Data-Driven Analytics for Obesity Management and Business Strategy

Author: Zhetan Zhang, Mingqi Yang

Team: Team ZY

1. Executive Summary

1.1 Decisions to be impacted

- Does this person have obesity?
- How to prevent obesity by changing lifestyle?
- What action should companies do to help obese people?
- What are the most important features that cause obesity?

1.2 Business value

- Nutrition companies and medical companies can design products that are more acceptable and effective for obese population based on their dietary habits
- Rehabilitation centers can provide more personalized services for obesity in order to broaden their prospective customers
- Internet companies can design mobile applications that can help potentially obese people to monitor their weights.

1.3 Motivation

Obesity is always a serious problem all over the world. According to the World Obesity Federation which is a membership that is dedicated to the study and management of obesity, there are a large percent of people that have an obesity problem. With their prediction, the number and percentage will continue to increase as you can see from the table below. In addition, obesity will impact the economy. Also, from the table that the impact of obesity on obesity has a trend to increase in the future. Because of this, our project's main purpose is to study the reason that causes obesity and based on the phenomena we find to give advice to companies to help obese people based on building machine learning models on predicting obesity type.

Table 1.3: Global economic impact of high BMI (BMI ≥25kg/m²) 2020–2035

	2020	2025	2030	2035
Economic impact (US$ at 2019 value) (trillions)	US$ 1.96	US$ 2.47	US$ 3.23	US$ 4.32
Impact as proportion of total global GDP	2.4%	2.5%	2.7%	2.9%

Adults (aged 20 years and over)

	Men 2020	Men 2025	Men 2030	Men 2035
Number with obesity (millions)	347	439	553	690
Proportion of all men	14%	16%	19%	23%
	Women 2020	**Women 2025**	**Women 2030**	**Women 2035**
Number with obesity (millions)	466	568	693	842
Proportion of all women	18%	21%	24%	27%

1.4 Data assets

- Estimation of obesity levels based on eating habits and physical condition . (2019). UCI Machine Learning Repository. https://doi.org/10.24432/C5H31Z.

This data set included an abundant set of variables, we can generate a wide-ranging view about how various factors collectively contribute to obesity. For example, by analyzing the features related to individuals' eating habits and their corresponding obesity levels, we can offer guidance to nutrition companies on developing more effective slimming products that are not only appealing to obese population but also effective in reducing weight. Besides, examining physical condition features also provide insights on how to suggest infrastructure companies such as public transit services or medical rehabilitation centers on optimizing convenience or construct appropriate exercise regimens for obese populations.

2. Data description/preprocessing

2.1 Data Description

The dataset we are using for this project is called "Estimation of obesity levels based on eating habits and physical condition, " sourced from the UCI Machine Learning Repository. The dataset essentially serves as classification tasks containing 16 features and 2111 samples, with the goal of predicting obesity levels in different individuals. Among 16 features, 6 of them contain nominal data (e.g. Gender, Transportation Used), 2 of them contain ordinal data (e.g. Consumption of food between meals), 1 of them contains interval data, which is Age, and 7 of them contain ratio data. (e.g. Height, Weight)

To visualize the data, we separate it into two parts— numerical data and categorical data. For numerical data, we use bar plots to show the cumulative distribution of the data set. We have two bar plots which show the distribution of the raw data and the data after normalizing. (See figure 1 and figure 2)

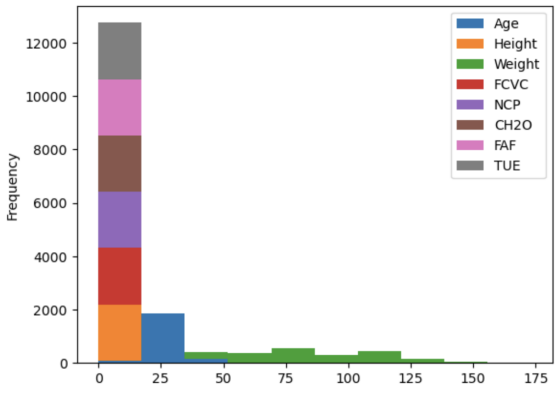


Figure 1: Raw data

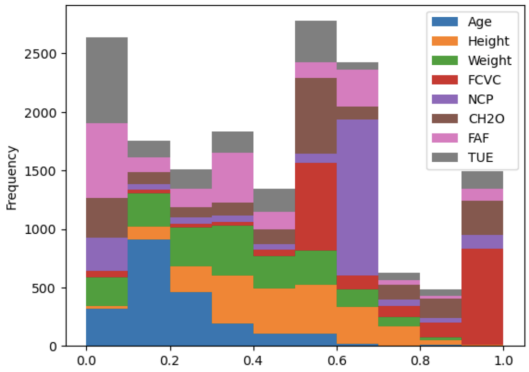


Figure 2: Normalized data

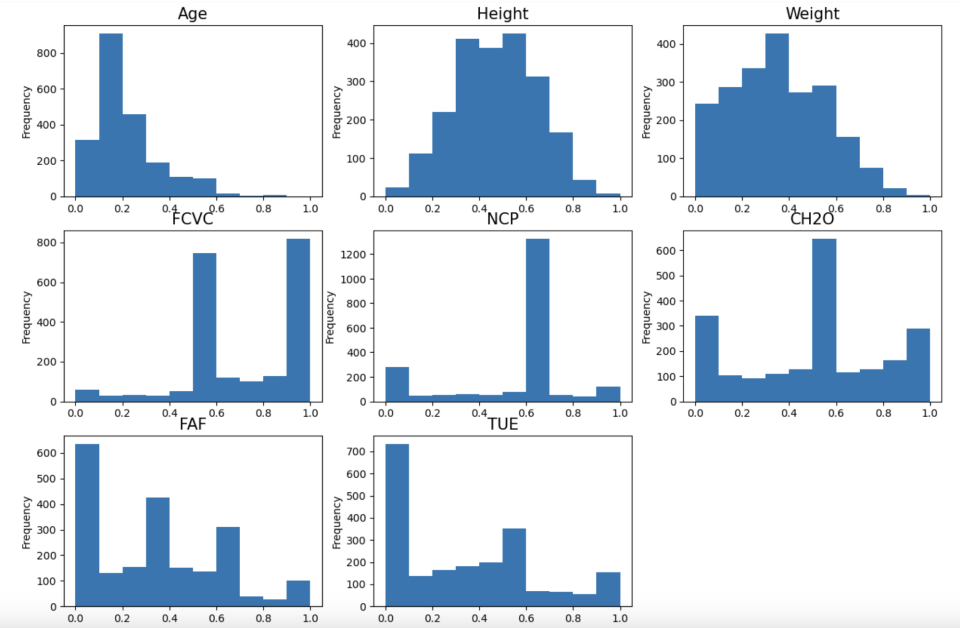


Figure 3: Data distribution

After normalizing, we can see that the data are all shrink into the range [0,1] and have better distribution than the raw data. Then, we can use correlation matrix to see if there any features have high correlation.(See figure 4)

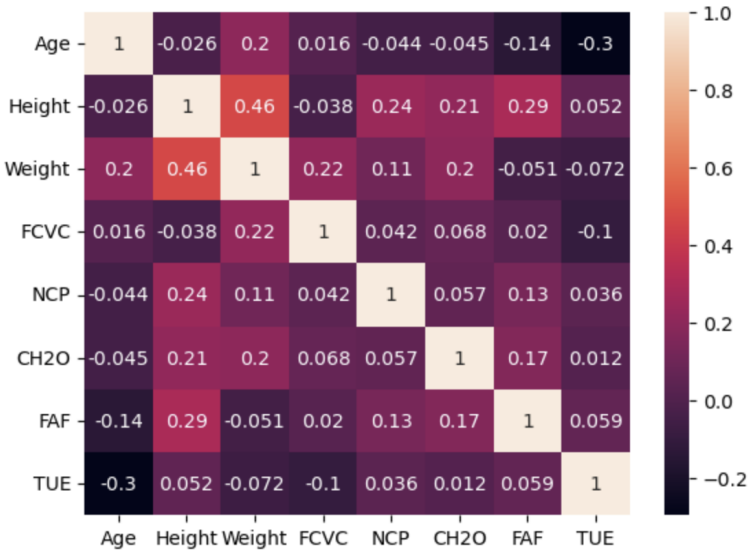


Figure 4: Correlation Heat map

From the correlation heat map, we can see there are no pairs of numeric features that have a correlation greater or equal to 0.8 which means that we can assume each numeric feature is independent of each other.

