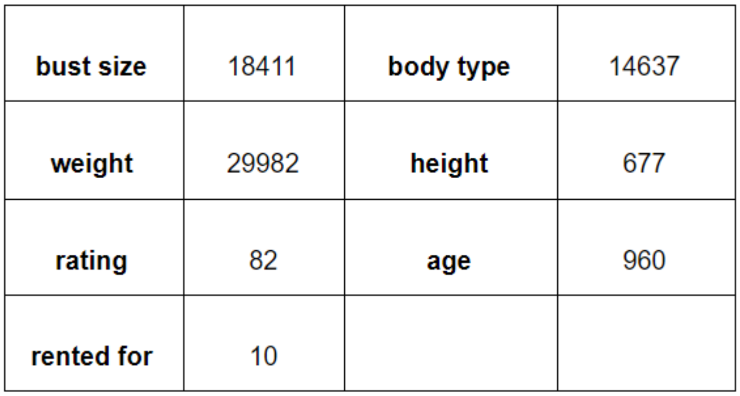
Clothing Fit Prediction: CSE 158 Final Project

1A. Identify Dataset:

Our group selected a dataset from Prof. Julian McCauley's own data repository: https://cseweb.ucsd.edu/~jmcauley/datasets.html#clothing_fit. We chose the *Renttherunway* dataset that McCauley collected from https://www.renttherunway.com/. The dataset is large enough for data analysis with 192,544 transactions(rows) and various features, including long text review, that we can try different types of machine learning tasks on. We believe that the predicted tasks on this dataset could be meaningful, because clothing fitting information is relevant in modern days where online clothing shopping becomes more dominant.

The dataset, however, is not ready for comprehensive data analysis and modelling, because of its outliers and missing values. We performed data imputation on the following columns with missing values:



bust size	18411	**body type**	14637
weight	29982	**height**	677
rating	82	**age**	960
rented for	10		

(Fig 1. Number of missing values in the dataset)

For the categorical columns, such as the body type, rented for, data imputation can be done by replace the missing values as “the other” type, and for numeric columns such as weight, height, bust size, and age, we can replace the missing value based on the mean value of the responding column. Then we decide to normalize the weight and height by subtracting the mean value, which gives us a better understanding of difference and a more accurate result. The age column has a few outliers, like people who are 115 years old. We therefore use log transformation on the age column so that the data would not be largely affected by the age numbers that are much older. We find incompleteness of data interesting because assuming that the missingness is not missing by design, it might not be missing at random, because some people might leave their bodily information blank because they do not want to report their real weight.

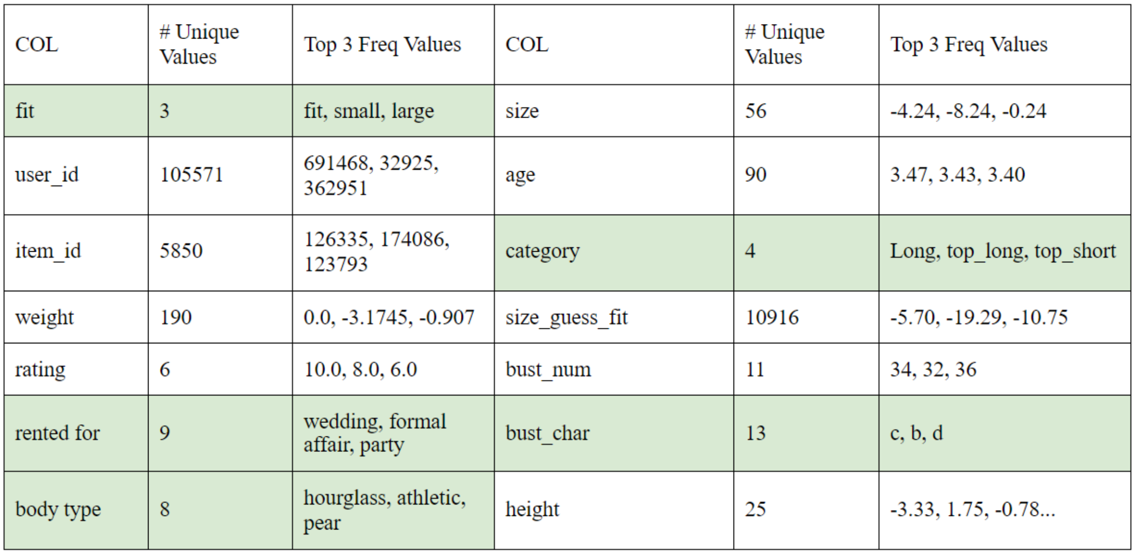
One more notable manipulation we have performed to the dataset, is that as an experiment, we added another feature that is the sum of weight, height, and size. In reality, people who are tall tend to have greater weights and need larger shirts. Also, We know that weight, height, and size are highly associated from direct inference and our exploratory data analysis in 1b, so it would be reasonable to combine these three features to check whether the clothes are fit.

