## Overview

The dataset in this report is from 'Rent the runway', a clothing rental company that provides female customers with various style of clothes for different occasions.

The dataset has fields that can be divided into three parts.

- Demographics/body shape data
  Age, bust size, body type, height, weight
- Rental information
  Item id, user id, size, category, rented for are recorded
- Self-reported fit feedback
  Fit, review summary, review text, review date, rating

With this dataset, we will perform an exploratory analysis to identify interesting phenomenon. Then, we will decide on a prediction task that can be done based on the information we have, following a model we set up to perform the prediction. Then we include a literature review of related and similar work done by researchers and professionals. Finally, we will summarize our report and state our conclusion.

## Explore the dataset

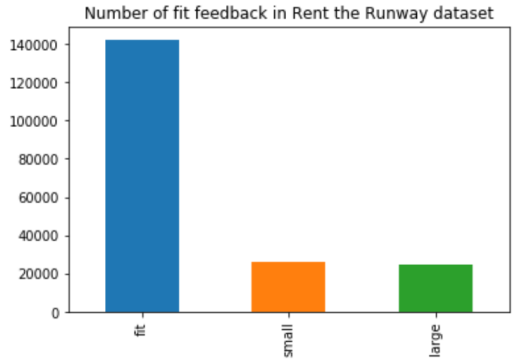
The dataset has 192544 transaction records and 15 columns. The total number of the unique customer is 105571. The distribution for the number of transactions each customer is displayed in the table below:

| Interval | >100 | 50-100 | 10-49 | 2-9 | 1 |
|---|---|---|---|---|---|
| count | 7 | 31 | 1093 | 32616 | 71842 |

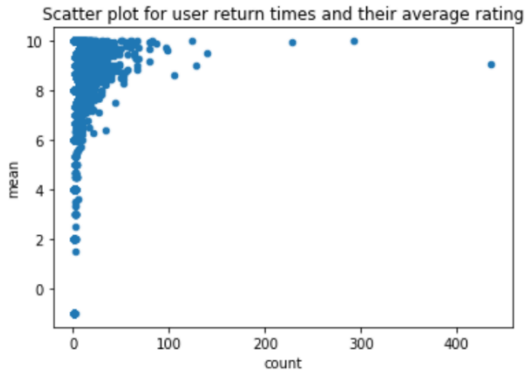
The customer that rent the most clothes have 436 transactions. Despite of this one large amount, the majority of the customers only rent clothes once. The total number of items that have been rented in Rent the Runway dataset is 5850 and the chart below indicate the distribution of times that one item is rented. From the table, we have very popular item that has been rented more than 2000 times and also majority of item rented under 10 times.

| Interval | >1000 | 100-1000 | 50-99 | 10-49 | 2-9 | 1 |
|---|---|---|---|---|---|---|
| count | 9 | 303 | 538 | 2538 | 2121 | 341 |

The fit column in the dataset has values: 'small', 'fit', and 'large', which are self-reported feedback from the customers about how the clothes they rented suit them and the value counts are displayed in the chart below.




From the chart, most people feel their clothes are fit, while small proportion of people are thinking the clothes are either small or large. If we use this column as the target label for our model development, we need to conquer the imbalance of label values.
We want to distinguish the column 'fit' and 'rating' here. 'Rating' represents overall customers' satisfaction for the service, while 'fit' only focuses on the size of the clothes. The chart below explores the relation between the average rating from customer and their return times. Users who return multiple times tend to have a higher average ratings than one-time users




Next, we will explore the body shape column, which records customers body shape type, trying

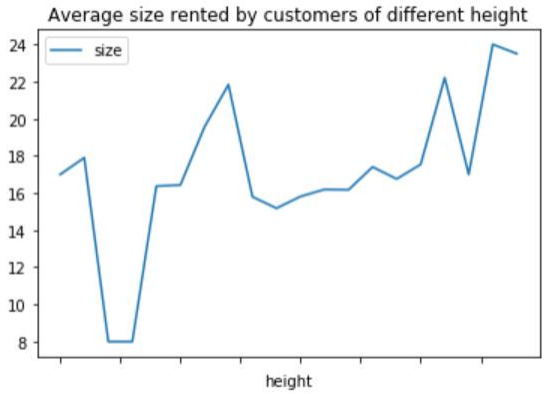
to find if there is a certain body type that tends to have a higher unfit rate. From the table below, we can see with occurrences of different body shape type, the unfit proportion for each category remains almost the same. Though it need to use hypothesis testing to confirm the variation between the proportion are not significant, we will not to do it here because it is not our main focus on this report.