For categorical data, we used pie charts to show the distribution.(See figure 5)

Figure 5: Categorical data Pie charts

2.2 Data Cleaning

We used one-hot-encoding to encode categorical data to transform categorical data into numeric data. For each unique category in the categorical variable, one-hot-encoding will create a new binary column. In the new binary columns, you assign a value of 1 if the original data point belongs to the category represented by that column, and 0 if it does not. These binary columns form a new feature matrix, where each row represents a data point, and each column represents a category from the original categorical variable.

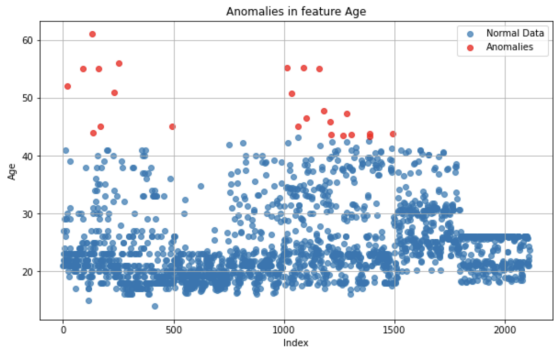
2.3 Outlier Detection

For identifying outliers in our dataset, we primarily employed different statistical methods on both numerical and categorical features. Specifically, for each numerical attribute, we calculate its corresponding standard deviation and mean value, which has shown below.

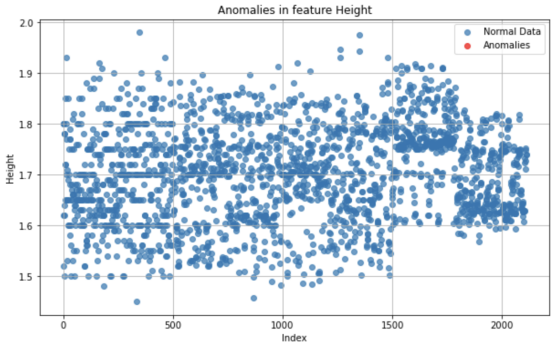
	Age	Height	Weight	FCVC	NCP	CH2O	FAF	TUE
count	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000
mean	24.312600	1.701677	86.586058	2.419043	2.685628	2.008011	1.010298	0.657866
std	6.345968	0.093305	26.191172	0.533927	0.778039	0.612953	0.850592	0.608927
min	14.000000	1.450000	39.000000	1.000000	1.000000	1.000000	0.000000	0.000000
25%	19.947192	1.630000	65.473343	2.000000	2.658738	1.584812	0.124505	0.000000
50%	22.777890	1.700499	83.000000	2.385502	3.000000	2.000000	1.000000	0.625350
75%	26.000000	1.768464	107.430682	3.000000	3.000000	2.477420	1.666678	1.000000
max	61.000000	1.980000	173.000000	3.000000	4.000000	3.000000	3.000000	2.000000

Then, we consider the feature values that are 3 standard deviations from the mean as anomalies. The graphs below shows the results of outlier detection on numeric

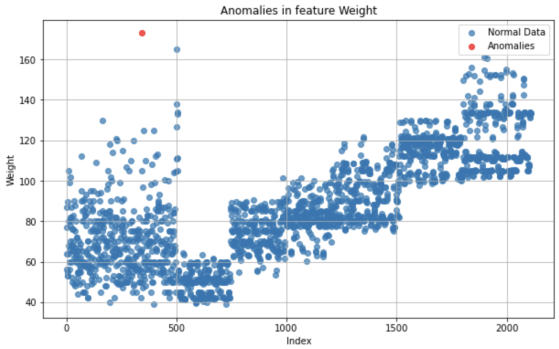
```
Feature Age:
Number of normal data points: 2087
Number of anomalies: 24
Percentage of anomalies: 1.149976042165788%
```




```
Feature Height:
Number of normal data points: 2111
Number of anomalies: 0
Percentage of anomalies: 0.0%
```




```
Feature Weight:
Number of normal data points: 2110
Number of anomalies: 1
Percentage of anomalies: 0.047393364928909956%
```



Based on the graphs above, we can see that only three features show the existence of outliers. Specifically, when examining the “age” feature, we observed that samples with ages above 40 are outliers; when examining the “weight” feature, we observe only one sample with weight over 160 kg is outlier. To address “age” outliers, we intend to initially apply models like logistic regression to assess their effect on the models’ accuracy. For the outlier in “weight” feature, we decide to ignore it because it is unlikely to significantly impact the general performance of the model.

For each categorical feature, we have chosen to classify feature values that appear fewer than 5 times as the outliers. Given that our dataset comprises around 2000 samples, we believe this outlier definition is justifiable. The graph below shows the results of outlier detection on categorical dataset:

```
Anomalies in Gender:
Series([], Name: Gender, dtype: int64)

Anomalies in family_history_with_overweight:
Series([], Name: family_history_with_overweight, dtype: int64)

Anomalies in FAVC:
Series([], Name: FAVC, dtype: int64)

Anomalies in CAEC:
Series([], Name: CAEC, dtype: int64)

Anomalies in SMOKE:
Series([], Name: SMOKE, dtype: int64)

Anomalies in SCC:
Series([], Name: SCC, dtype: int64)

Anomalies in CALC:
Always    1
Name: CALC, dtype: int64

Anomalies in MTRANS:
Series([], Name: MTRANS, dtype: int64)

Anomalies in NObeyesdad:
Series([], Name: NObeyesdad, dtype: int64)
```

To be specific, we try to create panda series for the outliers within each feature. Based on the graph, you can see that for most of the features, no outlier exists. However, for the feature ‘Consumption of alcohol (CALC)’, we identified a sample with the value "always" as an anomaly. Similar as before, we decide to ignore this single outlier because it will not have a significant effect on the model’s performance.

