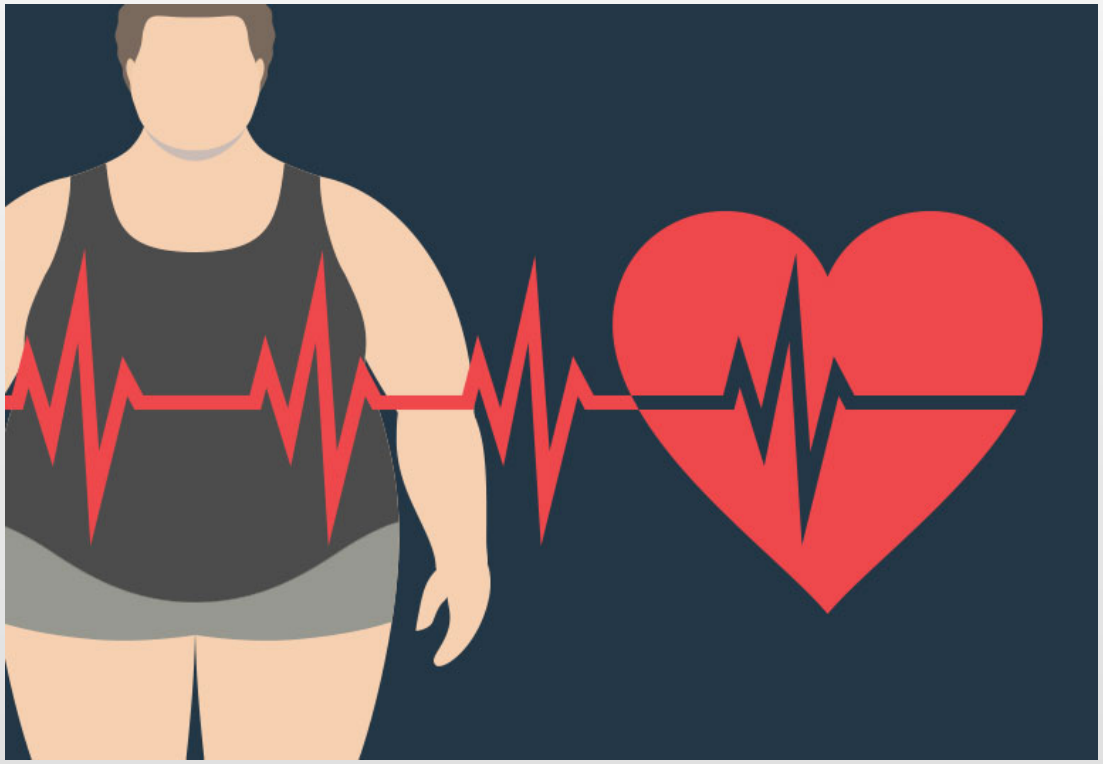
Machine Learning Python

Estimation of obesity levels based on eating habits and physical condition

DIA-3
Huang Tianqi
Guo Zhaojun

The plan

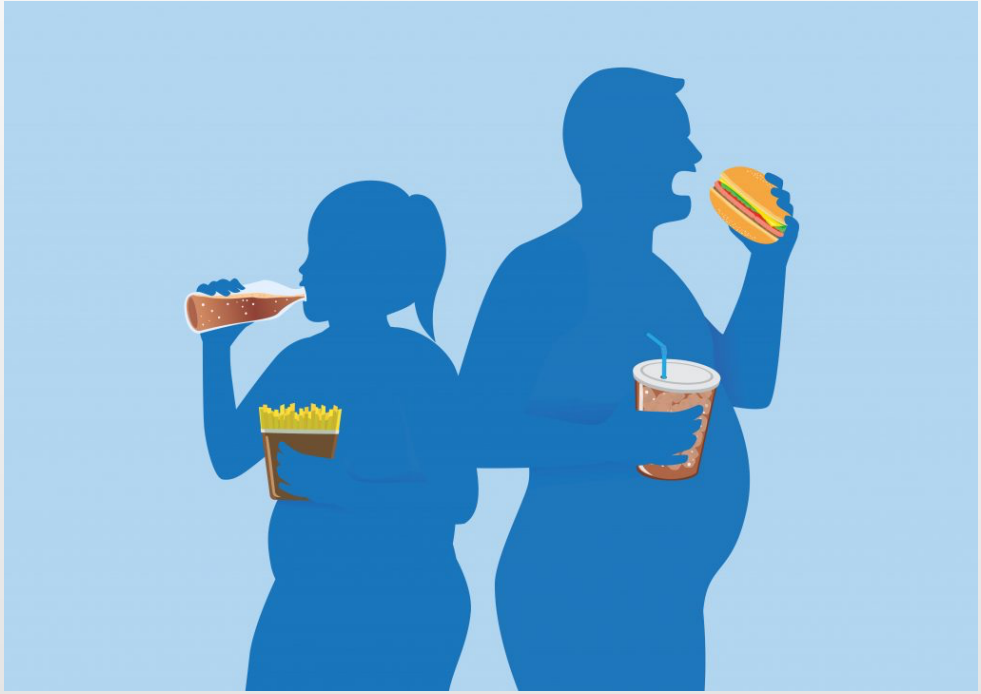
- 1.Introduction of our dataset
- 2.Process(Data visualization,Data preparation for modeling,Modelization and validation)
- 3.Transfer our model to Flask



1.Introduction of our dataset

- Consider the dataset given has many different variantes, mainly those can be divided into float type and category.So we have 2111 given data with 17 different variantes

- 1.Gender
- 2.Age
- 3.Height
- 4.Weight
- 5.family_history_with_overweight
- 6.FAVC(Frequent consumption of high caloric food)
- 7.FCVC(Frequency of consumption of vegetables)
- 8.NCP(Number of main meals)
- 9.CAEC(Consumption of food between meals)
- 10.CH20(Consumption of water daily)
- 11.CALC(Consumption of alcohol)
- 12.SCC(Calories consumption monitoring)
- 13.FAF(Physical activity frequency)
- 14.TUE(Time using technology devices)
- 15.MTRANS(Transportation used)
- 16.SMOKE
- 17.NObeyesdad :obesity levels (The target)



Import Libraires and resources

- We use panda, numpy, sklearn, seaborn, matplotlib in ordre to help us process the dataset

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import  ElasticNet
from sklearn.linear_model import  Ridge
from sklearn.model_selection import cross_val_score
import  statsmodels.formula.api as sm
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import  Lasso
from sklearn.model_selection import GridSearchCV
import matplotlib
from sklearn.decomposition import  PCA
from sklearn.cluster import  KMeans
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import imageio
# models we will be using
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor, plot_tree
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
# model validation techniques
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold
from matplotlib import gridspec
# mse: metric used
from sklearn.metrics import mean_squared_error, make_scorer
```

So we want to do a prediction of obesity levels by using eating habits and physical condition data.
Let’s begin!

Check the dataset

- We check the dataset and find out the dataset have float and category types data, and no missing values in the dataset which is not bad for our study.Also we can see for this test ,average age is very young

]:

	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

```
Gender                            category
Age                                float64
Height                             float64
Weight                             float64
family_history_with_overweight    category
FAVC                              category
FCVC                               float64
NCP                                float64
CAEC                              category
SMOKE                             category
CH2O                               float64
SCC                               category
FAF                                float64
TUE                                float64
CALC                              category
MTRANS                            category
NObeyesdad                        category
dtype: object
```

