NaN	NaN	23.0	24.0	2.0	NaN	NaN
2500.0	NaN	1.0	6.0	24.0	293.0	20.0	3.0	NaN
2750.0	NaN	NaN	NaN	NaN	12.0	9.0	1.0	NaN
3000.0	2.0	3.0	4.0	14.0	50.0	29.0	5.0	NaN
3250.0	NaN	NaN	NaN	NaN	NaN	3.0	2.0	NaN
3500.0	NaN	NaN	NaN	NaN	NaN	2.0	10.0	2.0
5000.0	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN

The table below shows how the majority of the data follows a pattern of 500 points for level_type A, and increasing by 500 point increments with each level_type. After filling in the level_type and points columns using this mapping, I was able to reduce the number of missing point values from 60% to just 1.5%!

This is what the mapping looked like:

```
point_level_mapping = {
    (0, 500): 'A',
    (501, 1000): 'B',
    (1001, 1500): 'C',
    (1501, 2000): 'D',
    (2001, 2500): 'E',
    (2501, 3000): 'F',
    (3001, 3500): 'G',
    (3501, 4000): 'H',
    (4001, 4500): 'I',
    (4501, 5000): 'J',
    (5001, 5500): 'K',
    (5501, 6000): 'L',
    (6001, 6500): 'M',
    (6501, 7000): 'N'}
```

Problems with points between the range of the tuples were assigned the corresponding level_type.

Finally, I also performed some feature engineering based on problem and user statistics. To start, I converted the tags column from lists of strings describing each problem, to a dense word count matrix. For each problem, I calculated the median number of attempts users took to solve the problem along

with other stats such as various quartiles and IQR. I did the same with the users and added these to their respective data tables. The final tables looks something like this (not all columns shown):

User Features

user_id_number	submission_count	problem_solved	contribution	country	user_attempts_median	user_attempts_min
1	84	73	10	13	1.0	1.0
2	688	599	-8	1	2.0	1.0
3	223	166	29	4	1.0	1.0
4	129	87	0	39	1.0	1.0
5	622	528	0	37	1.0	1.0

Problem Features

problem_id_number	level_type	points	problem_attempts_median	algorithms	and	binary	...	string
1	A	500.0	1.500000	0	0	0	...	0
2	G	3500.0	1.000000	0	0	0	...	0
3	B	1000.0	2.000000	0	0	1	...	0
4	D	2000.0	2.000000	0	0	0	...	0

Summary

1. Filled all missing values in “country” feature with the string “None”.
2. Used correlation between level_type and points to impute missing values in both columns, reducing all missing data in these features from 60% to 1.5%.
3. Converted categorical features to dummy variables so the data is compatible with regression models.
4. Converted “tags” feature from a list of strings to word count vectors for each problem.
5. Added features based on user and problem statistics.