We then used LOF(Local outlier factories) to detect outliers. LOF is a density-based method that identifies outliers by comparing the local density deviation of a data point with respect to its neighbors. The basic idea is that outliers are points that have a significantly lower local density compared to their neighbors. We used the default parameter of LOF in sklearn which is n_neighbor = 20. By using LOF, we detected 100 samples as outliers which is 100/2111 = 4.7% data. We will use the data set which delete these outliers and compare with the performance of the original data. The data set as shown following:

	Age	Height	Weight	FCVC	NCP	CH2O	FAF	TUE	Gender_Female	Gender_Male	...	CALC_Always	CALC_Frequently	CALC_
0	0.148936	0.320755	0.186567	0.5	0.666667	0.500000	0.000000	0.500000	1.0	0.0	...	0.0	0.0	
1	0.148936	0.132075	0.126866	1.0	0.666667	1.000000	1.000000	0.000000	1.0	0.0	...	0.0	0.0	
2	0.191489	0.660377	0.283582	0.5	0.666667	0.500000	0.666667	0.500000	0.0	1.0	...	0.0	1.0	
3	0.276596	0.660377	0.358209	1.0	0.666667	0.500000	0.666667	0.000000	0.0	1.0	...	0.0	1.0	
4	0.170213	0.622642	0.379104	0.5	0.000000	0.500000	0.000000	0.000000	0.0	1.0	...	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2105	0.155021	0.522491	0.689073	1.0	0.666667	0.398134	0.576111	0.448962	1.0	0.0	...	0.0	0.0	
2106	0.148443	0.491943	0.689616	1.0	0.666667	0.364070	0.558756	0.453124	1.0	0.0	...	0.0	0.0	
2107	0.169850	0.563366	0.707037	1.0	0.666667	0.502565	0.447130	0.299635	1.0	0.0	...	0.0	0.0	
2108	0.181362	0.570200	0.706637	1.0	0.666667	0.527097	0.471403	0.323144	1.0	0.0	...	0.0	0.0	
2110	0.205632	0.544974	0.705020	1.0	0.666667	0.931757	0.342151	0.357069	1.0	0.0	...	0.0	0.0	

2011 rows × 32 columns

2.3 Dimensional Reduction

In the real world, the datasets are often high-dimensional and large-sized. Therefore, we may want to perform feature engineering such as PCA on the datasets which apply the outlier detection method to reduce the number of features and still maintain the variance of most features, which may make our model less complex and faster to construct. In another aspect, the dimensional reduction might decrease the performance since we will not maintain all the variance of the data. In model selection, we will consider the tradeoff between the performance and running time.

We decided to use PCA as the algorithm to reduce the dimension of our data set. By using the function of PCA in sklearn, we can see the ratio of variance of each principal component. We decided to keep 90% of the variance. We got the top 12 principal components that can keep 90% of variance and we make a new data frame of the data after applying PCA as follows.

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	NObeyesdad
0	0.718496	0.991003	-0.445798	-0.423355	1.028903	-0.331062	-0.371232	0.246084	-0.125179	-0.284029	0.198707	-0.299375	Normal_Weight
1	0.890330	-0.188358	0.106929	0.071706	1.628798	-0.114107	0.710391	-0.207825	1.024193	0.404792	0.144820	0.252448	Normal_Weight
2	-0.548466	0.483278	0.451214	-0.411056	1.208940	-0.188385	-0.031003	0.360534	-0.092669	-0.085114	-0.018683	0.217095	Normal_Weight
3	-0.325803	0.830073	1.141759	0.754686	0.878313	-0.773872	0.211537	-0.107875	-0.332604	0.163632	-0.479624	0.093028	Overweight_Level_I
4	-0.019347	-0.011795	1.470648	0.211241	0.727958	-0.870064	-0.589020	-0.317891	-0.071458	-0.439348	-0.216259	0.005847	Overweight_Level_II
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2105	0.581137	-0.650855	-0.419165	-0.240143	0.063008	0.000984	0.201758	0.089077	-0.147368	0.328516	-0.096508	0.155164	Obesity_Type_III
2106	0.589239	-0.648720	-0.420600	-0.237813	0.055644	-0.001371	0.167273	0.101459	-0.138577	0.335972	-0.100992	0.152246	Obesity_Type_III
2107	0.574357	-0.675829	-0.430890	-0.224510	0.077922	0.013165	0.177602	-0.071669	-0.141773	0.255449	-0.079206	0.043088	Obesity_Type_III
2108	0.569800	-0.674926	-0.430160	-0.228890	0.082297	0.013154	0.212063	-0.058286	-0.148463	0.247699	-0.081164	0.052358	Obesity_Type_III
2110	0.555713	-0.705243	-0.447400	-0.255121	0.127186	0.024592	0.423261	-0.139836	-0.159140	-0.031917	-0.104090	-0.143493	Obesity_Type_III

2011 rows × 13 columns

3. Modeling approach

3.1 Logistic Regression

Our first purpose is to use some features to predict whether a person has obesity or not. This problem should be a binary classification problem and we need to separate the target feature “NObeyesdad " into two parts. One part should be “obesity” including all the obesity types as 1(positive class) and “no obesity” including normal weight and insufficient weight as 0(negative class). The model we choose to apply first is logistic regression since logistic regression is a powerful model in binary classification. In addition, by using logistic regression we can see the importance of each feature. This can help us figure out which factors affect obese people the most and help them change their lifestyle. We used the data set after applying the outlier detection method. Before applying logistic regression, we split the dataset into training(70%) and testing(30%) set by using train_test_split in the sklearn package. We also used the GridSearchCV tool in sklearn to tune the hyperparameters. The hyperparameters we tune are the regularization

strength ‘C’=[0.001,0.01,0.1,1], regularization type {‘l1’,’l2’}, and solver {'lbfgs', 'liblinear', 'newton-cg', 'newton-cholesky', 'sag', 'saga'}. By the result GridSearch gave us, we choose to use ‘C’ = 1, regularization type = ‘l1’, and solver = 'liblinear'.
By applying logistic regression, we have the following testing result.

```
Accuracy: 0.9810725552050473
F1 score: 0.987152034261242
Recall: 0.9913978494623656
Precision: 0.9829424307036247

 clasification report:
              precision    recall  f1-score   support

           0       0.98      0.95      0.96       169
           1       0.98      0.99      0.99       465

    accuracy                           0.98       634
   macro avg       0.98      0.97      0.98       634
weighted avg       0.98      0.98      0.98       634


 confussion matrix:
 [[161   8]
 [  4 461]]
```

Figure 6: Evaluation matrix

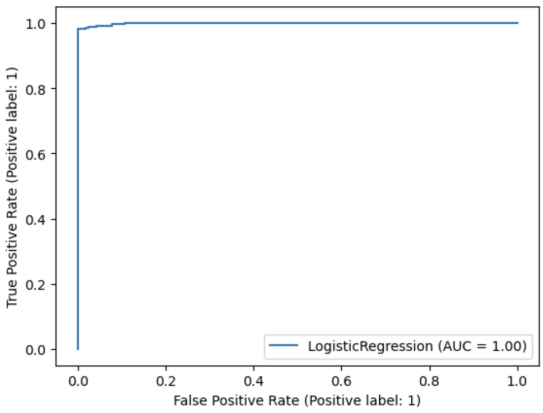


Figure 7: ROC

We can see the accuracy of this model is around 98% and the area under the curve is 1.0 which is really good. Besides this, we can see the weight of each feature. We can say Logistic regression is strong enough to predict the binary case of this problem. By checking the absolute value of each weight, we have the largest top 2 absolute weight features: “Weight” and “Height” which make sense since these are the main factors to determine if a person is overweight or not. See figure 8 as the whole weight list of features.