| | body type | unfit count | unfit proportion |
|---|---|---|---|
| hourglass | 55349 | 14493 | 0.261848 |
| athletic | 43667 | 11210 | 0.256716 |
| pear | 22135 | 5998 | 0.270974 |
| petite | 22131 | 5685 | 0.256879 |
| full bust | 15006 | 4232 | 0.282021 |
| straight & narrow | 14742 | 3744 | 0.253968 |
| apple | 4877 | 1390 | 0.285011 |

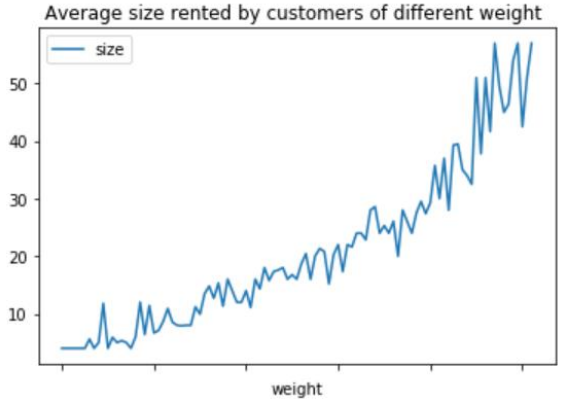
Similarly, we use the same approach to explore the 'rented for' column. This column contains information regarding to occasions that client rented for. The result is displayed in the following table. As we can see, the unfit proportion for each occasion falls in the 20% - 30% range. The occasion 'formal affair' has the lowest unfit proportion while 'everyday' has the highest unfit proportion. One guess is that in formal affair, people tend to be more careful about their selection, while for daily outfit they will be more casual and tend to try out different size and style.

| | rented for | unfit count | unfit proportion |
|---|---|---|---|
| wedding | 57784 | 14673.0 | 0.253928 |
| formal affair | 40408 | 9319.0 | 0.230623 |
| party | 35626 | 9752.0 | 0.273733 |
| everyday | 16822 | 5214.0 | 0.309951 |
| other | 15388 | 4054.0 | 0.263452 |
| work | 15042 | 4185.0 | 0.278221 |
| date | 7388 | 2127.0 | 0.287899 |

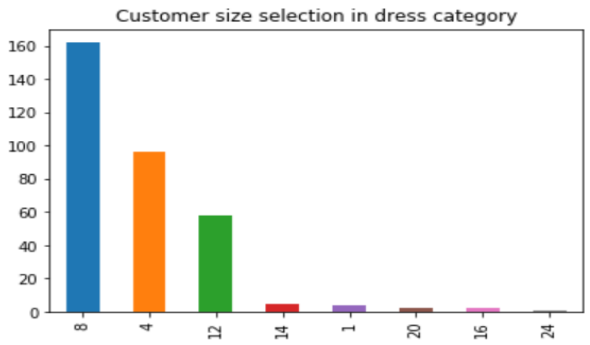
After the general bivariate analysis between categorical column with fitting result, we want to focus more on one particular popular item to find what sizes people rent and how their body shape information varies. The following chart show the average size rented by customer of different height for item 126355, which is a dress. Here we cannot identify a clear trend based on the height or the body shape of the customers. This is counterintuitive, since we usually deem taller people would wear larger size of clothes.




However, when we plot average size against weight, trend is more explicit in this case. The size is more correlated with weight in this datset.




If we focus on a particular customer, who have multiple transactions, taking the customer with user_id = 691468 for example, and check the size variation this customer rented in dress category. The results are displayed in chart below: It is weird that one particular customer rent dresses with various sizes. One possible explanation is this one customer rent clothes not just for herself, but also for family and friends using a single account with same body shape data.




We will try to address this problem more in the following investigation.

**Prediction task**:

With this dataset, we plan to build a prediction model using customers' body size, age data to predict the fitting results and further recommend an item with suitable size to customers based on their body shape data, age and the occasion they rented for. This prediction model can help the business to improve their user experience by recommending attractive items to potential customers and provide an accurate size guide.

Evaluation and Validity

To evaluate our prediction results, we will split the available dataset into training, validation, and test set to 8:1:1 and get the accuracy by the proportion of correctly predicted fit results. However, from our data exploration above, we can see the fitting results are highly imbalanced, with 70% results are 'fit'. Thus, using f-beta score will be more suitable for model evaluation. In order to assure the validity of our model's prediction, we adjust our model's parameter value on our validation set to achieve the optimized score. Then we test the model on the test set to avoid the overfitting issue. The test set assures our model's generalization ability.