1B. Exploratory Data Analysis:

After data cleaning and imputation, we can perform exploratory analysis on the dataset. The data is structured in a data frame format, making it reasonable to take all its columns as features to fit in the model. The data has 17 columns, including the "fit" column which can potentially be the target/label of the predictive task. There are 7 numerical columns and 10 other types of columns, we can classify them as the following based on their nature:

- ID string: user_id, item_id
- Category string: fit, rented for (reason why the user rent the item), body type, category (item category), bust_char (the ABCD letter size of user's cup size);
- Numerical float: weight (user), rating (user on item), height (user), size (of item), age (user), size_guess_fit (weight+height+size), bust_num (user);
- Long text string: review_text, review_summary, review_date;

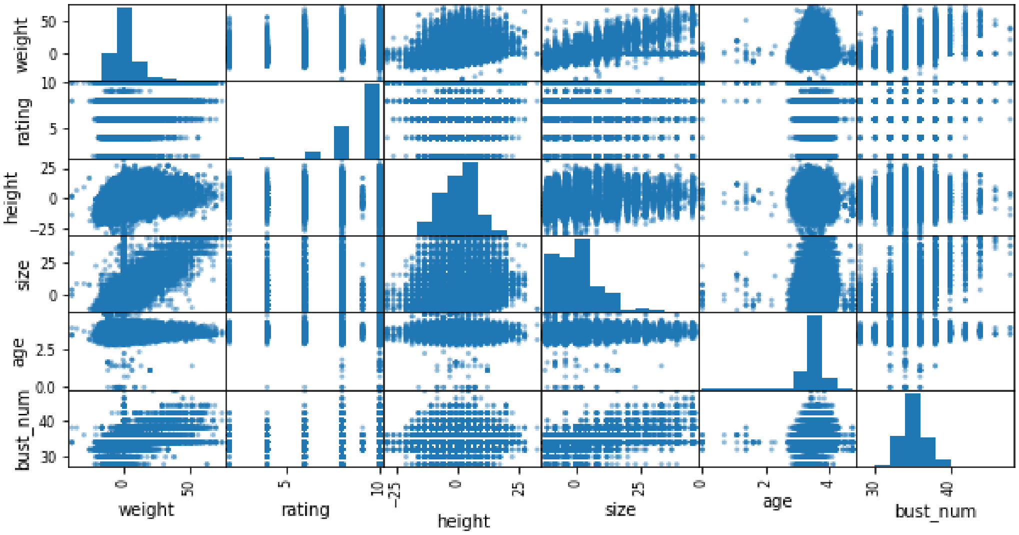
In fig2, we present some examples of each column, that is not a long text string. We calculated the number of unique values for each column, and the top 3 most frequent values. Note that some columns do not make conventional sense, such as the weight column, and that is because we have zero-meaned the column in the previous data cleaning section.



COL	# Unique Values	Top 3 Freq Values	COL	# Unique Values	Top 3 Freq Values
fit	3	fit, small, large	size	56	-4.24, -8.24, -0.24
user_id	105571	691468, 32925, 362951	age	90	3.47, 3.43, 3.40
item_id	5850	126335, 174086, 123793	category	4	Long, top_long, top_short
weight	190	0.0, -3.1745, -0.907	size_guess_fit	10916	-5.70, -19.29, -10.75
rating	6	10.0, 8.0, 6.0	bust_num	11	34, 32, 36
rented for	9	wedding, formal affair, party	bust_char	13	c, b, d
body type	8	hourglass, athletic, pear	height	25	-3.33, 1.75, -0.78...

(Fig 2.Columns after data cleaning & filling in missing values)

There are 105571 unique users and 5850 unique items. In the cells with light green color, we have categorical columns that can be transformed into one hot encoded features. Data cleaning has helped us reduce the number of unique "categories" from 68 down to only 7. For the numerical columns, however, we can explore further by plotting the scatter matrix.



(Fig 3. The scatter matrix of selected columns)

From the scatter matrix, we can see that there is the strongest association between weight and size, and they each have observable associations with other columns such as height and bust nums. However, age and rating do not seem to have associations with other columns at all. Rating is strongly skewed to the left, and weight, height, age, and bust_num all have roughly normal distributions.