```
df[categorical_cols].describe()
```

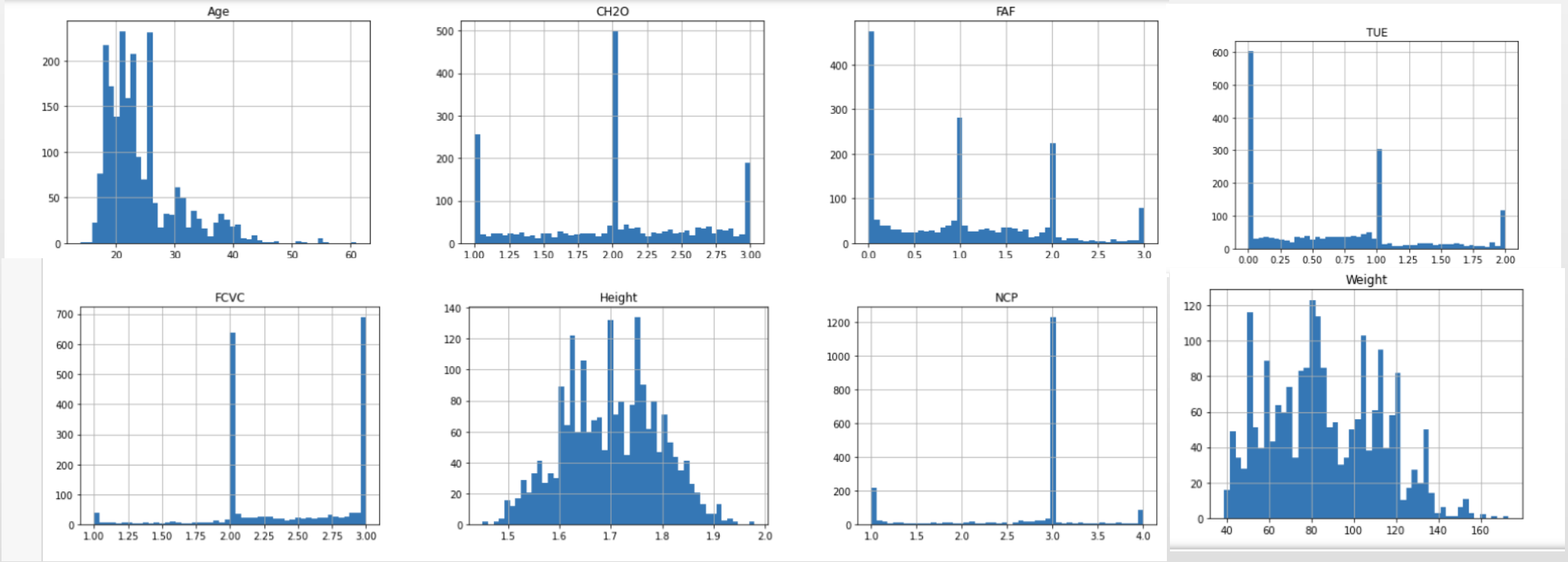
	Gender	family_history_with_overweight	FAVC	CAEC	SMOKE	SCC	CALC	MTRANS	NObeyesdad
count	2111	2111	2111	2111	2111	2111	2111	2111	2111
unique	2	2	2	4	2	2	4	5	7
top	Male	yes	yes	Sometimes	no	no	Sometimes	Public_Transportation	Obesity_Type_I
freq	1068	1726	1866	1765	2067	2015	1401	1580	351

Data visualization

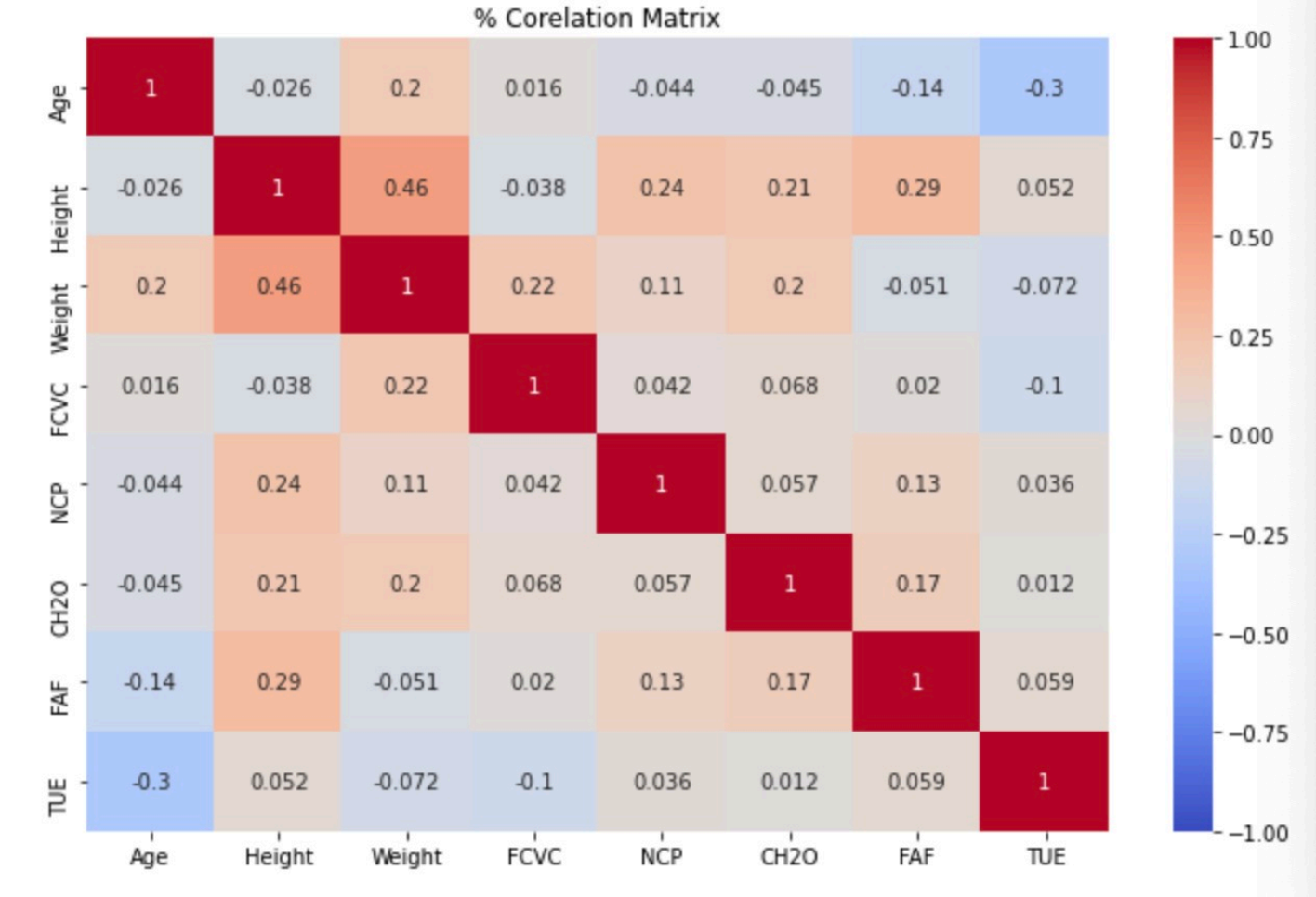
- Histogramme,
- Correlation matrix,
- Univariate analysis,
- Multivariate analysis

Histogramme

- We can see those tester mainly are 20–30 years old, about 1/4 of people rarely do excerises, but most of the people eat a lot vegetables, most people eat 3 meals per day.
- As for time using technology devices is like average ,about a little more than 1/3 seldom use it,1/3 use them ordinary and the other use it a lot.

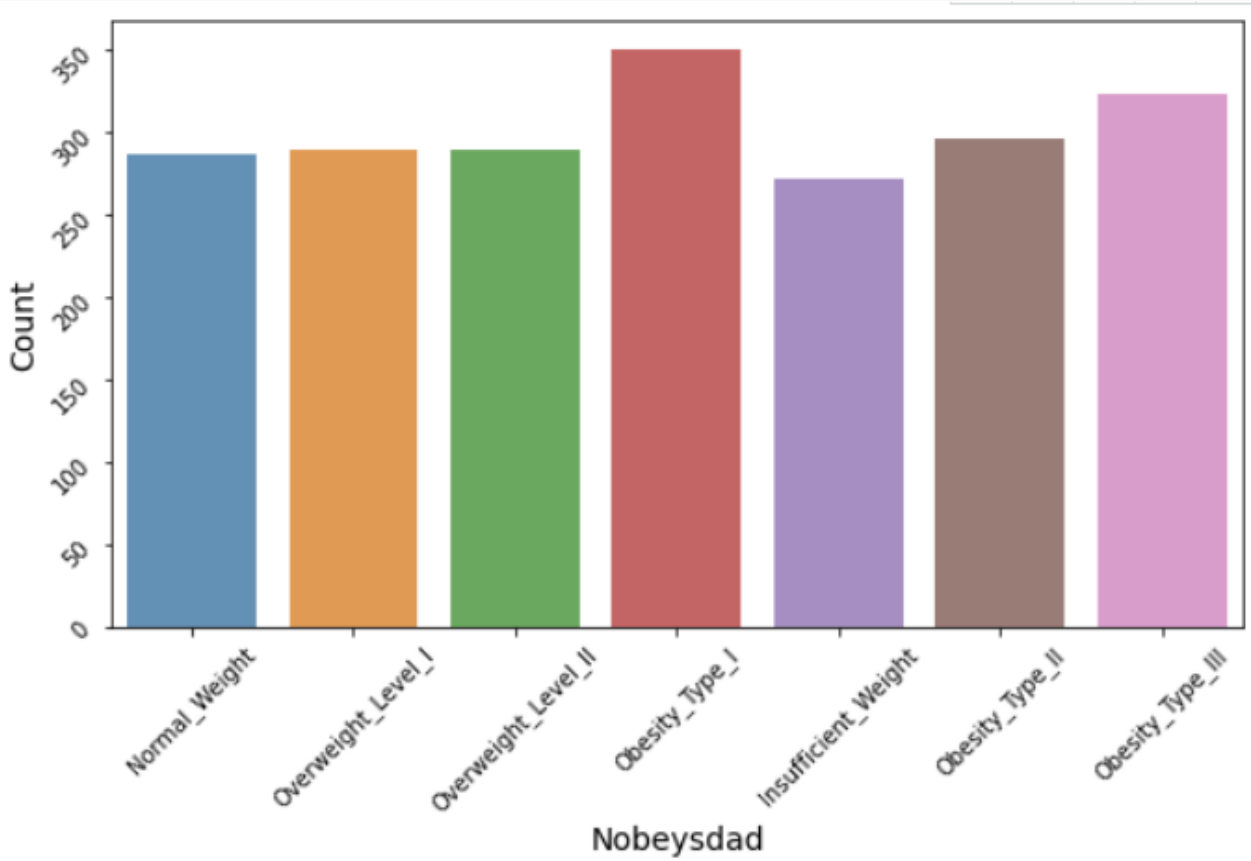


Correlation Matrix



1.From this matrix we can tell the connection between those different variantes.