Weight	50.100529
Height	-14.669091
CAEC_Always	-1.578271
CAEC_Frequently	-1.448574
SCC_no	-1.166828
MTRANS_Walking	-0.971899
FCVC	-0.709010
CALC_no	-0.658451
family_history_with_overweight_no	-0.607338
Gender_Male	-0.553988
FAVC_no	-0.525710
SMOKE_yes	-0.505212
FAF	-0.376239
NCP	-0.332370
MTRANS_Automobile	0.233425

MTRANS_Motorbike	-0.209183
TUE	-0.010380
CALC_Always	0.000000
MTRANS_Public_Transportation	0.000000
MTRANS_Bike	0.000000
CALC_Sometimes	0.000000
CALC_Frequently	0.000000
SMOKE_no	0.000000
SCC_yes	0.000000
CAEC_no	0.000000
CAEC_Sometimes	0.000000
CH2O	0.000000
FAVC_yes	0.000000
family_history_with_overweight_yes	0.000000
Gender_Female	0.000000
Age	0.000000

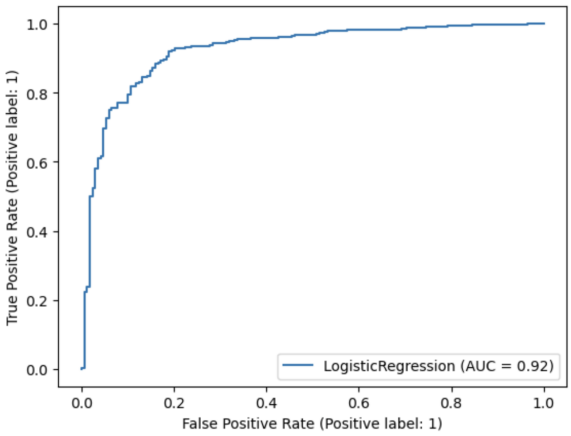
Figure 8: Feature weight

By the result we got before, we can find that terms “weight” and “height”have much larger weight than other terms. Therefore, we want to drop these terms to build a new model and this model can be interpreted as predicting obesity type only by using people’s lifestyle. This model can solve some decisions to be impacted we mentioned at the beginning of this report. We set the

same hyperparameter set for GridSearch and we also got C = 1,penalty = 'l1', solver = 'liblinear'. By using these hyperparameters, we got the following result.

```
Accuracy: 0.889589905362776
F1 score: 0.9250535331905783
Recall: 0.9290322580645162
Precision: 0.9211087420042644

 clasification report:
              precision    recall  f1-score   support

           0       0.80      0.78      0.79       169
           1       0.92      0.93      0.93       465

    accuracy                           0.89       634
   macro avg       0.86      0.86      0.86       634
weighted avg       0.89      0.89      0.89       634


 confussion matrix:
 [[132  37]
 [ 33 432]]
```

Evaluation matrix



ROC

From the result, we can find there is an obvious decrease of the precision between the original model and the new model. Therefore, we might need to consider some complexer models to make predictions by dropping “weight” and “height”.

We also apply Logistic regression on multiclass classification under both cases, drop top 3 features and not drop. By using the same method of GridSearch, we get C = 1, Regularization type = ‘l1’, and solver = ‘saga’ for not drop 3 features; C = 1, Regularization type = 'l1', solver = 'liblinear' for drop 3 features.
By training the model, we got the following result separately.

```
Accuracy: 0.8359621451104101
F1 score: 0.8285673836145673
Recall: 0.832321315499317
Precision: 0.830271613413605

 clasification report:
              precision    recall  f1-score   support

           1       0.76      0.64      0.70        89
           2       0.71      0.73      0.72        85
           3       0.82      0.82      0.82       114
           4       0.88      1.00      0.93        85
           5       0.97      0.99      0.98        92
           6       0.80      0.66      0.72        77
           7       0.88      0.99      0.93        92

    accuracy                           0.84       634
   macro avg       0.83      0.83      0.83       634
weighted avg       0.83      0.84      0.83       634


 confussion matrix:
 [[57 17  4  0  0 11  0]
 [ 7 62 15  0  0  1  0]
 [ 1  5 93 12  3  0  0]
 [ 0  0  0 85  0  0  0]
 [ 0  0  1  0 91  0  0]
 [10  3  0  0  0 51 13]
 [ 0  0  0  0  0  1 91]]
```

Multiclass classification with all features

```
Accuracy: 0.5993690851735016
F1 score: 0.5736848551920846
Recall: 0.5912485104928952
Precision: 0.5907041541123775

 clasification report:
              precision    recall  f1-score   support

           1       0.66      0.39      0.49        89
           2       0.45      0.20      0.28        85
           3       0.50      0.62      0.55       114
           4       0.56      0.84      0.67        85
           5       0.90      0.99      0.94        92
           6       0.45      0.40      0.42        77
           7       0.62      0.70      0.66        92

    accuracy                           0.60       634
   macro avg       0.59      0.59      0.57       634
weighted avg       0.59      0.60      0.58       634


 confussion matrix:
 [[35  6 12 15  2 15  4]
 [ 3 17 37 18  2  4  4]
 [ 3  7 71 18  4  5  6]
 [ 0  0 13 71  0  1  0]
 [ 0  1  0  0 91  0  0]
 [ 7  6  4  2  2 31 25]
 [ 5  1  6  3  0 13 64]]
```

Multiclass classification with drop 3 features

By the result, we can see that both models did not have such good results as the binary classification. If we drop ‘weight’ and ‘height’, we can find the model accuracy is 0.59 which is not good for predicting. Therefore, for the multiclass classification, we need to use more complex models on predicting them.

3.2 K-Nearest Neighbors

The second model we used is Knn. Knn is a simple but widely used algorithm in machine learning, especially in classification problems. There exists two important hyperparameters in Knn which are the “n_neighbors” and the ‘metric’. We used grid search to choose the best “n_neighbors” from the range 1 to 10 and ‘metric’ in ['euclidean', 'manhattan', 'minkowski']. We also build Knn on both binary and multiclass classification cases. After that, we apply Knn on the case that drops the top three weight features “weight” and “height”.

For both binary classification and multivariate classification , the GridSearch gave us the best hyperparameter pair is “n_neighbors = 1” and “metric = manhattan” for both cases and we got the following results,

```
Accuracy: 0.9258675078864353
F1 score: 0.9258675078864353
Recall: 0.9258675078864353
Precision: 0.9258675078864353

 clasification report:
               precision    recall  f1-score   support

           0       0.86      0.86      0.86       169
           1       0.95      0.95      0.95       465

    accuracy                           0.93       634
   macro avg       0.90      0.91      0.91       634
weighted avg       0.93      0.93      0.93       634


 confussion matrix:
 [[146  23]
 [ 24 441]]
```

Binary classification

```
Accuracy: 0.8422712933753943
F1 score: 0.8351673357158056
Recall: 0.837082914638801
Precision: 0.8353215016302183

 clasification report:
               precision    recall  f1-score   support

           1       0.81      0.72      0.76        89
           2       0.76      0.82      0.79        85
           3       0.87      0.85      0.86       114
           4       0.95      0.99      0.97        85
           5       0.98      0.99      0.98        92
           6       0.66      0.60      0.63        77
           7       0.82      0.89      0.85        92

    accuracy                           0.84       634
   macro avg       0.84      0.84      0.84       634
weighted avg       0.84      0.84      0.84       634


 confussion matrix:
 [[64  6  4  0  0 12  3]
 [ 6 70  5  1  1  2  0]
 [ 2  6 97  1  1  3  4]
 [ 0  1  0 84  0  0  0]
 [ 0  1  0  0 91  0  0]
 [ 7  8  4  1  0 46 11]
 [ 0  0  2  1  0  7 82]]
```

Multi-class classification

We can find the binary classification performance is worse than Logistic regression but have a pretty good performance overall. However, for multivariable classification, some variables have high precision like 5:Obesity_Type_III and some variables have low precision like 6:Normal_Weight. This phenomenon leads to a not good overall performance.