2. Predictive Task

Our predictive task, similar to the task in Decomposing fit semantics for product size recommendation in metric spaces in McAurley's paper, is fit prediction [3]. There are three labels of the fit columns: fit, large, and small. Because the data publishers have collected all the personal information of the users, we would assume that knowing the weight, height, and age of the renter, it would be very helpful to predict whether clothes of a certain size will be satisfying to the renter. There are also features, such as the review, the reason for renting, that we know we cannot inherently attain at the instant the user made the decision to rent, but they are still helpful for me to make a decision of fitness when there are hundreds of thousands of rent reviews.

The value count for our prediction label "fit" has the following: out of 192544 values, 142058 are "fit", 22779 of them fit "small", and 24707 of them fit "large". This means, for our predictor, if we train a bad model just simply output "fit" for all the new data, because the majority of the training label is "fit", we would have a 73.7578% accuracy. Therefore, it is important to note that accuracy would not be the only error metrics that we use.

With three class labels, there is no direct way to calculate the number of true positive, false positive, true negative, and false negative, that allows different error metric calculations. However, we can do that by calculating the average of error metrics (such as precision and recall) for each class label. The default error measurement for sklearn package that we will use (mode.score(x_valid, y_valid)) use the accuracy as the error metrics, so we will try to attain the highest accuracy as a goal when changing the hyper-parameters, but still keep our eyes on other types of error metrics, and report them in the end.

The baseline model that we proposed is simplifying using the “weight”, “height”, “size”, “age” factor to train a classifier that predicts fitness. This baseline model is actually a very reasonable approach for the renting company, if they have previously collected the “weight”, “height”, and “age” information on users before the rent, and based on the size of the clothes they rent, the company can output a prediction on whether the cloth will fit the users. This will benefit both the company, as they will expect lower return rate and higher satisfaction from the users, and the users who are uncertain about what size of clothes to rent.

A more complex model will take advantage of the other features collected in the dataset, and we want this model to predict the fitness correctly more than 73% of the time. The model will make more intelligent guesses if we implement a model using Python natural language processing library on the review summary and review text part of each instance, and transform the other categorical columns into readable data (one hot encoded) that the model can fit.

3a. Proposed Model: Feature selection

We split the 192544 data points into size 180000 training data and size 12544 validation set. In the previous model, we proposed a baseline model that only takes a limited amount of numeric features to make predictions. It turned out in our early analysis, that the accuracy of the baseline model is not ideal. (We will report it in the final section) It is just little higher than the accuracy if we predict the majority.

When we fed more features to the model, we ran into scalability issues. There are 180000 rows in the matrix, and adding each new feature to the train matrix created a challenge as our computer's memory is limited. It certainly is a method to just take a smaller sample, but we figured that using an expensive package like pandas dataframe or using a method like "np.hstack" to combine two matrices are not the best idea for memory preservation. Python's default list addition can add two nested lists with 180000 items in less than 5 seconds (np.hstack takes more than 10 minutes to do the same) without creating a copy of the data in the memory.

As we use one hot encoding on rented_for, category, bust_char, body type columns, which are not numerical, we keep the numerical columns and put it into the features. We then combined these features together to create a training matrix. Moreover, we decide to use TF-IDF instead of counting the popular words on our summary and text columns since we also want to know the weight of each popular word in the corresponding text and gives us a better reflection on the understanding of the popular words. However, when we put all the features into the model, we only get 73.62% accuracy, which is worse than the baseline model (74%).

Later, we found out the body type and bust_size are distracting our model since we can not really find the relationship about how the body type and bust size affect the fit of clothing. Also, we believe that age should not be a factor to influence the fit of clothing because obesity can increase the risk of unfitting, which depends on their diet and exercise frequency. After removing the body type and bust size, we have an accuracy of about 0.7908, which is much better than the baseline model.

As we dug into the problem deeply, we realized that the cause of unfitting the dress or shirt is the difference between their understanding of their own weight and expected size. So we think after we use TF-IDF on summary and text, those numerical columns like weight, height, size, category would distract the model, so we decided to delete all those features. However, we chose to keep the rating feature because people are most likely to reflect the fitting of clothing from text summary and rating. After choosing the best feature and putting it into our model, we get 81.73 accuracy, which is the highest accuracy we achieved.