2.The feature ‘height’ and ‘weight’ actually have a stronger connection to other variates compare to the rest variates

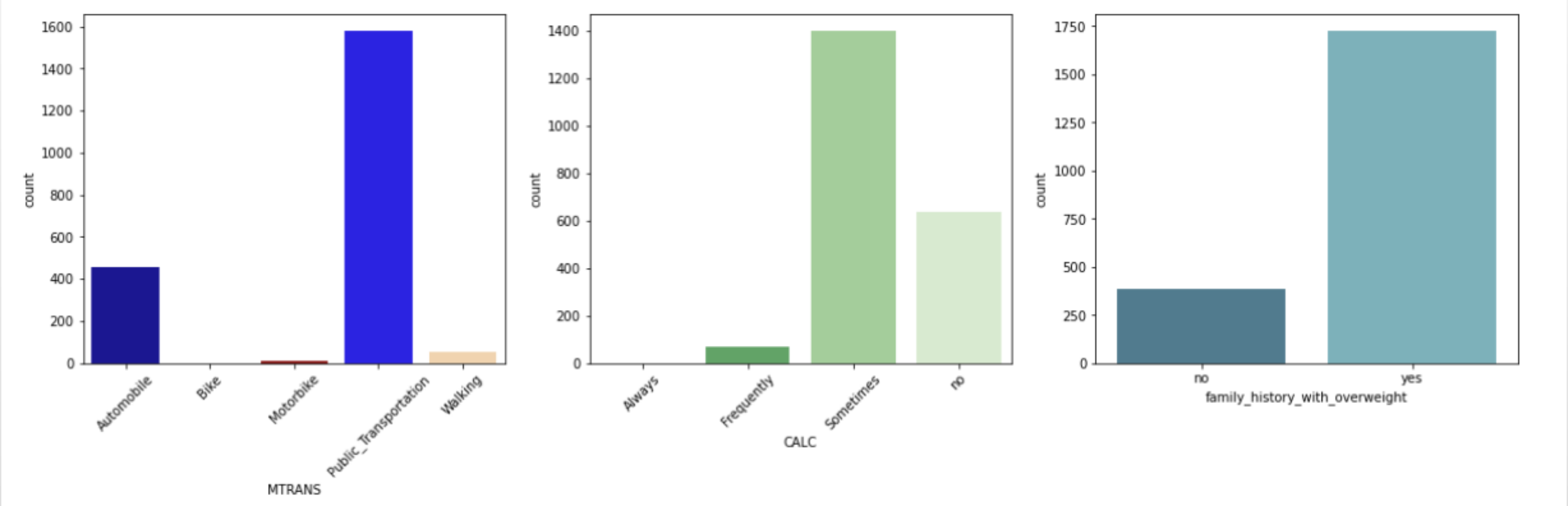
Univariate analysis



Its just simply a plot to count the number of each obesity level, we can find out each category have a very similar number
which indicate this dataset is a very good dataset in labeling.

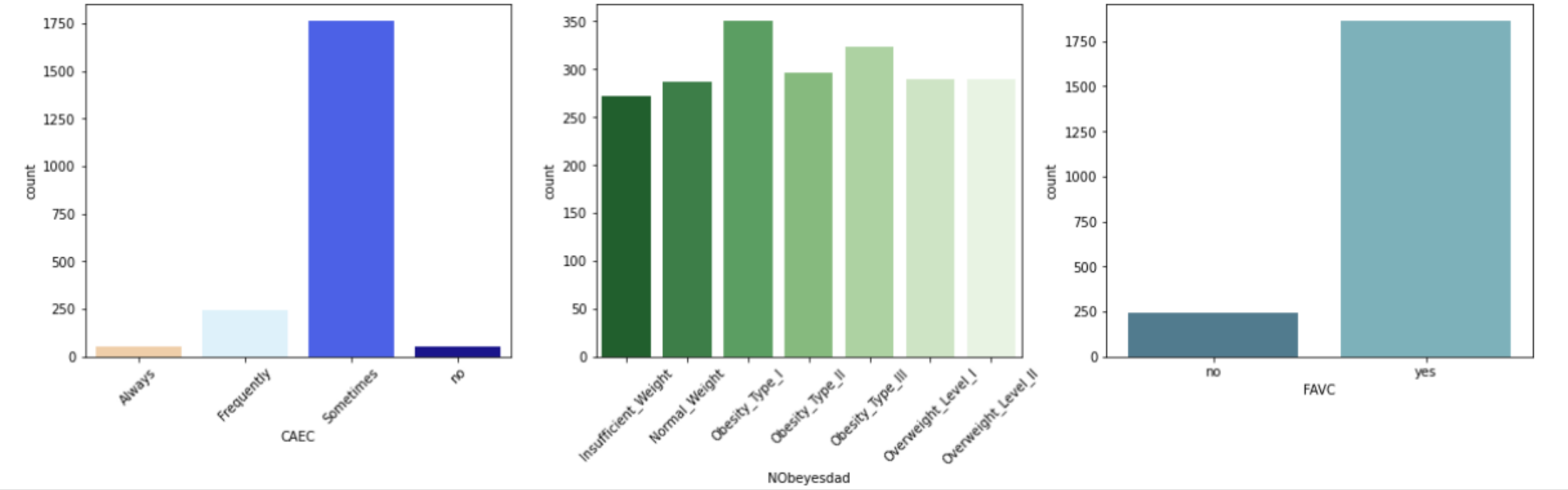
Univariate analysis

1.About 75% of people use Public transportation.
2.About 66% of people have a drink sometimes(a normal frequency).
3.About 82% of people have a family history with overweight.



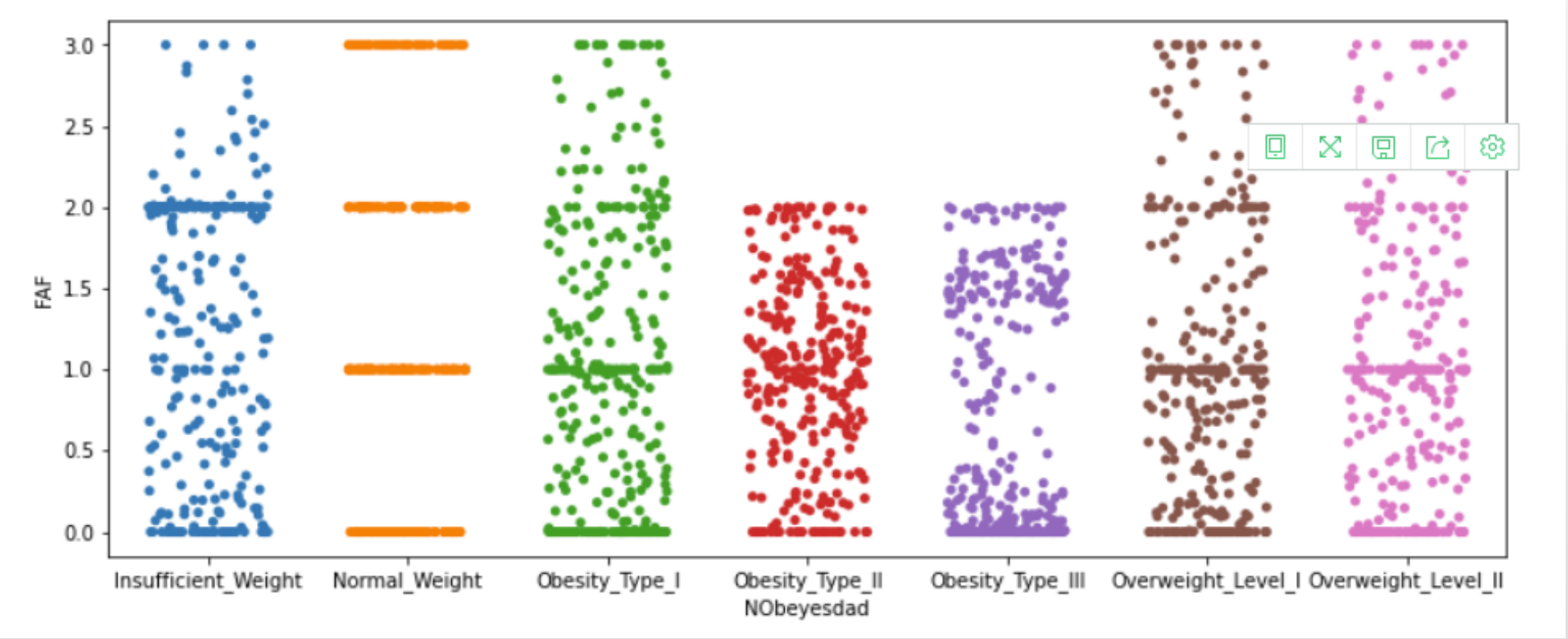
Univariate analysis

1.About 80% of people sometimes have consumption of food between meals.(CAEC)
2.About 75% of people eat high caloric food sometimes(a normal frequency)(FAVC).

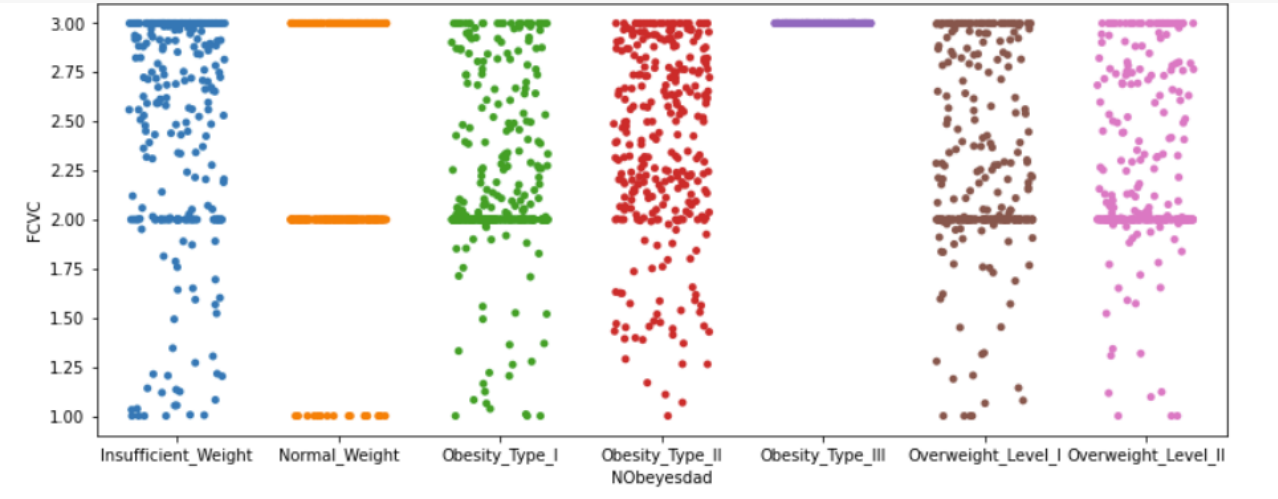


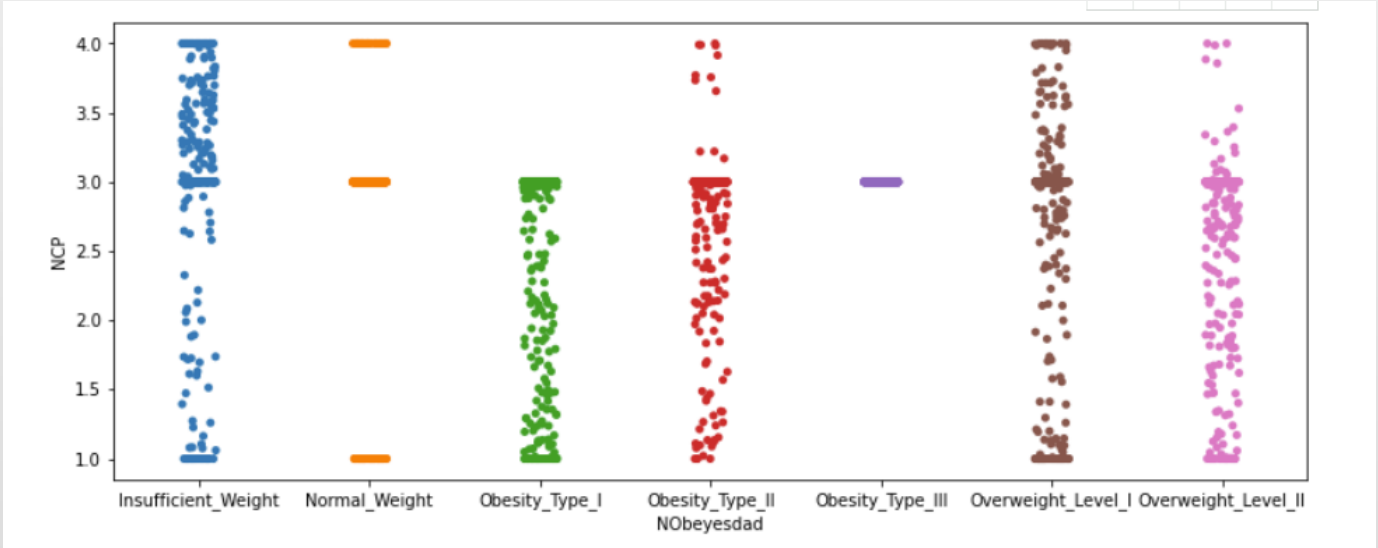
Single variante