We then apply Knn on the data set that drops “weight” and “height”. The GridSearch gave us the best hyperparameter pair for binary classification “n_neighbors = 5”, “metric = manhattan” and for multiclass classification “n_neighbors = 1”, “metric = manhattan”. By using the best hyperparameters, we got the following results.

```
Accuracy: 0.9221854304635762
F1 score: 0.8982388597934523
Recall: 0.8911036036036035
Precision: 0.9061810154525387

 clasification report:
               precision    recall  f1-score   support

           0       0.87      0.82      0.85       160
           1       0.94      0.96      0.95       444

    accuracy                           0.92       604
   macro avg       0.91      0.89      0.90       604
weighted avg       0.92      0.92      0.92       604


 confussion matrix:
 [[132  28]
 [ 19 425]]
```

Binary classification with drop 3 features

```
Accuracy: 0.793046357615894
F1 score: 0.7821561820354768
Recall: 0.7838540335642319
Precision: 0.7825580893639801

 clasification report:
               precision    recall  f1-score   support

           1       0.74      0.64      0.69        76
           2       0.74      0.71      0.73        77
           3       0.77      0.76      0.76       100
           4       0.83      0.91      0.87        93
           5       0.97      1.00      0.98        98
           6       0.60      0.57      0.58        81
           7       0.82      0.89      0.85        79

    accuracy                           0.79       604
   macro avg       0.78      0.78      0.78       604
weighted avg       0.79      0.79      0.79       604


 confussion matrix:
 [[49  3  6  4  1 11  2]
 [ 6 55  4  6  1  5  0]
 [ 3  6 76  4  1  7  3]
 [ 0  1  5 85  0  1  1]
 [ 0  0  0  0 98  0  0]
 [ 8  9  6  3  0 46  9]
 [ 0  0  2  0  0  7 70]]
```

Multiclass classification with drop 3 features

From the result, we find both cases have a decreased accuracy on prediction which makes sense since we dropped some informative features. However, the overall accuracy did not have such a large difference like Logistic regression. We think this phenomenon is because Knn is dependent on distance and Logistic regression is dependent on regression.

3.3 Random Forest Tree

The third model we employed is the Random Forest Tree algorithm. Random Forest Tree is a learning-based approach that incorporates various decision trees to construct a “forest” and generate predictions. Specifically, each decision tree in the “forest” will make their own predictions based on a subset of data and features, and the “forest” will choose the most common outcomes among all trees to determine the final decision. Like KNN, Random Forest Tree can be used for both binary classification and multi-class classification. Compared to KNN, Random Forest Tree is more robust in handling the larger datasets and non-linear relationships. However, Random Forest Tree needs more computational resources and training time due to the construction of several decision trees.

To determine if an individual is obese and identify the specific obesity type they may have, we utilize the Random Forest Tree for both binary and multi-class classification tasks. The results for each task are shown in the figure below:

```
Accuracy: 0.9700315457413249
F1 score: 0.9795479009687836
Recall: 0.978494623655914
Precision: 0.9806034482758621

 clasification report:
               precision    recall  f1-score   support

           0       0.94      0.95      0.94       169
           1       0.98      0.98      0.98       465

    accuracy                           0.97       634
   macro avg       0.96      0.96      0.96       634
weighted avg       0.97      0.97      0.97       634


 confussion matrix:
 [[160   9]
 [ 10 455]]
```

(Binary Classification)

```
Accuracy: 0.9258675078864353
F1 score: 0.9234036905903121
Recall: 0.9257840540235982
Precision: 0.9267647472384187

 clasification report:
               precision    recall  f1-score   support

           1       0.95      0.78      0.85        89
           2       0.86      0.93      0.89        85
           3       0.99      0.92      0.95       114
           4       1.00      1.00      1.00        85
           5       1.00      1.00      1.00        92
           6       0.74      0.91      0.81        77
           7       0.96      0.95      0.95        92

    accuracy                           0.93       634
   macro avg       0.93      0.93      0.92       634
weighted avg       0.93      0.93      0.93       634


 confussion matrix:
 [[ 69   4   0   0   0  16   0]
 [  3  79   1   0   0   2   0]
 [  1   6 105   0   0   2   0]
 [  0   0   0  85   0   0   0]
 [  0   0   0   0  92   0   0]
 [  0   3   0   0   0  70   4]
 [  0   0   0   0   0   5  87]]
```

(Multi-class Classification)

For now, we have not done any hyperparameter tuning yet. In other words, we rely on the default settings of scikit-learn package's Random Forest Classifier to make predictions. Compared to KNN and Logistic Regression, we can see that Random Forest Tree performs better in both classification tasks even with the default settings. However, we can still observe that Random Forest tree does not perform well in classifying "6: Normal Weight" category in the multi-class classification. In light of this, we need to dig into the samples that it wrongly classified to get more information.

We also perform the Random Forest Tree algorithm on the dataset that is free from outliers. The result for both classification task have been shown below:

```
Accuracy: 0.9735099337748344
F1 score: 0.9820627802690582
Recall: 0.9864864864864865
Precision: 0.9776785714285714

 clasification report:
               precision    recall  f1-score   support

           0       0.96      0.94      0.95       160
           1       0.98      0.99      0.98       444

    accuracy                           0.97       604
   macro avg       0.97      0.96      0.97       604
weighted avg       0.97      0.97      0.97       604


 confussion matrix:
 [[150  10]
 [  6 438]]
```

(Binary Classification)

```
Accuracy: 0.9420529801324503
F1 score: 0.9384146004837258
Recall: 0.9387776418489481
Precision: 0.9389362411613318

 clasification report:
               precision    recall  f1-score   support

           1       0.92      0.86      0.88        76
           2       0.90      0.95      0.92        77
           3       0.99      0.94      0.96       100
           4       1.00      0.99      0.99        93
           5       0.99      1.00      0.99        98
           6       0.81      0.86      0.84        81
           7       0.96      0.97      0.97        79

    accuracy                           0.94       604
   macro avg       0.94      0.94      0.94       604
weighted avg       0.94      0.94      0.94       604


 confussion matrix:
 [[65  2  1  0  0  8  0]
 [ 0 73  0  0  0  4  0]
 [ 2  2 94  0  0  2  0]
 [ 0  0  0 92  1  0  0]
 [ 0  0  0  0 98  0  0]
 [ 4  4  0  0  0 70  3]
 [ 0  0  0  0  0  2 77]]
```

(Multi-class Classification)

Based on the graphs above, we can see that the outliers really have some negative effect on the performance of the Random Forest Tree and cleaning them up really improves the accuracy of the classifier.