Baseline

In our baseline, we use KMeans from sklearn to cluster customers body shape data. We train our baseline model with 80% of the data and compute the average size of the clothes in each cluster. Then we find the cluster each validate set data belongs to and compute the difference between the average size of the cluster and the true size that customer bought. If difference > 5, we predict large, else if difference <-5, we predict small, if the difference falls in the range of [-5,5], we predict fit. The threshold and number of clusters are subjected to change when we test out our baseline model to achieve maximum performance.

Data cleaning

In order to test our baseline, we need to preprocess the dataset and transform it into proper format. The following bullet points are the field we use as our feature and our way to transform it:

- Age: The age group a user belongs to can change a user's definition of fit. For young people, an oversized hoodie is fit to their taste, while the elderly may have a hard time appreciate this style. The age column in the dataset is in type of string, so we convert it into integer.
- Body shape data: These data directly reveals information about user body shape and directly contribute to fitting prediction. Similar body shape users tend to rent similar size of the clothes. The 'body type' column is a categorical column, we will use OneHotEncoding to transform it into numerical data.
  For 'height' column, it stores string type of data with height in inches. We convert it into centimeters using inch to cm formula and store them as float.
  For 'weight' column, we strip the unit 'lbs' at the end of the data and convert it into integers.
  As for the bust size, the number it starts with represents the size for bust and the characters at the end represents the cup size. So, we split these two parts into separate columns 'bust', and 'bust cup size'. We find the corresponding value for each bust cup size and convert this column into the numeric values.

- Rental information: The rental data including information about the transactions. The ‘user_id’ and ‘item_id’ columns are for identifying customers and items in data type of string.
  The category data indicates the category of the clothes, so we use OneHotEncoding to transform it into numeric values.
  Similarly, for ‘rented for’ column. For ‘review_date’ column, we convert the date of transactions into column by computing the number of days to the earliest transaction and store it as integers.
  As for review_text column, we use tfidf vectorizer to transform the text data with dictionary size of 1000.

After transforming all the fields, we inspect the null values in the dataset. We decide to drop all the rows that contain null entries because it add to training difficulty. The number of rows shrink to 146381, which is still enough for our analysis.

After cleaning the data, we apply the baseline model to the dataset, and reaches the accuracy around 74% and f-beta score with weighted average and beta = 0.7 of 57% score on the test set when the fit threshold was set to 5.

**Model and Result**:

We tested three models, Logistic regression, K nearest neighbors and linear support vector machine, aiming to predict the fitting result for certain customer with a clothes item. With users' information, our features contain ages, body type, bust size, bust cup size, heights, weights, number of days passing, category of the product, size of the product and purpose for the renting. We also add feedback information, features contain ratings of the overall shopping experience and 1000-dimensional vectors referring to customers’ review text. Extended from the baseline, we add a feature about the cluster label as another feature to our dataset. Therefore, the number of dimensions of our feature is 1012.

The first model was Logistic regression. Logistic Regression was similar to Linear Regression, but the difference was that Logistic Regression provided with constant output while Linear Regression with continuous output. The algorithm behind Logistic Regression in our model was to use a linear decision boundary to separate customers with different “fit” category by assigning a weight to each feature. Moreover, the model here used sigmoid function to convert the output from linear function to the probability (detailed functions below)

$$y = \beta 0 + \beta 1X1 + \beta 2X2 + \ldots\ldots + \beta nXn$$

$$p = 1/1 + e^{-y}$$

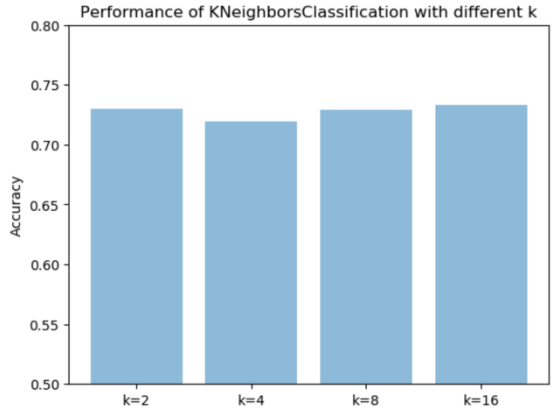
$$p = 1/1 + e^{-(\beta 0 + \beta 1X1 + \beta 2X2 \ldots\ldots \beta nXn)}$$