- Here we use ‘striplot’ from ‘seaborn’ to visualize single variante, this is the plot of FAF(Physical activity frequency),we can tell that FAF have no special connection or at least haver no simple connection to obesity because for most of people no matter they workout a lot or absolutely not ,they can still be insufficient weight or overweight

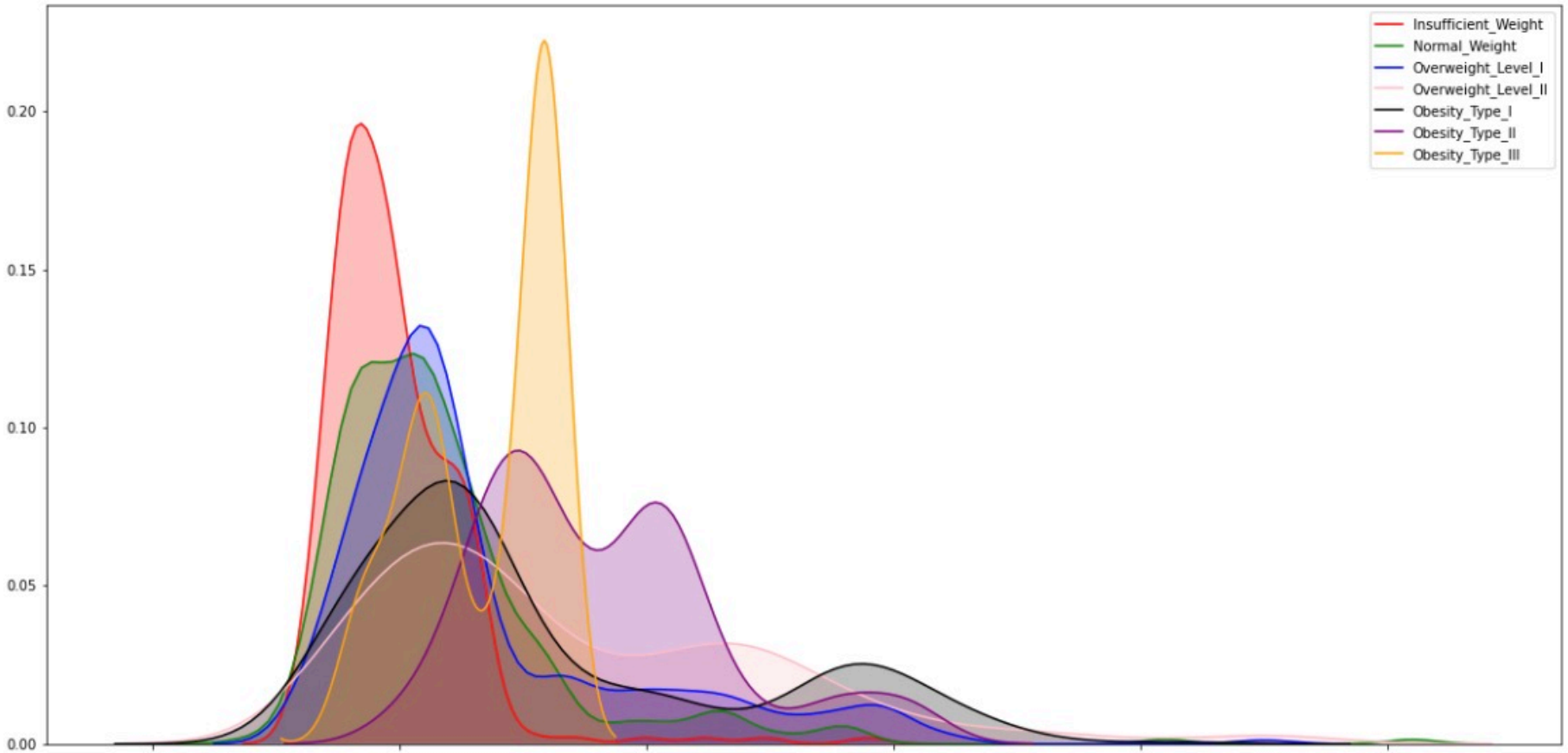


- so same resulat as the previous one ,we can also find that out come in coleration matrix



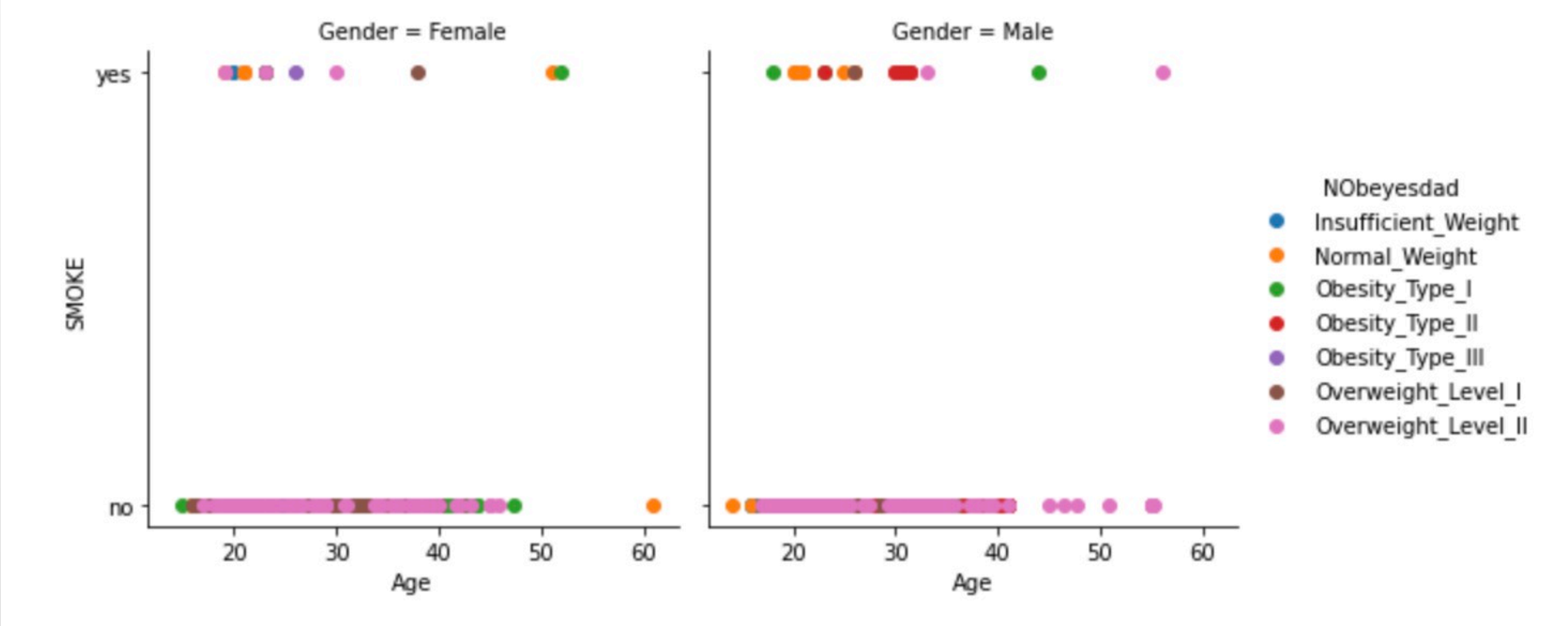


This is a density plot for age, we can tell at the age of 15-25, there is a high chance of insufficient weight probably because we are still growing and some might be growing a little slower than others, and at age 25-30, there is a high chance of super obesity.



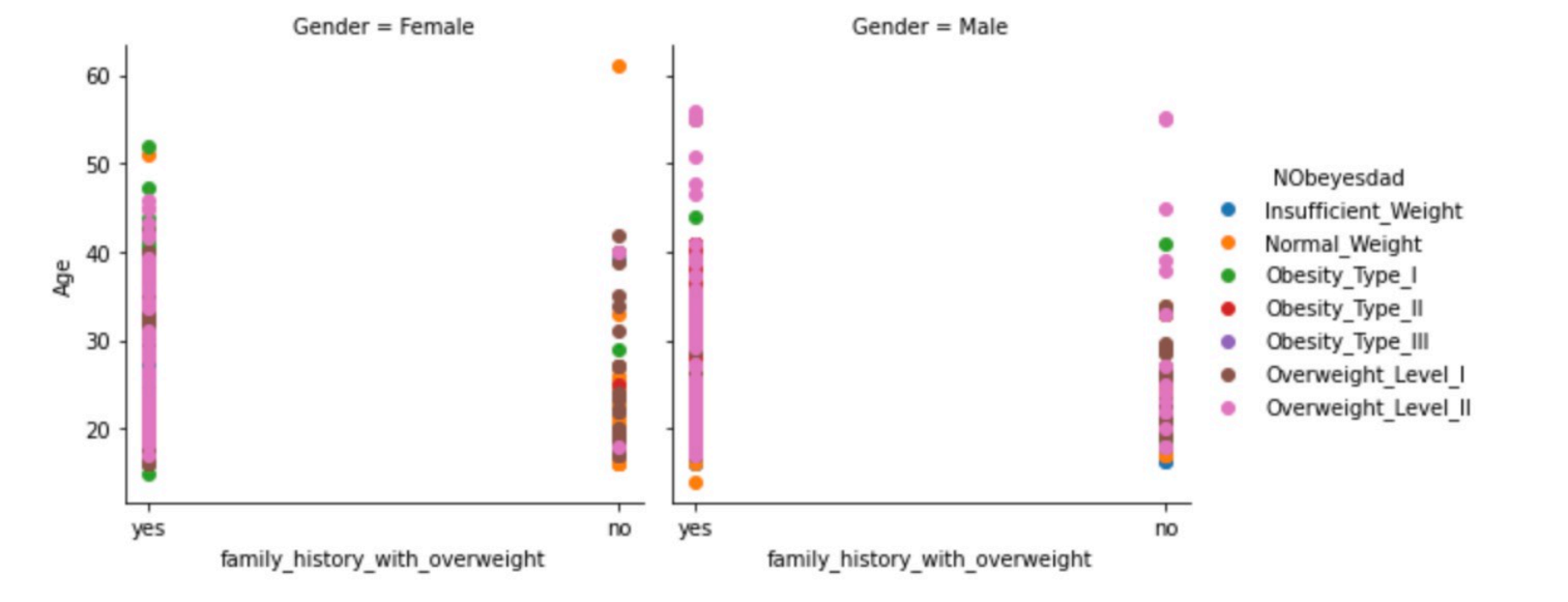
Multivariate analysis

Here we use ‘seaborn FaceGrid’ to analysis 3 different variates which are Smoke,Age and Gender and we find out most of the Overweight_Level_II dont smoke, at appears on all the the range of ages



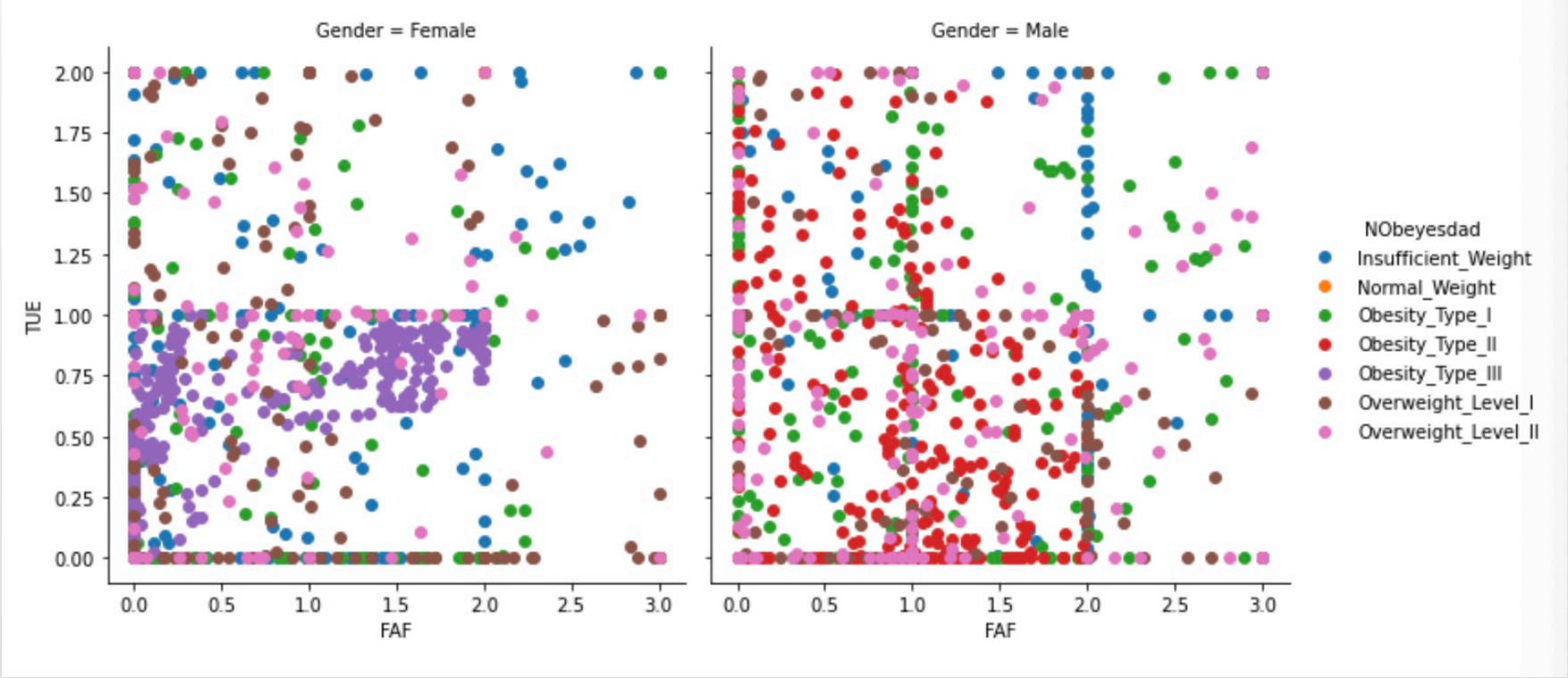