Recall that Height and Weight are the two most important features for predicting obesity, we also drop these two features to see how the rest of the features interact with the prediction of obesity. The results of the prediction without Height and Weight have shown in the graphs below:

```
Accuracy: 0.9072847682119205
F1 score: 0.9370786516853933
Recall: 0.9391891891891891
Precision: 0.9349775784753364

 clasification report:
               precision    recall  f1-score   support

           0       0.83      0.82      0.82       160
           1       0.93      0.94      0.94       444

    accuracy                           0.91       604
   macro avg       0.88      0.88      0.88       604
weighted avg       0.91      0.91      0.91       604


 confussion matrix:
 [[131  29]
 [ 27 417]]
```

(Binary Classification)

```
Accuracy: 0.8079470198675497
F1 score: 0.8002110122444307
Recall: 0.8002889331465978
Precision: 0.803746109049475

 clasification report:
               precision    recall  f1-score   support

           1       0.77      0.67      0.72        76
           2       0.71      0.64      0.67        77
           3       0.79      0.75      0.77       100
           4       0.87      0.89      0.88        93
           5       0.99      1.00      0.99        98
           6       0.61      0.74      0.67        81
           7       0.88      0.91      0.89        79

    accuracy                           0.81       604
   macro avg       0.80      0.80      0.80       604
weighted avg       0.81      0.81      0.81       604


 confussion matrix:
 [[51  3  6  2  0 12  2]
 [ 5 49  8  5  0 10  0]
 [ 4  6 75  5  1  7  2]
 [ 1  4  2 83  0  3  0]
 [ 0  0  0  0 98  0  0]
 [ 5  7  3  0  0 60  6]
 [ 0  0  1  0  0  6 72]]
```

(Multi-class Classification)

Obviously, we observe an apparent decrease in the performance of Random Forest Classifiers. However, we can notice that the performance of classifying “5: Obesity Type III” is still impressive with absence of Height and Weight. From this perspective, we may conclude that the rest of the features still play important roles in the cause of Obesity Type III.

3.4 Model Decision & Further Testing

For now, we have tested three models: logistic regression, k-nearest neighbors and random forest trees. Based on the performance we’ve shown above through different cases, we have determined that the Random Forest Tree will be our model of choice for determining final performance, for it outperforms the logistic regression and k-nearest neighbors in both binary and multi-class classifications. With regard to this, the first thing we need to do is to fine tune the hyperparameters of random forest models. The hyperparameters we choose to update are: n_estimators–the number of trees in the forest, max_depth–the maximum depth of the tree, min_samples_split–the minimum number of samples required to split an internal node, and min_samples_leaf–the minimum number of samples required to be at a leaf node. Finally, we get the best groups of hyperparameters, which are shown in the graphs below:

```
{'max_depth': 20,
 'min_samples_leaf': 2,
 'min_samples_split': 2,
 'n_estimators': 200}
```

(Binary classification)

```
{'max_depth': 20,
 'min_samples_leaf': 1,
 'min_samples_split': 5,
 'n_estimators': 100}
```

(Multi-class classification)

The results of binary and multi-class classifications after fine tuning the hyperparameters are shown below:

```
Accuracy: 0.9735099337748344
F1 score: 0.9658540032506537
Recall: 0.9639921171171171
Precision: 0.9677584151671681

 clasification report:
               precision    recall  f1-score   support

           0       0.96      0.94      0.95       160
           1       0.98      0.98      0.98       444

    accuracy                           0.97       604
   macro avg       0.97      0.96      0.97       604
weighted avg       0.97      0.97      0.97       604


 confussion matrix:
 [[151   9]
 [  7 437]]
```

(Binary Classification)

```
Accuracy: 0.9403973509933775
F1 score: 0.9363850876881348
Recall: 0.936471226459826
Precision: 0.9374619343166319

 clasification report:
               precision    recall  f1-score   support

           1       0.93      0.84      0.88        76
           2       0.90      0.94      0.92        77
           3       0.99      0.95      0.97       100
           4       1.00      0.99      0.99        93
           5       0.99      1.00      0.99        98
           6       0.80      0.86      0.83        81
           7       0.95      0.97      0.96        79

    accuracy                           0.94       604
   macro avg       0.94      0.94      0.94       604
weighted avg       0.94      0.94      0.94       604


 confussion matrix:
 [[64  4  1  0  0  7  0]
 [ 0 72  0  0  0  5  0]
 [ 2  0 95  0  0  3  0]
 [ 0  0  0 92  1  0  0]
 [ 0  0  0  0 98  0  0]
 [ 3  4  0  0  0 70  4]
 [ 0  0  0  0  0  2 77]]
```

(Multi-class Classification)

Through the results, we can see that there is not much difference between the default random forest models and fine-tuned random forest models. I think one main reason is that the default one has already shown very impressive results, with 97.35% accuracy for binary classification and 94.21% of accuracy for multi-class classification. In this case, there is less room to improve its performance by just modifying the hyperparameters. Another possible reason is that we only choose a small set of possible hyperparameters for the model to choose. Due to limited computational resources, we only test three values for each of the hyperparameters.

Recall that Weight and Height are the primary factors of determining obesity, we also fine tune the hyperparameters of the Random Forest to predict the obesity without these two features. The results have shown below:

```
Accuracy: 0.9205298013245033
F1 score: 0.8987766046589576
Recall: 0.9019707207207208
Precision: 0.8957317073170732

 clasification report:
               precision    recall  f1-score   support

           0       0.84      0.86      0.85       160
           1       0.95      0.94      0.95       444

    accuracy                           0.92       604
   macro avg       0.90      0.90      0.90       604
weighted avg       0.92      0.92      0.92       604


 confussion matrix:
 [[138  22]
 [ 26 418]]
```

(Binary Classification)

```
Accuracy: 0.859271523178808
F1 score: 0.85445191518382
Recall: 0.8527466808934566
Precision: 0.8575478538503148

 clasification report:
              precision    recall  f1-score   support

           1       0.85      0.76      0.81        76
           2       0.81      0.77      0.79        77
           3       0.83      0.85      0.84       100
           4       0.91      0.92      0.92        93
           5       0.99      1.00      0.99        98
           6       0.66      0.73      0.69        81
           7       0.95      0.94      0.94        79

    accuracy                           0.86       604
   macro avg       0.86      0.85      0.85       604
weighted avg       0.86      0.86      0.86       604


 confussion matrix:
 [[58  1  5  1  0 11  0]
 [ 1 59  6  5  0  6  0]
 [ 4  3 85  1  1  6  0]
 [ 0  3  1 86  0  3  0]
 [ 0  0  0  0 98  0  0]
 [ 5  7  5  1  0 59  4]
 [ 0  0  1  0  0  4 74]]
```

(Multi-class Classification)

Though there is no significant improvement in binary classification, there is some improvement in the multi-class classification compared to the default Random Forest model. However, we can still see that the performance of predicting category 6 “Normal Weight” is still poor after fine tuning the hyperparameters. Based on the confusion matrix, we can see that there may be some ambiguity between category 1 “Overweight Level I” and category 6 “Normal Weight.” I think one possible reason to explain this phenomenon is that there are some samples that lie on the boundary between normal weight and overweight, which does not show a clear pattern for the Random Forest model to classify such differences during the prediction, resulting in a low prediction accuracy.

After that, we test the performance of the Random Forest model solely based on eating habit features or physical conditions. The goal of this test is to see which type of features contribute more to the cause of obesity. The results of binary classification with respect to each type of features have shown below:

```
Accuracy: 0.8642384105960265
F1 score: 0.9076576576576577
Recall: 0.9076576576576577
Precision: 0.9076576576576577

 clasification report:
              precision    recall  f1-score   support

           0       0.74      0.74      0.74       160
           1       0.91      0.91      0.91       444

    accuracy                           0.86       604
   macro avg       0.83      0.83      0.83       604
weighted avg       0.86      0.86      0.86       604


 confussion matrix:
 [[119  41]
 [ 41 403]]
```

(Eating Habits)

```
Accuracy: 0.8178807947019867
F1 score: 0.8806941431670282
Recall: 0.9144144144144144
Precision: 0.8493723849372385

 clasification report:
              precision    recall  f1-score   support

           0       0.70      0.55      0.62       160
           1       0.85      0.91      0.88       444

    accuracy                           0.82       604
   macro avg       0.77      0.73      0.75       604
weighted avg       0.81      0.82      0.81       604


 confussion matrix:
 [[ 88  72]
 [ 38 406]]
```

(Physical Conditions)

Through the results, we can see that eating habits outperforms physical conditions, meaning that eating habits is the more significant one that causes obesity. Looking at the confusion matrix of physical condition features, we can see that there are many false positive samples during the prediction. Such a phenomenon represents that some samples are classified as obese but actually they are not, implying unhealthy physical conditions does not explicitly lead to obesity.