3b. Model selection

However, to jump to the conclusion that we find the best predictor, we must try different machine learning algorithms. Decision tree classifier and random forest classifier can attain at max around 80% accuracy, after we grid-search for the best parameters, such as the max_depth, and min_leaf. Interestingly, we need a large max_depth value (much greater than 10) for the random forest classifier to get a good estimation on the validation set. Logistic regression, however, outperformed other models, including SVM. We had an unsuccessful attempt to use the KNN (euclidean distance as default) to classify the points in the validation dataset, but it was too time consuming, because of the large number of features (2000+).

There are different ways for the natural language processing part of our feature. Other than the tf-idf model, we also tried the bag of words model with stopwords removed. Not surprisingly, the BOW gives a slightly lower, but similar prediction score than the tf-idf model. For both bag of word approach and tf-idf approach, we found that using the 1000 most frequent words is sufficient for the model, because too many features create greater challenge on the memory, and did not show improvement in prediction score.

In the best model we have found, the logistic regression model, we let the max iteration to be as large as possible, or just 100, and we tried different C value using grid search, and we found that smaller C value or larger C value than the default C = 1.0 did not improve the model.

Max iteration 10000 produced consistently good results. Therefore, we left the logistic regression with the default hyperparameter settings with max iteration 10000 to make sure the algorithm converges fully.

4. Related Literature

Rent the Runway is an online platform that services dress and accessory rentals. The data set RentTheRunway contains users' own information and reviews for the product, such as weight, height, age, the size of clothes, rating, text review and etc. The fit feedback contains three classes--small, fit, and large. Rishabn Misra. etc suggests that the data set is used to automatically predict the fitting condition based on other information the user reports that can be applied in fit guidance [3].

Similar data set that has been studied is the sales data from Myntra e-commerce platform. G. Mohammed Abdulla etc. models the size recommendation problem as a classification task [2]. The learning task is to predict the fitting probability of size of clothes, i.e. S, M, L, XL etc. based on the observant feature and the latent feature. The data is composed of the user's self report like the chest size etc.. G. Mohammed Abdulla etc. invents a new feature extraction method in the use of SKU. Each product has various sizes and each combination of size and the product is known as a SKU [2]. They build SKU observable feature vectors based on one-hot encoding the categorical features such as type of clothes and SKU latent feature vector based training word2vec on every SKU which is a semantic text. For example, G. Mohammed Abdulla etc. mentions that a typical SKU is like "Roadster-Casual-Slim-M" in text form and can be fed to word2vec model to extract latent features. After constructing features including SKU features, they formulate a binary classifier with softmax to predict the probability of fitting. They split the data set in time sequence from Jan 2015 to Oct 2016 as training data, from Nov 2016 to Dec 2016 as validation data, and from Jan 2017 to Feb 2017 as test data. The model with combined features gets 81.28% in accuracy.

The state-of-the-art method posted on RecSys 2019 to study this type of data is a deep learning based system to predict size and fit for e-commerce data sets. This method optimizes parameters in global to learn global abstraction of size and fit related content [1]. The basic idea behind is to employ a deep learning model on embedded features to obtain population-level representations of information of user's size and fit condition.

One of the data sets Abdul-Saboor Sheikh etc. used is also RentTheRunWay. They represent the dataset as too sparse to model a vast number of individuals on a personalized level [1]. This conclusion is consistent with our finding when applying our model on a test set to personalize the prediction. Our dataset is also incomplete and sparse with missing entries and might be hard to personalize fitting predictions due to the shortage of dataset itself.

5a. Results.

Mentioned in 3b, we found that logistic regression worked the best in this prediction problem. For many of the combination of features in figure 4, logistic regression outperformed other classification algorithms. For example, for the best feature combination (selected in green color below), random forest classifiers gave the second best result, 0.8093, after grid-search hyperparameter tuning, while logistic regression attains accuracy over 0.81 consistently.

Below we have a few instances of our model/feature selections and their corresponding scores measured in accuracy.

	Accuracy
Baseline Model Numerical: weight; height; size	0.7411
One hot: rented_for; category;bust_char;body_type; Numerical: bust_num; age; size; weight; height; rating TF-IDF: summary; text	0.7362
One hot: rented_for; category; Numerical: size; weight; height; rating TF-IDF: summary; text	0.7908
One hot: category; Numerical: rating TF-IDF: summary; text	0.800223
One hot: category; Numerical: TF-IDF: summary; text	0.803
One hot: Numerical: TF-IDF: summary; text	0.805
One hot: Numerical: rating TF-IDF: summary; text	0.8173