Multivariate analysis

Here we analysis 3 different variates which are family_history_with_overweight, Age and Gender and we find out most of the people who have family overweight history have a higher chance to be overweight



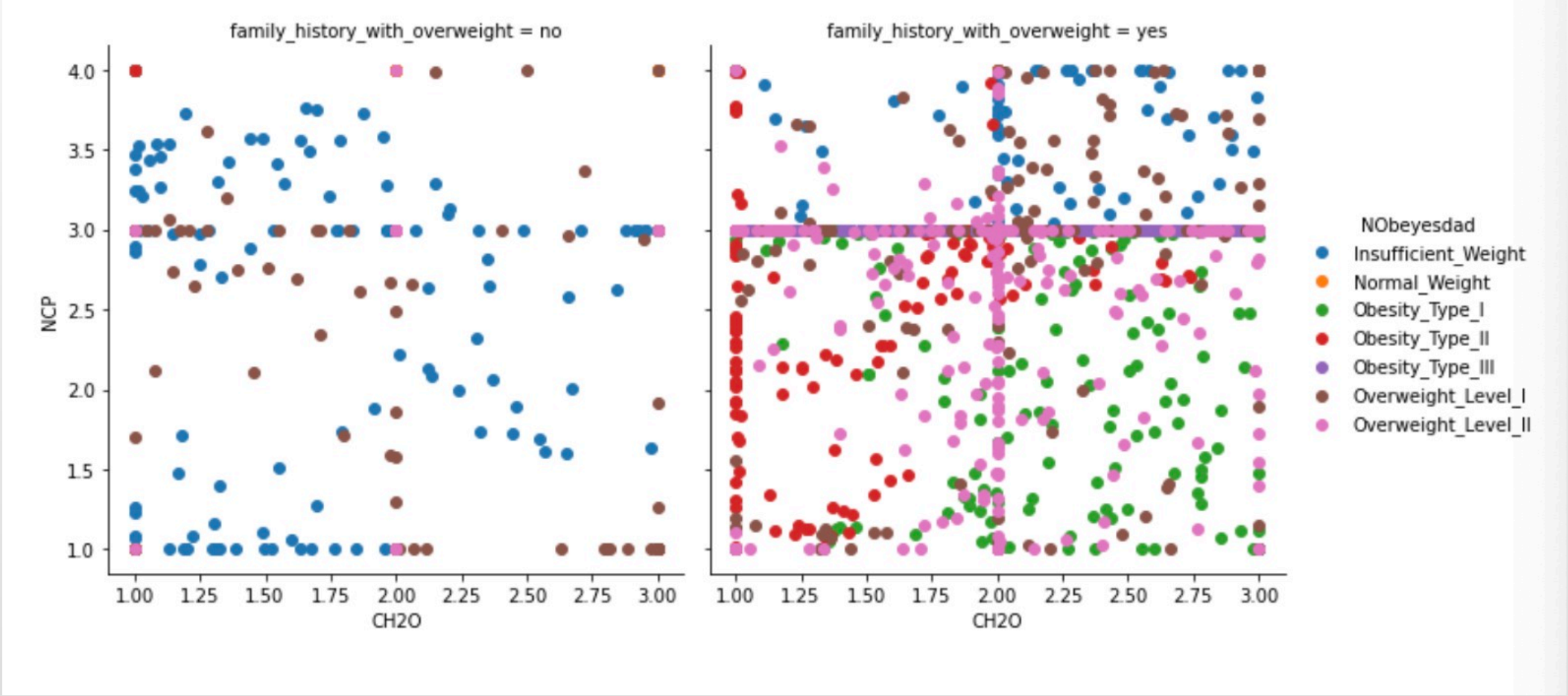
Multivariate analysis

Here we analysis 3 different variates which are FAF,Gender and TUE, so we can tell for either Female and Male, if TUE is lower than 1 and FAF lower than 2, people are more likely to have an overweight problem.



Multivariate analysis

Here we analysis 3 different variates which are NCP,CH2O and family history with overweight,so we can tell most people who dont have overweight history are not likely to have a overweight problem, and for people who drink 2 litre water per day and eat 3 meals per day are likely to have a overweight problem.



Data preprocessing

- Label processing(Here we transfer our category feature to numbers.)
- We consider the 0-6 which represent the obesity level.