Similar results occur in the multi-class classification, which have been shown below:

```
Accuracy: 0.6804635761589404
F1 score: 0.660555692383812
Recall: 0.6662670943865355
Precision: 0.6695171213824311

 clasification report:
              precision    recall  f1-score   support

           1       0.64      0.37      0.47        76
           2       0.45      0.42      0.43        77
           3       0.63      0.62      0.62       100
           4       0.72      0.78      0.75        93
           5       0.96      1.00      0.98        98
           6       0.56      0.74      0.63        81
           7       0.73      0.73      0.73        79

    accuracy                           0.68       604
   macro avg       0.67      0.67      0.66       604
weighted avg       0.68      0.68      0.67       604


 confussion matrix:
 [[28  7 12  8  0 16  5]
 [ 3 32 18  8  0 12  4]
 [ 5 11 62 10  3  7  2]
 [ 3  8  2 73  0  5  2]
 [ 0  0  0  0 98  0  0]
 [ 1  9  2  1  0 60  8]
 [ 4  4  3  1  1  8 58]]
```

(Eating Habits)

```
Accuracy: 0.49503311258278143
F1 score: 0.46774144342139073
Recall: 0.4836934853541615
Precision: 0.46967609813273276

 clasification report:
              precision    recall  f1-score   support

           1       0.44      0.32      0.37        76
           2       0.25      0.18      0.21        77
           3       0.39      0.38      0.39       100
           4       0.56      0.52      0.54        93
           5       0.64      0.85      0.73        98
           6       0.52      0.77      0.62        81
           7       0.49      0.38      0.43        79

    accuracy                           0.50       604
   macro avg       0.47      0.48      0.47       604
weighted avg       0.48      0.50      0.48       604


 confussion matrix:
 [[24  6 10 18  4 11  3]
 [ 4 14 21  9 12 10  7]
 [11 12 38  4 11 15  9]
 [ 4 13 12 48  9  3  4]
 [ 0  2  1  4 83  1  7]
 [ 9  4  5  0  0 62  1]
 [ 3  4 10  3 11 18 30]]
```

(Physical Condition)

Through the accuracy scores, we can see that both eating habits and physical conditions show poor performance in multi-class classification. This situation shows that the obesity level does not solely depend on one type of feature but a combination of them. From the confusion matrix of both types of features, we can see that there is ambiguity in classifying category 1, 2, 3, 4, representing the overweight level I, II and obesity level I, II. However, we can still conclude that eating habits perform comparatively better than physical condition since eating habits contain fewer off-diagonal elements.

Finally, we run the Random Forest on the principal components we got in the previous sections. The results for binary and multi-class classification have shown below:

```
Accuracy: 0.9089403973509934
F1 score: 0.9384098544232924
Recall: 0.9436936936936937
Precision: 0.933184855233853

 clasification report:
              precision    recall  f1-score   support

           0       0.84      0.81      0.83       160
           1       0.93      0.94      0.94       444

    accuracy                           0.91       604
   macro avg       0.89      0.88      0.88       604
weighted avg       0.91      0.91      0.91       604


 confussion matrix:
 [[130  30]
 [ 25 419]]
```

(Binary Classification)

```
Accuracy: 0.7947019867549668
F1 score: 0.7854501125083121
Recall: 0.7851216063492262
Precision: 0.7871274218981393

 clasification report:
               precision    recall  f1-score   support

           1       0.72      0.63      0.67        76
           2       0.72      0.69      0.70        77
           3       0.76      0.77      0.77       100
           4       0.84      0.90      0.87        93
           5       0.98      1.00      0.99        98
           6       0.61      0.64      0.63        81
           7       0.88      0.86      0.87        79

    accuracy                           0.79       604
   macro avg       0.79      0.79      0.79       604
weighted avg       0.79      0.79      0.79       604


 confussion matrix:
 [[48  2  8  3  0 11  4]
 [ 5 53  5  7  1  6  0]
 [ 3  9 77  4  1  6  0]
 [ 1  3  4 84  0  1  0]
 [ 0  0  0  0 98  0  0]
 [10  7  5  2  0 52  5]
 [ 0  0  2  0  0  9 68]]
```

(Multi-class Classification)

Based on the results above, we can see that both binary and multi-class classification perform worse compared to the performance of the raw dataset. Specifically, the binary accuracy score decreased for about 0.07 and the multi-class accuracy store decreased for about 0.15. Such accuracy reduction represents that keeping 90% of variance may be available for predicting obesity but not work for predicting specific obesity levels. Such specification requires us to take different variance with respect to each level into account instead of the general variance among the dataset.

3.4 Machine Learning Workflow

The first thing we've done for the workflow is the data checking. Specifically, we acquired a dataset from UCI Machine Learning Repository, where the input features are elements of space

X and the target variables are elements of space **Y**. Then, examined the dataset to see if there are any missing values and found no absent entries. If it does, we will perform a function ***f*** to handle these missing values,

$$\forall x \in X : x = \begin{cases} x, & \text{if } x \neq \text{null} \\ f(X), & \text{otherwise} \end{cases}$$

where ***x*** is the data point in **X** and ***f(X)*** is the function we apply to fulfill the missing values

Then, we perform descriptive analytics on the dataset by visualizing the distribution of each feature. In particular, we employ bar plots and histogram for numerical features and pie charts for categorical features. Besides, we also draw the correlation matrix to determine whether there are internal relationships among different variables.

After that, we applied one-hot-encoding on the categorical features to prepare them for our model analysis. Moreover, we normalized all the data values between 0 and 1, which is convenient for our potential linear model applications. Mathematically, suppose ***c*** represents each column of **X** and ***D*** represent the domain, we perform one-hot encoding like this:

$$encode(c) = D \in [0, 1]$$

For now, we choose the Random Forest Model for our obesity level estimation. The reason is that it shows impressive performance compared to other models. Then, we randomly split the dataset into a training set and a test set, allocating 70% of the data for training and 30% of the data for testing. In MLM form, we can express our steps like this:

- The input space is $\mathbf{X} = \mathbf{R}^m$ and the output space is $\mathbf{Y} = \mathbf{R}$
- Since we do not embed prior knowledge for now, the error prior $P = 1$
- The corresponding learning Morphism is complex for random forest tree, since it does not have traditional tranable weight parameters like logistic regressions
    - What we are doing is to specify a Random Forest model with N trees. Each tree t_i is trained on the training data
    - A random subset of features is selected by each node of each tree
- The risk function is the entropy loss measure the uncertainty between the true label and test prediction

  $$L(p_i) = -\sum_{i=1}^{C} p_i \log_2(p_i)$$

  where p_i represents the proportion of samples that belongs to a specific class i and C is the total number of classes.