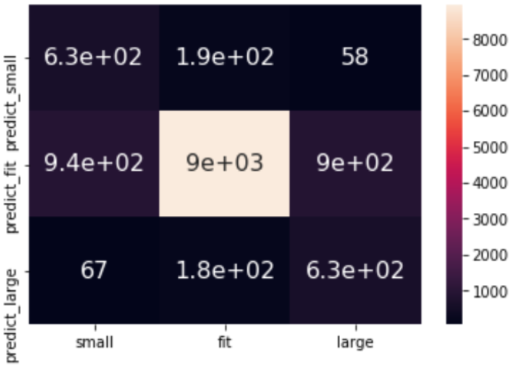
(Fig 4, Logistic regression models' accuracies on different features)

The result is, that if we use the tf-idf model on the text summary and text, using 1000 max feature(words), combined with the rating of the product, we will attain the highest accuracy score with logistic regression with C = 1 and large enough max iterations, we would attain consistently high accuracy on the unseen data.

Shown in fig 5, is the confusion matrix for further error metric analysis on the unseen data, using the best model we attain:

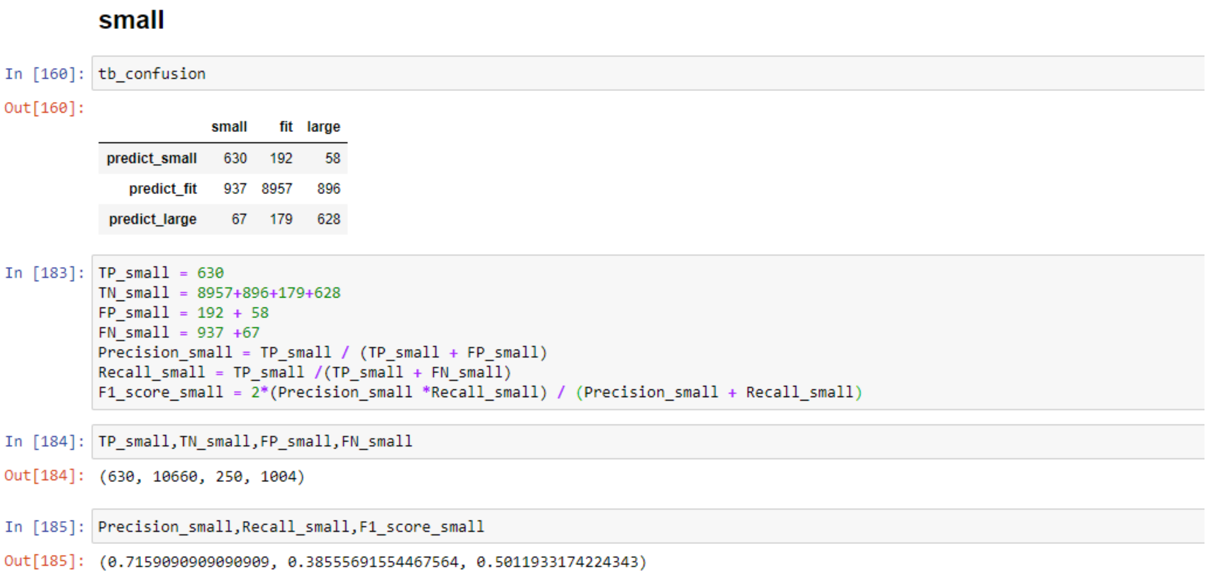
	small	fit	large
predict_small	630	192	58
predict_fit	937	8957	896
predict_large	67	179	628

(Fig 5, Confusion matrix on validation set)



(Fig 6. Heat map of the confusion matrix)

And finally, below is the result of our error metrics, performed on each of the class label (small, large, fit)



