In assignment 3, we are asked to preprocess the data, and fit the data into a Naïve Bayes classifier to get higher accuracy. My preprocessing is based on Pipeline and Grid Search method.*

1. Preprocess the data

Remove the rows, I tried to remove the row with at least one NA or outlier in it. However, each row has at least one missing value or outlier and we will end up with an empty dataset, so it's useless.

Impute the NA values with global value. I filled the NA value with -10000. However, it doesn't work because the Naïve Bayes classifier can't take the infinite values of the attributes.

Impute the NA values with the mean/median value. Here I imputed the NA with mean of each attribute. The average accuracy of mean imputation is 0.007 higher than that of median imputation.

Smooth the data by binning into intervals. Next, I tended to smooth the noise in attributes by binning into 5 groups with identical width. However, GaussianNB classifier is used to handle continuous data, and binning makes data more categorical. Plus, some attributes' values are too close to each other, so it threw the error that "the bins should be monotonously increasing or decreasing", when using 'quantile' and 'k-means' to define the width of intervals, which means bins can't be split properly with tiny intervals, which might overlap together. Anyway, the accuracy decreased slightly so I gave it up.

Normalization. I tried to normalize samples individually to unit norm by Normalizer transformer. I don't know the specific reason why it works in Naïve Bayesian. But I know it makes the training process better for most algorithms, because it can avoid the problem that great difference in the scale of the numbers when attempting to combine the values as features. Accuracy increased by 0.0027.

Feature selection. I have done feature selection two times in data preprocessing. One is manual selection before data imputation. I looked at the data and noticed a regular diagonal pattern that attributes' values on the diagonal line from left to right have the same value every ten adjacent attributes. I think those attributes might be repeated recordings or time series recordings, which they are highly correlated. So, I tried to manually choose the first attribute from every ten attributes and keep those 310 attributes in dataset. The accuracy decreased by 0.0084 from 66.96% to 66.12% on average, with the variance decreased by 39%, which is under my expectation. (I also tried to use ExtraTreesClassifier to choose attributes in this step, but I don't know why the accuracy is even lower than manual selection.) The second time is when I used SelectKBest method to select 50 best attributes from 310 attributes after data cleaning and normalization, which decreases the accuracy to 64.81%. I think it is because the model used to be overfitting with 3120 attributes. When attributes were reduced too much, the accuracy was reduced to the underfitting area. So, I think there might be some other methods to find the best number of attributes to get rid of underfitting and overfitting.

Grid Search. I set different values for parameters to let Python search through. It found out that using mean to impute the missing value and choose 150 attributes in the second times will have the highest accuracy. Although if I set random_state = 0, it always has higher accuracy than other random_seed which can reach to 71.79%, it's 67.84% on average of 100 different random seed.

2. Five most important features

five best attributes	Attributes	Scores
No.1	CO2 0	0.1732
No.2	m59.0491 0	0.1542
No.3	m18.0338 0	0.1341
No.4	m29.9974 0	0.1141
No.5	m41.0386 0	0.108

Important features should be less correlated to each other and have highest scores. I chose uncorrelated attributes manually based on my understanding of observation data and used SelectKBest method to choose the highest scores. The result is listed on the left table.

*Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011