- Then, the MLM can be written as

$$ML_{RFT} = (\mathbb{R}^m, \mathbb{R}, P = 1, L(p_i))$$

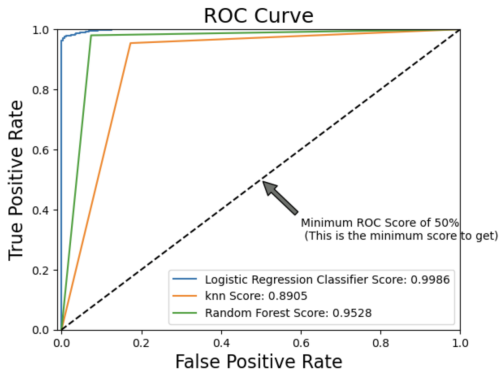
3.5 Cross Validation and ROC, AUC

We used K-fold cross validation as the tool for model selection. We choose to use 5-folds to apply the cross validation. Since our data is balanced, we choose to use the mean of accuracy as the evaluate metric to measure the goodness of our model. The 5-folds cross validation will randomly separate the training set into 5 parts and use the same hyperparameters to train the model 5 times (each time for a part of the data), and test the result on the validation set. We will finally get 5 values of accuracy. The final cross validation score will be the mean of 5 values of accuracy. Note that since our data set is not fully balanced for multiclass classification and unbalanced for binary classification, we choose to use 'balanced_accuracy' as the evaluation metric to measure our model.

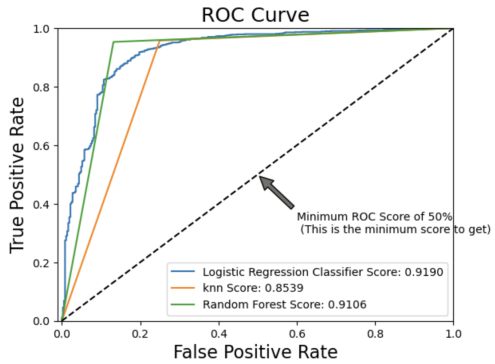
We apply cross validation on four cases: binary classification, multiclass classification, binary classification with drop features, and multiclass classification with drop features. The result table we got as following

	Binary Classification	Multiclass Classification	Binary Classification with drop features	Multiclass Classification with drop featues
Logistic Regression	0.965964	0.844227	0.838917	0.601506
K-Nearest Neighbors	0.890422	0.832634	0.853882	0.752031
Random Forest	0.960284	0.939172	0.900151	0.823473
PCA Random Forest	0.882971	0.777066	None	None

We also draw the ROC curve for our binary classification as following



Binary classification without drop feature



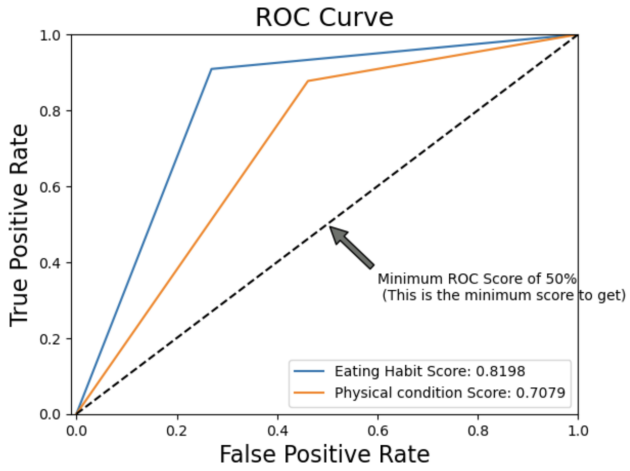
Binary classification with drop feature

Note: The score of each model is AUC value.

For the Random Forest model, we also apply cross validation on the eating habits features prediction and physical conditions features prediction in two cases: binary and multi-class classification. The result table has shown below:

	Binary Classification	Multiclass Classification
Eating Habit	0.819198	0.657232
Physical Condition	0.718252	0.475376

The corresponding ROC curve is also shown below:



As we mentioned in the previous section that eating habits features contribute more in causing obesity compared to physical condition features, such cross validation results match our conclusion, where the AUC value is bigger for eating habits than that of physical conditions, representing a higher level of performance for classification tasks.

4. Result and insights

From the above figures and evaluated metrics, we concluded the following result and insights:

Result and Insights 1:

From the result we got, we found that for binary classification without drop ‘weight’ and ‘height’, Logistic regression has a 0.96 cross validation score which is large enough. We believe this condition is because of the terms ‘weight’, and ‘height’. These two terms account for most part of the weights in Logistic regression. This makes sense since we have BMI which depend on weight and height to measure whether people are obese or not. Therefore, we believe if we want to determine if a person is obese or not, it is enough to use Logistic Regression as the model to predict.

Result and Insights 2:

For the original data multiclass classification, Random Forest has the best performance among these three models with a 0.94 score on cross validation. To classify a person's type of obesity, we believe Random Forest should be the best model. When we drop features “weight” and

“height”, we can see the model accuracy has an obvious decrease. From the cross validation result, we can see Random Forest has a relatively better performance on both binary and multiclass classification among three algorithms. Therefore, we believe Random Forest should be a better choice to predict obesity type by using people’s lifestyle features.

Result and Insights 3:
From evaluating matrices and confusion matrices we got above for all models, we can find for the multiclass classification, the class 6 which is “"Normal_Weight” has a lower accuracy compared to other classes. We think the reason might be that a normal weight class could be relatively harder to classify according to the people’s lifestyle. There also could be because of the data quality of the original dataset or the not enough data of the dataset.

5. Conclusions

With regard to the increasing problem of obesity, the primary focus of our project is to assist individuals who experience obese by managing and reducing their weights, as well as to prevent the people at risk of obesity from becoming overweight. Our goal is to translate the information we got from the obesity dataset to various companies in order to propagate our goal. Based on our results, we have concluded that both eating habits and physical conditions are important in the cause of obesity. Such a conclusion is beneficial for the nutrition company when they want to design efficient products for reducing weight. Specifically, these companies should take physical conditions into account so that their products are more acceptable by obese population from a lifestyle perspective. On the other hand, we also observe that eating habits play a more important role than physical conditions. Such observation is important for the internet companies when they want to design an application to help obese population monitor their weight. To be specific, it is always a perplexing question: which one is important, the lack of exercise or eating too much. In this case, what we found is valuable for these companies to make preference when making the application–focusing more on the eating habits of the obese individuals than their physical conditions. Last but not least, we also find ambiguity in determining whether a person is in normal weight or overweight level I. Such confusion may make it difficult to determine what kind of people should be considered at the risk of obesity. However, what we find and what we get are just a glimpse of the general problem of obesity around the world. To make our decision more concise and more representative, we still need to keep collecting data of different modalities and updating our model to be more powerful that can be generalized to a wide range of tasks.

6. References

1. Estimation of obesity levels based on eating habits and physical condition . (2019). UCI Machine Learning Repository. https://doi.org/10.24432/C5H31Z.

2. Fabio Mendoza Palechor, Alexis de la Hoz Manotas, Dataset for estimation of obesity levels based on eating habits and physical condition in individuals from Colombia, Peru and Mexico, https://doi.org/10.1016/j.dib.2019.104344.(https://www.sciencedirect.com/science/article/pii/S2352340919306985)
3. World Obesity Federation, World Obesity Atlas 2023. https://data.worldobesity.org/publications/?cat=19
4. Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011. https://scikit-learn.org/stable/