small

```
In [160]: tb_confusion
Out[160]:
```

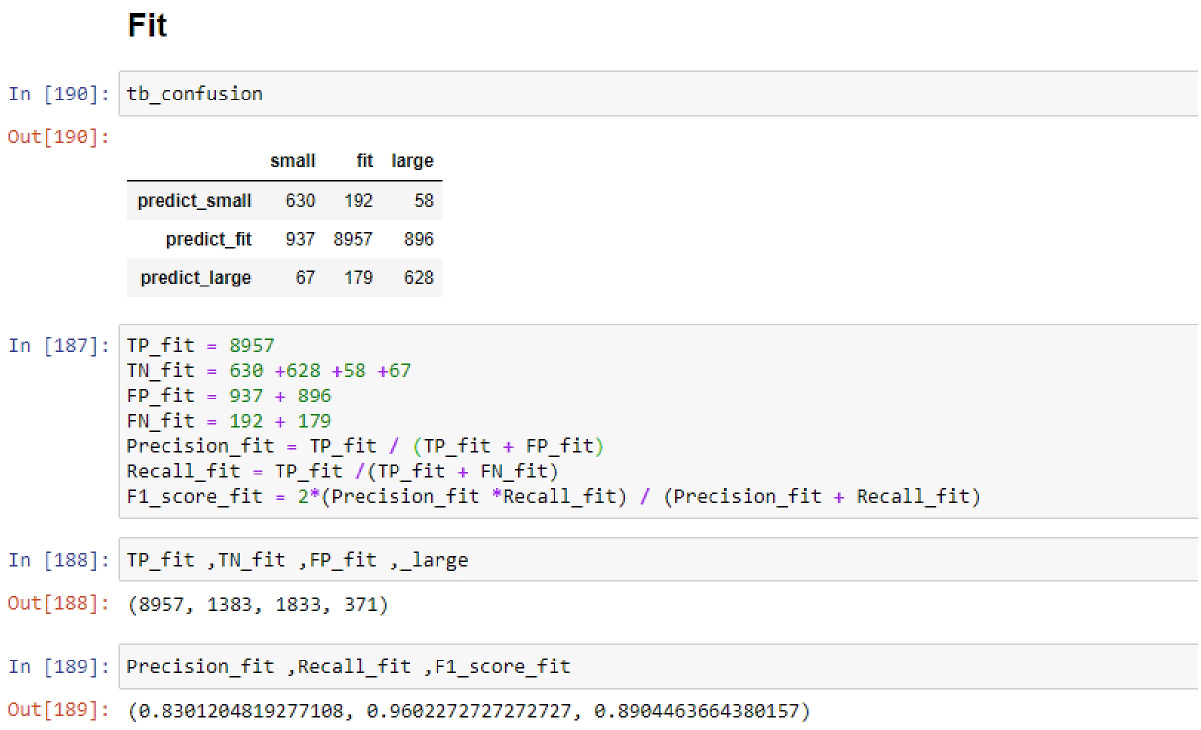
	small	fit	large
predict_small	630	192	58
predict_fit	937	8957	896
predict_large	67	179	628

```
In [183]: TP_small = 630
          TN_small = 8957+896+179+628
          FP_small = 192 + 58
          FN_small = 937 +67
          Precision_small = TP_small / (TP_small + FP_small)
          Recall_small = TP_small /(TP_small + FN_small)
          F1_score_small = 2*(Precision_small *Recall_small) / (Precision_small + Recall_small)

In [184]: TP_small,TN_small,FP_small,FN_small
Out[184]: (630, 10660, 250, 1004)

In [185]: Precision_small,Recall_small,F1_score_small
Out[185]: (0.7159090909090909, 0.38555691554467564, 0.5011933174224343)
```