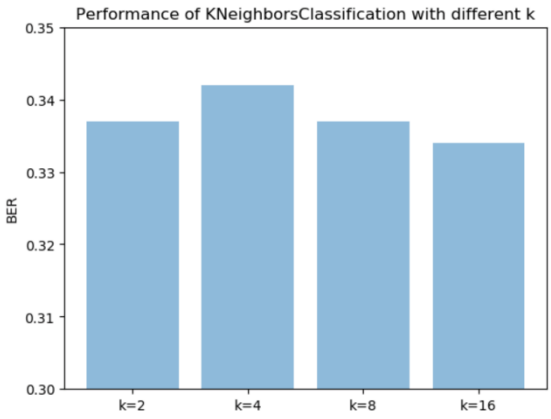
In the first phase of using logistic regression, the accuracy was around 65%. It was not ideal. Due to the imbalanced training data, our model couldn’t learn ‘small’ and ‘large’ categories well. Therefore, we added ‘class_weight’=‘balanced’ in the model, allowing the data with ‘small’ and ‘large’ labels repeating more times in the training phase. As a result, the accuracy of the LR model is slightly over 80%.

The second model we used was support vector machines, finding the decision boundary that maximize the distance between the closest members of separate classes. In SVM model, our team chose two different kinds of decision boundary: linear and non-linear. An svm with a linear kernel is similar to logistic regression, but the performance, slightly over 70%, of linear support vector classification was not as good as logistic regression. As for SVM with non-linear decision boundaries, it took  days to finish training the dataset with unideal performance. In conclusion, SVM was fairly robust against overfilling, especially in high-dimensional space, but SVM was memory intensive, which wasn’t appropriate here.

The last model we used was KNeighborsClassification: this model was based on the baseline that instead of using only the users’ features, we added more about users’

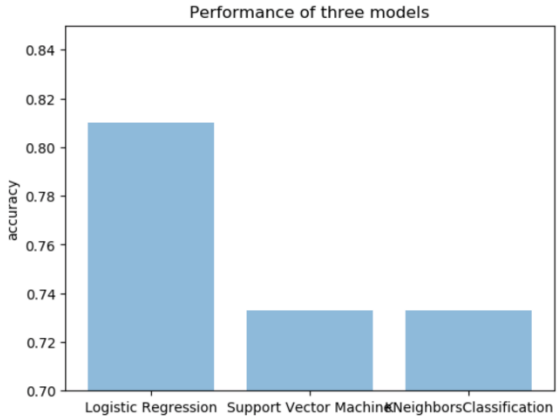
feedback, like ratings and high-dimensional text vectors to the model. Our team aimed to cluster the customers from a simple majority vote of the nearest neighbors of each point: a query point was assigned the data class which has the most representatives within the nearest neighbors of the point. Therefore, in the KNeighborsClassification model, we experimented on different size of neighbors by assigning n_components to different integers. In general, a larger k suppressed the effects of noises in the dataset, but made the classification boundaries less distinct. Here is the table representing the accuracy and balanced error rate of different k.

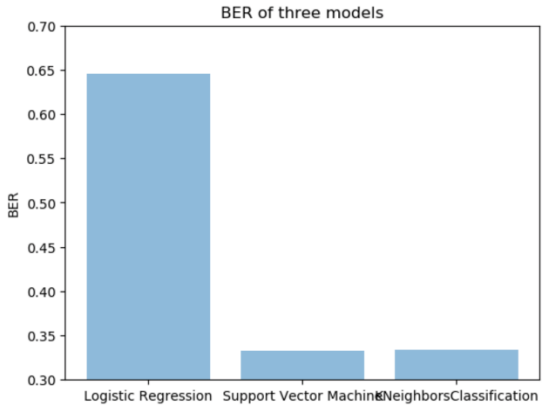



According to the graph, we found that altering k didn't improve the model very much.

Based on experiments with different models and parameters, our team decided to use logistic regression with the best performance. Compared with other two models we used, Logistic regression predicted with a nice probabilistic interpretation and the algorithm could be regularized to avoid overfitting. However, since this model used linear decision boundaries, it was hard to classify data points with non-linear decision boundaries or more complex relationship. Therefore, the reason why the performance couldn't be further improved in our dataset was probably because the relationship between features and fit category wasn't completely linear, or simple in other words.




According to the two graphs of accuracy and BER, we could see that though Logistic Regression had the best performance, it also had the worst Balanced Error Rate. Balanced error rate was a number referring to the balanced probability of misclassification. However, in our dataset, the proportion of data with different labels was imbalanced. Therefore, BER was not a good index to the performance of the model.

