Report

October:

We mainly did:

1) Deleted the bundles with size==1 or size > 95 quantile or size < 5 quantile. Obtained succint version and stored in 'succint_bundle_data.json'.

   The succint version have 23353 entries. max bundle size: 13
   min bundle size: 2
   mean bundle size: 6.6
   sd: 2.4
   max likes: 107
   min likes: 1
   mean likes: 4.2
   sd:6.1

2) Evaluated the different ways to extract the nouns from title.
3) Created an item set which over around 78%*(result.txt)* of the items in the titles show up in this item set*(itemSet3.txt)*.
4) Found the frequency for the items in the title*(freq.txt)*.
5) Found unidentified items and counted their frequency*(unidentified_title.txt, unidentified_word_freq.txt)*.

Details see: file:///Users/yingchenliu/Library/Containers/com.apple.Safari/Data/Downloads/Bundle_data%20Heather.html. Code file: *tmp.py, word_frequency.py*

November

We mainly did:

1) Updated the item set, removed some items that did not containd in title
2) Figured out a way to deal with the different forms of nouns(eg, +s, +es...)
3) We extracted features based on the show up of the items in a bundle, and split into training and testing set
4) A regression neural network is build and trained with the features extraxted, and using statsitc way to evaluate the training results
5) We also evaluate the data from the aspect of business by doing the marketing lift analysis

2nd

1.Test the correlation between the item.

Code file: *correlation.py, correlation_list*

9th

1. Thinking about how to transfer 'shirts' to 'shirt'

```
change => if (title_word == item) : do something
to     => if (title_word == item or title_word == (item+'s') or title_word == (item+'es')): do somethin
```

2. Get the features

```
based on 1, we can have a 2d array [[], [], ... , []]
the outer array will be the features for all the bundles, while the inner array (every element of the ar
The inner array will be at the length same as the items in your item list. If an item shows up in the ti
```

(code file: get_feature.py)

2a. Store the features

```
Getting the features can be time-consuming.
So we can store the features locally, in the form of numpy array, by simply call function numpy.save()

(code file: feature_itemSet4.npy)
```

3. Seperate the testing data and training data

We realized that some items in *itemSet4.txt* have never appeared in the bundel, so we removed all the rows of 0 value (which means these items have never appeared) from *feature_itemSet4.npy* and got *feature_itemSet5.npy*. We then deleted these items from *itemSet4.txt* and got *itemSet5.txt*.

```
After these, we get a super large array with shape x*y, x is the number of bundles, while y is the leng
Tried some simple ways to seperate the testing data:
1) use random number generator functions
2) more easily, choose one bundle for every five bundle(Final choice)
```

We obtained 20057 (~80%) training data and 5000(~20%) testing data. (code file: feature_processing.py, training.npy, testing.npy)

4. Choosing the proper model/deep-learning methods to handle the data

```
This is the unsupervised learning, because no labels will be provided.
Tried some basic types of methods.
```

We used K-means and skitlearn. We didn't test the model because the testing criterion are unknown. (code file: model.py)

4a. Set up proper standards to evaluate the model

```
how are we going to evalute the model and make use of the testing data?
what are the formal ways to do this in this field?
what is the ultimate purpose of this project?
....
```

5. Want some more improvements

```
1) optimize the model (try different parameters, adjusting the parameters based on performance...)
2) try new models
3) gain more data (find another similar dataset maybe?)
4) optimize the features (for now, we only obtain features from title, we can get more info from descri
5) ...
```

19th

1.According to last weeks feedback, we acquired again the training and testing data. We also revised an error regarding the item 'watch'.

2.Trained a regression network and adjusted the parameter to improve performance. Including:

increase dropout
shuffle data at the end of each epoch
adjust the number of unit of each layer
change the optimizer to sgd
adjust epoch and batch size

The training process and final result are stored in *record.txt*.
The testing date gives a mean_absolute_error of 3.6. *code file: model.py*

3.We also normalized data and trained a model stored in *model_with_normalizing.py* The training process and final result are stored in *record_with_normalizing.txt*

code file: x_test.npy, x_train.npy, y_test.npy, y_train.npy

Appendix: (Testing data)

dropout: 0.2 0.3 none
unit: 64, 128, 256
num of hidden layers: 1, 2
optimizer: sgd, adam
epoch: 100, 150

29th

1. We visualized the mean_absolute_error by comparing the predicted value with the real value. *(predict_and_real.csv).*
   Then we compute the Pearson's correlation and 2-tailed p-value: ('0.0968342585456855', '6.807742799752774e-12'). This result means our predictions are failed. They have no linear correlation with the real date.

code file: pearsons_correlation.py 2. We test the market lift and confidence in between every 2 items. The lift scores are stored in *lift_scores.csv*.

December

We mainly did:

1. Select star product

Purpose: whether there existing some items which will lift the likes of bundles

result.csv countains:
likes of bundles
title, price and outfit_count of products

Code file: tmp.py

We failed to detect relations between these features.

January

We mainly did:

This week, we built the different regression architecture to evaluate the correlation between the bundle information and the 'likes' received for a bundle.
Data: We have an array at shape (1, 253) to store the type of the items in the bundle, e.g., watch, shoes... We have an array at shape (1, 10) to store the color information in the bundle. We selectde 10 basic color: 'brown', 'black', 'blue', 'red', 'grey', 'white', 'gold', 'rose', 'pink', 'silver', if one of the color appears in the bundle, the index of this color is 1, otherwise is 0. We calculated the mean, sum, std infor for the price of every item and store in a (1, 3) array. We calculated the mean, sum, std infor for the likes of every item in the bundle and store in a (1, 3) array. We calculated the mean, sum, std infor for the outfit count of every item and store in a (1, 3) array. As a result, for every bundle, the training feature shape is (1, 253+10+3+3+3) = (1, 272) The training label is the likes for this bundle, at the shape (1, 1).

Model:

random forest: Because of the large size of the feature (272 numbers), we decided to use random forest as our first system.
xgboost: Because xgboost can do feature selection, it is suitable for this task.

svm: svm is a traditional machine learning model widely used in regression
FNN: feedforward neural network is a basic nn structure widely used in regression.

Optimizing: For the first three models, we used the gridsearch to select the optimized parameter in a wide range (see the code for specific range). For FNN, training failed (the loss and acc shows NaN). The reason would be the feature needs to be normalized due to the wide range of price, likes, etc appeared in the dataset.

Result: The result is recorded in optimizing_result.txt. xgboost has the best performance in the testing data and obtained a pearson's correlation at about 0.18