Fig 7. Model performance metrics on Small labels



Fit

```
In [190]: tb_confusion
Out[190]:
```

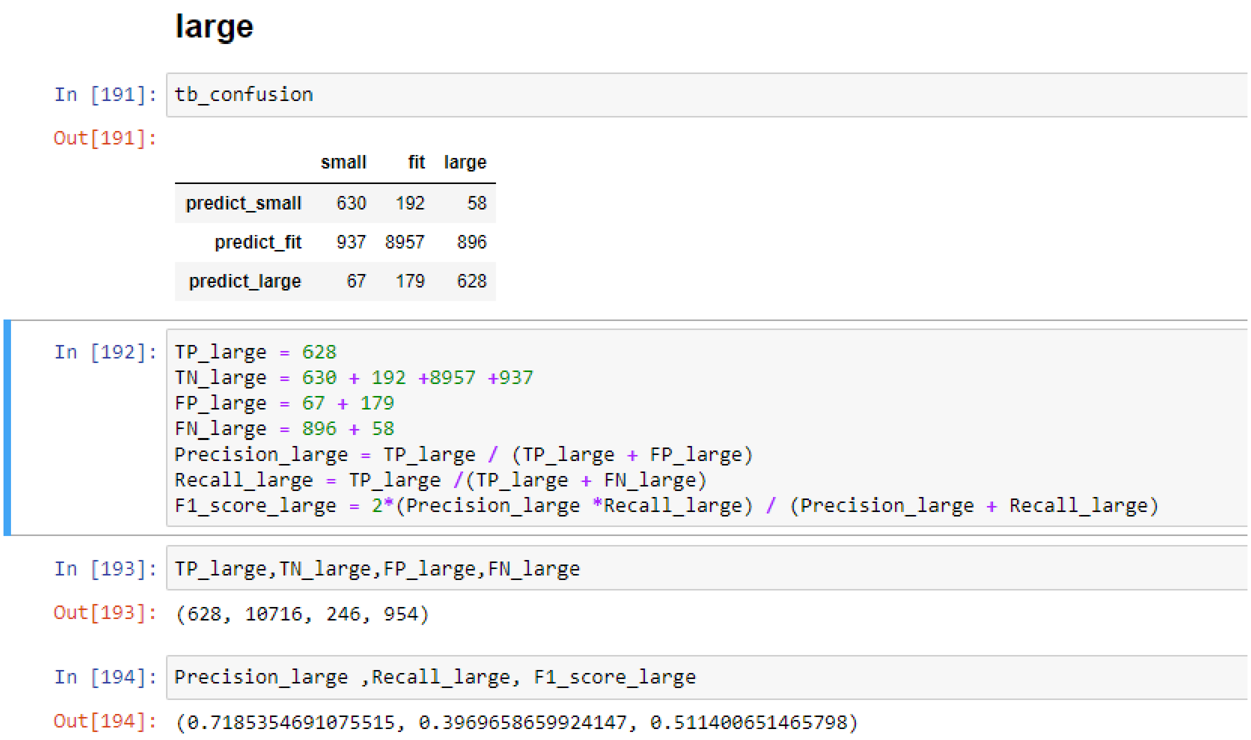
	small	fit	large
predict_small	630	192	58
predict_fit	937	8957	896
predict_large	67	179	628

```
In [187]: TP_fit = 8957
          TN_fit = 630 +628 +58 +67
          FP_fit = 937 + 896
          FN_fit = 192 + 179
          Precision_fit = TP_fit / (TP_fit + FP_fit)
          Recall_fit = TP_fit /(TP_fit + FN_fit)
          F1_score_fit = 2*(Precision_fit *Recall_fit) / (Precision_fit + Recall_fit)

In [188]: TP_fit ,TN_fit ,FP_fit ,_large
Out[188]: (8957, 1383, 1833, 371)

In [189]: Precision_fit ,Recall_fit ,F1_score_fit
Out[189]: (0.8301204819277108, 0.9602272727272727, 0.8904463664380157)
```

Fig 7. Model performance metrics on Fit labels



large

```
In [191]: tb_confusion
Out[191]:
```

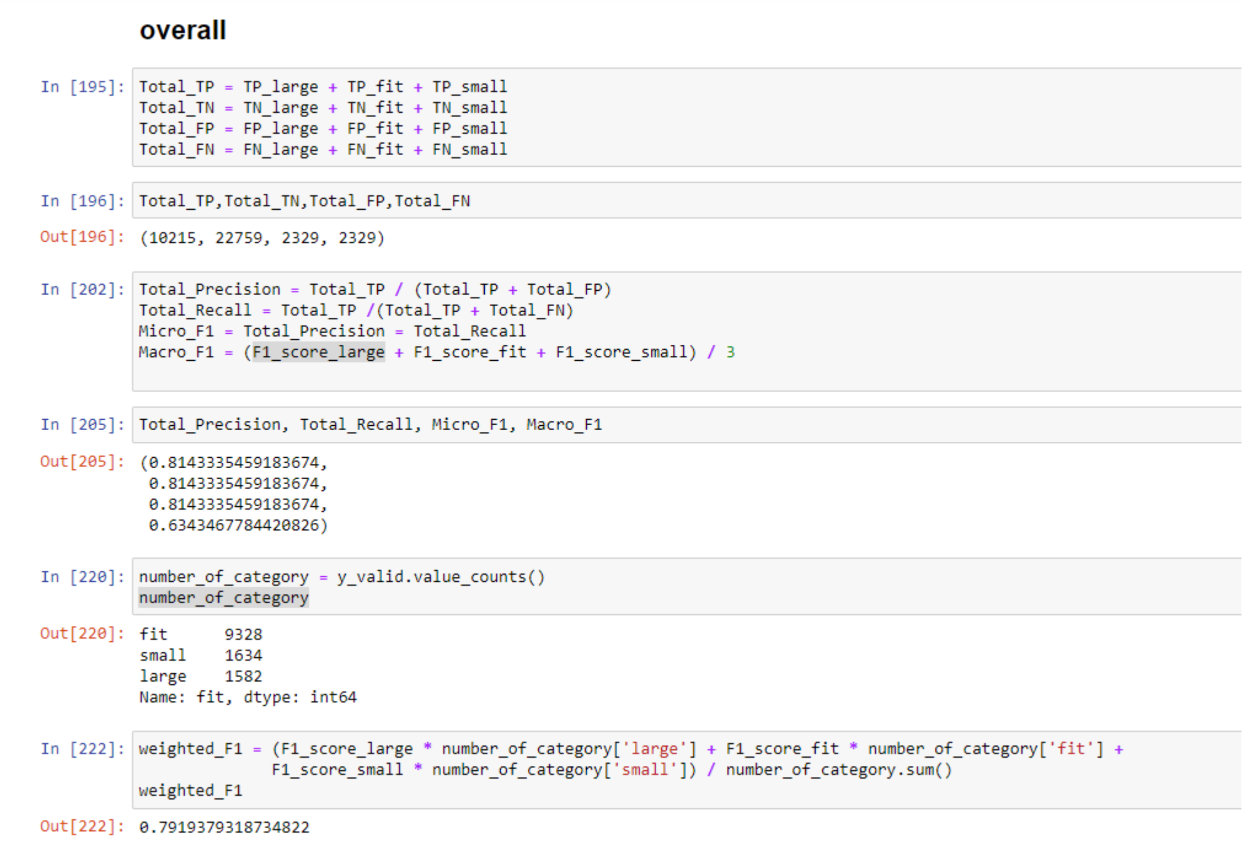
	small	fit	large
predict_small	630	192	58
predict_fit	937	8957	896
predict_large	67	179	628

```
In [192]: TP_large = 628
          TN_large = 630 + 192 +8957 +937
          FP_large = 67 + 179
          FN_large = 896 + 58
          Precision_large = TP_large / (TP_large + FP_large)
          Recall_large = TP_large /(TP_large + FN_large)
          F1_score_large = 2*(Precision_large *Recall_large) / (Precision_large + Recall_large)

In [193]: TP_large,TN_large,FP_large,FN_large
Out[193]: (628, 10716, 246, 954)

In [194]: Precision_large ,Recall_large, F1_score_large
Out[194]: (0.7185354691075515, 0.3969658659924147, 0.511400651465798)
```