```
df.NObeyesdad=df.NObeyesdad.map(dict(Insufficient_Weight=int(0),Normal_Weight=int(1),
                                     Overweight_Level_I=int(2),Overweight_Level_II=int(3),
                                     Obesity_Type_I=int(4),Obesity_Type_II=int(5),
                                     Obesity_Type_III=int(6)))
```

```
df.family_history_with_overweight=df.family_history_with_overweight.eq('yes').mul(1)
df.FAVC=df.FAVC.eq('yes').mul(1)
df.SCC=df.SCC.eq('yes').mul(1)
df.SMOKE=df.SMOKE.eq('yes').mul(1)
df.CAEC=df.CAEC.map(dict(no=0,Sometimes=1,Frequently=2,Always=3))
df.CALC=df.CALC.map(dict(no=0,Sometimes=1,Frequently=2,Always=3))
df.Gender=df.Gender.map(dict(Male=1,Female=0))
```

```
df.MTRANS=df.MTRANS.map(dict(Public_Transportation=int(0),
                             Automobile=int(1),Walking=int(2),
                             Motorbike=int(3),Bike=int(4)))
```

Choose the feature

We have 16 features, we want to choose most important features to train our model.We can find that the feature:SMOKE,SCC,FAVC,MTRANS have lower importance than others, so we drop it.