**Literature review:**

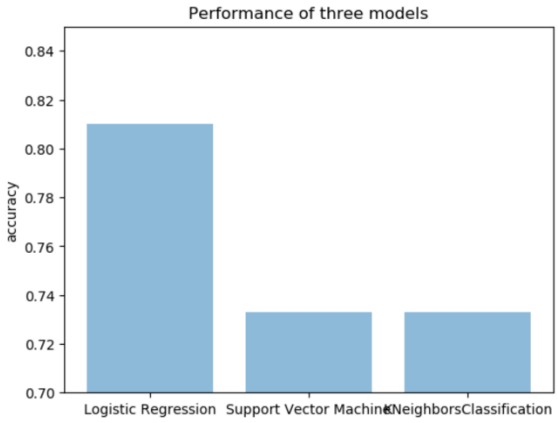
Our dataset is provided by Rent the Runway, which is an online rental service enable female customers rent clothes in different styles

for various occasions. This dataset was used mainly on size and item recommendations under online shopping scenario where customers input their body shape data, age, and occasions they rent for and got recommendations of clothes that fits the occasion with size based on the prediction result of the system.

Some related literature includes the *Recommend Product Size to Customers by Vivek Sembium, Rajeev Rastogi, Atul Saroop, Srujana Mefugu*. In this paper, researchers build a size recommender system by extracting the true size of clothes and customers based on the review they reported. They define a Hinge loss function and logistic loss function to fit on a training set and use gradient descent to get optimized weights vector, which minimizes the difference between true clothes size and customer size. Their approach was designed to conquer problems that are prevailing in this type of dataset:
1. Data sparsity, where a few items and customers contribute a lot transactions while the rest only have a few.
2. Cold start, where new customers need a size recommendation that have no past transaction
3. Multi-persona, people tend to rent for family and friends in distinct sizes but under the same account.
The first problem are solved by incorporating demographics information and product features into the model. The Hinge Loss and Logistic Loss functions are extended to include this information with an additional weights vector for all these features. This weight vector for demographic and product information will also be calculated using gradient descent.
The last problem can be addressed by hierarchical clustering. For each customers with lots of purchases, fit outcome will be clustered based on the product's true size and able to identify the latent personas behind a single account.
Another related work is *Decomposing Fit Semantics for Product Size Recommendation in Metric Spaces by Rishabh Misra, Mengting Wan, and Julian McAuley*. This work use the same dataset and another Modcloth dataset, which sells vintage women clothing and accessories online. Researchers focus more on extracting semantic information from review text to understand the aspect of fitting that each customers care about and make prediction of the fit.
The state-of-art method to study these types of dataset could be recognized as two types: Logistic Regression and Metric Learning. Both of them could be the final classification before assuming single of multiple latent factor or latent variable for each customers in the dataset. (From the literature we are using). By assigning proper latent factor, large size product has larger latent factor, fit semantic could be built. However, the problem with imbalanced label still exists. Here, metric learning approach could solve this. Firstly, Metric Learning helps finding the metric distance between a transaction in one class to another transaction in the same class and that transaction to another transaction in different class. In this way, we could identify the closest neighbor of any single transaction. Secondly, Metric Learning learn by linear transformation of the input matrix and use kth nearest neighbour for final classification, which is similar to the method we utilized. Another aspect we learn from the papers are the way researchers building their loss functions. By accurately defining the loss function based on different scenario can precisely increase the predicting accuracy we care about.

The conclusion from the literature is as follows. Using latent variable on transactions and logistic regression on final classification has relatively low improvement from the ModCloth dataset to the RentTheRunWay dataset since logistic regression could not seize biased terms easily. Using latent factors on transactions and Metric Learning as final classification has largest improvement. Also, for cold start, metric learning also outperforms linear regression and metric learning improves quickly as the training data becomes more sufficient.

**Conclusion:**




As presented above, the result of Logistic Regre outperformed the result of either Support Vector Machine with Linear Kernel and KNeighborsClassification. Compared the better accuracy with the worse accuracy of the model, our team found that the success of Logistic Regression was because output of LR had a nice probabilistic interpretation and algorithm could be regularized to avoid overfitting when biases and outliers existed in our dataset. Applying KNeighbors could not solve the problems in the dataset. In our dataset, the main two problems were data sparsity and multi-persona, which meant that clustering customers and generalizing the fit size for each group didn't work well. Also, because there wasn't a consistent size schema for all items on the website, it was difficult to predict the fit with size of a certain item. As for support vector machine, using linear kernel was similar to using linear regression, but underperformed compared with logistic regression. Using non-linear kernel was extremely time and memory intensive though it was fairly robust against overfilling, especially in high-dimensional space.

Compared with baseline, instead of using only the customers' information including age, body type, height and weight, we added more feedback information referring to the overall rating of shopping experience and TF-IDF score for most frequent unigram and bigram of the review text. The performance was better. Our team thought that TF-IDF captured the key information of the transaction, like "dress", "gorgeous" and "not fit", conveying the information of fit.