Fig 7. Model performance metrics on large labels



overall

```
In [195]: Total_TP = TP_large + TP_fit + TP_small
          Total_TN = TN_large + TN_fit + TN_small
          Total_FP = FP_large + FP_fit + FP_small
          Total_FN = FN_large + FN_fit + FN_small

In [196]: Total_TP,Total_TN,Total_FP,Total_FN

Out[196]: (10215, 22759, 2329, 2329)

In [202]: Total_Precision = Total_TP / (Total_TP + Total_FP)
          Total_Recall = Total_TP /(Total_TP + Total_FN)
          Micro_F1 = Total_Precision = Total_Recall
          Macro_F1 = (F1_score_large + F1_score_fit + F1_score_small) / 3

In [205]: Total_Precision, Total_Recall, Micro_F1, Macro_F1

Out[205]: (0.8143335459183674,
           0.8143335459183674,
           0.8143335459183674,
           0.6343467784420826)

In [220]: number_of_category = y_valid.value_counts()
          number_of_category

Out[220]: fit      9328
          small    1634
          large    1582
          Name: fit, dtype: int64

In [222]: weighted_F1 = (F1_score_large * number_of_category['large'] + F1_score_fit * number_of_category['fit'] +
                         F1_score_small * number_of_category['small']) / number_of_category.sum()
          weighted_F1

Out[222]: 0.7919379318734822
```

Fig 7. Model performance metrics on overall labels

5b. Critique/Further improvement idea

As we can see through other error metrics, when the model is trying to predict the true small and large items, although accuracy and weighted_f1 score seems high, the recall score (also known as sensitivity), indicating how much items of the true labels are correctly identified, is much lower compared to the precision score. There is definitely room for improvement and we believe this score can increase if more sophisticated methods are used. Also, the natural language processing model might be too unrealistic to run in real time for the company to implement, so it is useful for analysis of large quantities of data, but there could be much more efficient ways to attain similarly accurate predictions by using other features.

6. Citations.

[1] Abdul-Saboor Sheikh, Romain Guigoures, Evgenii Koriagin, Yuen King Ho, Reza Shirvany, Roland Vollgraf, Urs Bergmann. “A Deep Learning System for Predicting Size and Fit in Fashion E-Commerce.” *RecSys*, 2019.

[2] G. Mohammed Abdulla, Sumit Borar. “Size Recommendation System for Fashion E-commerce.” *KDD Workshop on Machine Learning Meets Fashion*, 2017.

[3] Rishabh Misra, Mengting Wan, Julian McAuley. “Decomposing Fit Semantics for Product Size Recommendation in Metric Spaces.” *RecSys*, 2018.