```
from sklearn import metrics
from sklearn.ensemble import ExtraTreesClassifier
model = ExtraTreesClassifier()
model.fit(X, y)
# display the relative importance of each attribute
print(model.feature_importances_)
```

```
[0.07962754 0.07912536 0.08151465 0.27046628 0.04731553 0.02581594
 0.07999671 0.05459426 0.03963989 0.00409607 0.04630274 0.01243931
 0.04219234 0.04561666 0.05481569 0.03644103]
```

```
print(X.columns)
```

```
Index(['Gender', 'Age', 'Height', 'Weight', 'family_history_with_overweight',
       'FCVC', 'NCP', 'CAEC', 'CH2O', 'FAF', 'TUE', 'CALC'],
      dtype='object')
```

```
# so we drop SMOKE SCC FAVC MTRANS
dfs=dfs.drop("SMOKE",axis=1)
dfs=dfs.drop("SCC",axis=1)
dfs=dfs.drop("FAVC",axis=1)
dfs=dfs.drop(['MTRANS'], axis=1)
```

Split the data

- We split the dataset into training data and test data by 0.3 which means 30% test data and 70% training data.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

Modeling

- **Logistic Regression**

- So we simply using logistic regression first, ‘sklearn’ provide very good logistic regression, we here set ‘max_iter’ to 1000 in ordre to have a higher accuracy

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=1000)
model = model.fit(X_train,y_train)
pred = model.predict(X_test)
pred
```

```
/Users/mariette/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_logistic.py
:762: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

- **Logistic Regression**

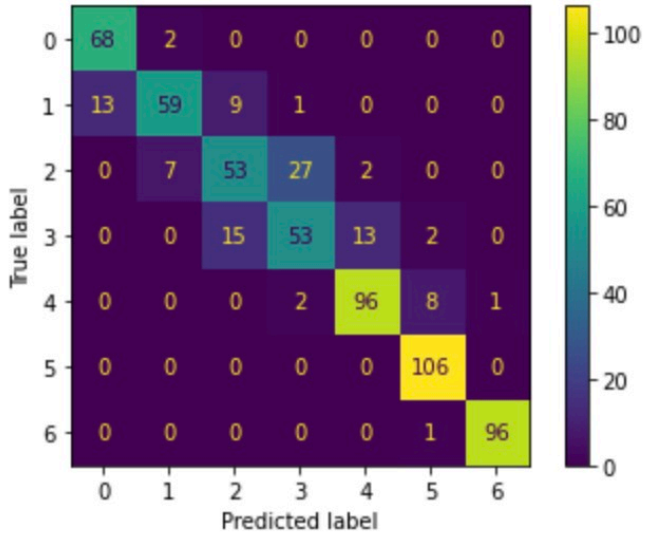
We can see the accuracy is about 80.4% and here is the confusion matrix, so we can see problems are most likely in level 0-4, actually logistic regression doesn't have a very good performance here

```
from sklearn.metrics import accuracy_score
accuracy_score(y_true=y_test, y_pred=model.predict(X_test))
```

```
0.8375394321766562
```

```
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(model, X=X_test, y_true=y_test)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fa3894972b0>
```



• Classification Decision Tree

```
# Classification and Regression Trees
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
# a tree with depth = 2
tree = DecisionTreeClassifier(random_state=3,criterion='entropy')
tree=tree.fit(X_train,y_train)
predicted = tree.predict(X_test)
```

```
def create_and_show_tree(data, y, estimator, ax=None):
    if not ax:
        fig, ax = plt.subplots(1, 1, figsize=(estimator.max_depth*5,5))
    estimator.fit(data, y)
    _ = plot_tree(estimator, ax=ax, fontsize=12)
    return estimator
```

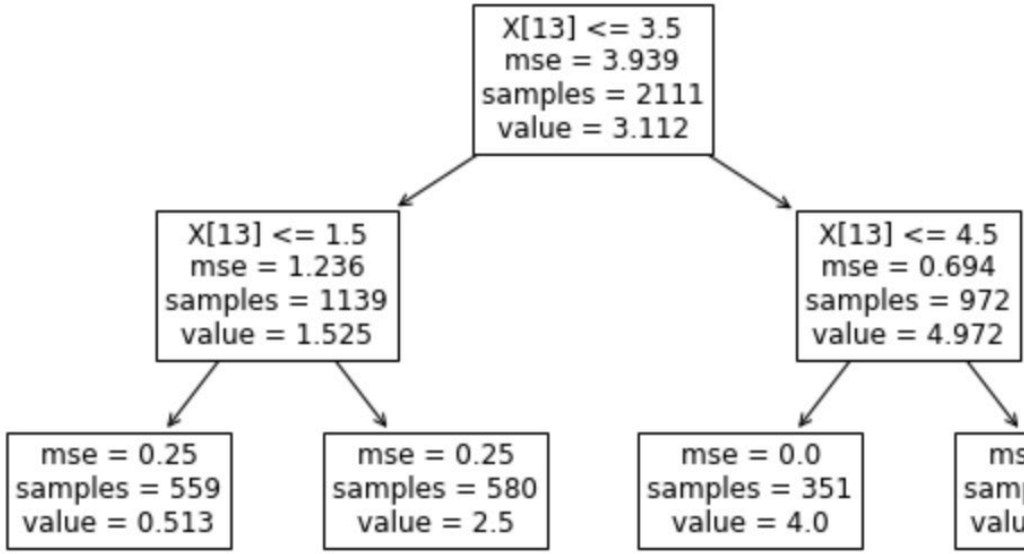
```
metrics.confusion_matrix(y_test, predicted)
```

```
array([[ 78,   0,   0,   0,   0,   0,   0],
       [  3,  90,   3,   1,   0,   0,   0],
       [  0,   4,  82,   2,   0,   0,   0],
       [  0,   0,   2,  81,   3,   0,   0],
       [  0,   0,   0,   2,  97,   0,   0],
       [  0,   0,   0,   0,   2,  81,   1],
       [  0,   0,   0,   0,   0,   0, 102]])
```

```
accuracy_score(y_true=y_test, y_pred=predicted)
```

```
0.9637223974763407
```

```
DecisionTreeRegressor(max_depth=2)
```



- # Randomforest

Using sklearn random forest consider our PC cpu, we set 10000 tree, and set random seed as 0, use all our cpu to calculate and predict the outcome
As our expectation ,random-forest give us the best outcome so far with a nearly 95.5% accuracy, we actually set different seed to calculate several times and the outcome is between 94%-96%.

RandomForest

```
from sklearn.ensemble import RandomForestClassifier
```

```
forest=RandomForestClassifier(n_estimators=10000,random_state=0,n_jobs=-1)
forest.fit(X_train,y_train)
pred_test_y=forest.predict(X_test)
```

```
accuracy_score(y_test,pred_test_y)
```

```
0.9463722397476341
```

```
metrics.confusion_matrix(y_test,pred_test_y)
```

```
array([[74,  4,  0,  0,  0,  0,  0],
       [ 1, 78,  2,  0,  0,  0,  0],
       [ 0, 11, 81,  3,  1,  0,  0],
       [ 0,  4,  4, 79,  1,  0,  0],
       [ 0,  0,  0,  2, 98,  0,  0],
       [ 0,  0,  0,  0,  0, 96,  0],
       [ 0,  0,  0,  0,  0,  1, 94]])
```

GridSearch

Use GridSearchCV to compare the effects of different models and different parameters on the experimental results.

```
model_params = {
    'svm':{
        'model':svm.SVC(gamma='auto'),
        'params':{
            'C':[1,10,20],
            'kernel':['rbf','linear']
        }
    },
    'random_forest':{
        'model':RandomForestClassifier(),
        'params':{
            'n_estimators':[1,5,10]
        }
    },
    'logistic_regression':{
        'model':LogisticRegression(),
        'params':{
            'C':[1,5,10]
        }
    },
    'naive_bayes_gaussian':{
        'model':GaussianNB(),
        'params':{}
    },
    'naive_bayes_multinomial':{
        'model':MultinomialNB(),
        'params':{}
    },
    'decision_tree':{
        'model':DecisionTreeClassifier(),
        'params':{
            'criterion':['gini','entropy']
        }
    }
}
```

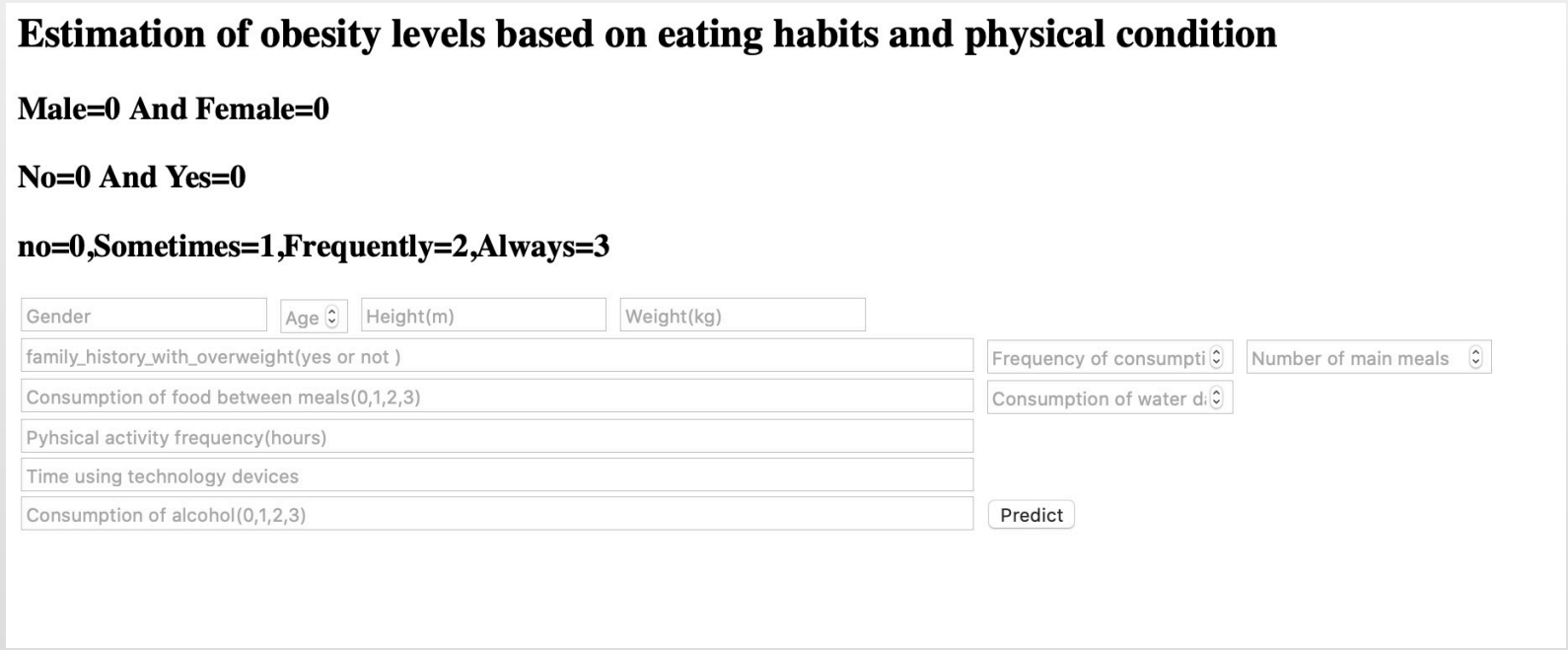
	model	best_score	best_params
0	svm	0.945843	{'C': 20, 'kernel': 'linear'}
1	random_forest	0.916727	{'n_estimators': 10}
2	logistic_regression	0.657432	{'C': 5}
3	naive_bayes_gaussian	0.658727	{}
4	naive_bayes_multinomial	0.565337	{}
5	decision_tree	0.924847	{'criterion': 'gini'}

Transfer our dataset to Flask

- We use pickle to save our model in order to use in the Flask

```
loaded_model = pickle.load(open(filename, 'rb'))
result = loaded_model(X_test, y_test)
print(result)
```

- And we create a web where you can test your obesity level by enter your data(your habit). And you can test many times



Conclusion

- By doing this analysis, we can know that our eating habits and physical condition is very important to our health, it can influence our obesity level. People who have lower calorie food, always do physical exercises, drink more water can have a healthier obesity level.
- Thanks you for your attention.