Improvement-to-Naive-Bayes-Classifier

In this part, I propose a new idea consisting of two affiliated approaches to improve the Naive Bayes classifier, which is a prevalent probabilistic classifier. The method is also based on the Bayes' theorem. I will review the general idea of Bayesian classifier first, and then discuss about the methodologies for each method. Later on I will analyze the testing results of these methods. Last but not least, I will summarize the pros and cons for each of them.

Idea of Bayesian Classifier

Before entering into the introduction of each method, let's recall the Bayes theorem first. In Bayes theorem, we wish to compute the conditional probability of an event A given that another event B has happened. The formula $Pr(A|B) = \frac{Pr(A,B)}{Pr(B)} = \frac{Pr(B|A)*Pr(A)}{Pr(B)}$ can transfer the calculation of the joint probability of events A and B to the posterior probability of event B given A and the prior probability of A.

When it comes to the classification tasks in machine learning, suppose the input X is n-dimensional. (i.e. Feature are X_0 ~ X_{n-1}). Given the input data $(x_0, x_1, \ldots, x_{n-1})$, we would like to compute the posterior probability of each possible class Y_i and select the one with the largest posterior probability. Thus, according to the Bayes theorem, we have the following formula:

$$Pr(Y_i|X_0 = x_0, X_1 = x_1, \ldots, X_{n-1} = x_{n-1}) = \frac{Pr(X_0=x_0,X_1=x_1,\ldots,X_{n-1}=x_{n-1}|Y_i)*Pr(Y_i)}{Pr(X_0=x_0,X_1=x_1,\ldots,X_{n-1}=x_{n-1})}$$

Since for each class, the denominator remains the same, only $Pr(X_0 = x_0, X_1 = x_1, \ldots, X_{n-1} = x_{n-1}|Y_i)$ and $Pr(Y_i)$ affect the result. Indeed, the most crucial part is to compute the joint conditional probability. All of the methods in the following section differs only in the way of computing $Pr(X_0 = x_0, X_1 = x_1, \ldots, X_{n-1} = x_{n-1}|Y_i)$.

Methodologies

In general, the features (or say, the variables) can be roughly divided into 3 types: *continuously distributed*, *discretely distributed but rankable*, *discretely distributed but unrankable*. For the discrete but unrankable features, we just treat them as mutually independent with the other features. But for continuous features and discrete but rankable features, in this research, I propose two methods which are **Gaussian Copula** and **Comonotonicity** to compute the conditional joint densities. Within each of them, there are two branches which are **aggregation** and **clustering**. For aggregation, we just compute the conditional joint densities of all continuous features and discrete but rankable features. For clustering, we first compute the correlation coefficient matrix and take the absolute value elementwisely. Next, we use 1 minus the absolute value of correlation coefficient matrix. In this way, we get a distance matrix D in which the higher correlation between two features X_i and X_j (no matter whether positive correlation or negative

correlation), the smaller D_{ij} would be. Then we use agglomerative nested clustering (AGNES) method to do hierarchical clustering to these features. For any two clusters, the distance between them equals to the shortest distance between the features in these two clusters. Note that we should set a minimum correlation min_corr so that the clustering algorithm will stop if the distance between any two clusters is larger than $1 - min\_corr$.

Naive Bayes Classifier

In Naive Bayes, the "naive" assumption is the mutual independence of all the features. In this way, the joint conditional probability equals to the product of conditional probabilities of each X_i. Thus, regardless of the dependence structure of input features, we use $\prod_{i=0}^{n-1} Pr(X_i = x_i|Y_i)$ to replace $Pr(X_0 = x_0, X_1 = x_1, \ldots, X_{n-1} = x_{n-1}|Y_i)$. If a feature X_i is continuous-valued, $Pr(X_i = x_i|Y_i)$ is usually computed based on Gaussian distribution with a mean μ and variance σ : $Pr(X_i = x_i|Y_i) = N(X_i = x_i|\mu_{Y_i}, \sigma_{Y_i}) = \frac{1}{\sqrt{2\pi}\sigma_{Y_i}} e^{\frac{(x_i - \mu_{Y_i})^2}{2\sigma^2}}$.

Since Naive Bayesian classification required each conditional probability to be non-zero, otherwise, the product of posterior probabilities may become zero, we should also employ Laplacian correction to avoid the zero probability problem. It can be achieved by adding 1 to the number of occurences to each case.

In general, although the Naive Bayes classifier ignores the dependence structure of input features, it indeed performs well on many datasets. Intuitively speaking, this might be due to the nature of classification tasks. We just need to get accurate ranking of posterior probabilities on each class rather than computing the exact posterior probability. Thus, if the features share relatively weak dependencies and the number of features is small, Naive Bayes can be robust. However, things becomes different when it comes to strong dependencies and high dimensional data.

Comonotonic Classifier

I propose an approach to estimate the conditional joint density based on comonotonicity in Actuarial Science. Just like what we did in the previous section, there are two sub-methods. The first one is to treat all continuous or discrete but rankable features as comonotonic. The other one is to conduct the method only within clusters.

In probability theory, comonotonicity refers to perfect positive dependence among the components of a random vector. Mathematically, a set $S \subset \mathbf{R^n}$ is comonotonic if $\forall(x_1, \ldots, x_n)$ and $\forall(y_1, \ldots, y_n) \in S$, if $\exists i \in \{1, \ldots, n\}$ such that $x_i < y_i$, then $x_j < y_j \quad \forall j \in \{1, \ldots, n\}$. **[1]** In other words, a comonotonic set S is a totally ordered set.

Comonotonicity is widely used in risk measures. Before delving into the probability measure, we recall the method to sample from a certain probability distribution. Suppose a random variable X has CDF F_X, since F_X ~ $U(0, 1)$, the way to sample from X's distribution is to uniformly generate a random number u from 0 to 1 and the sample value $x = F_X^{-1}(u)$. Then in the case of comonotonicity, suppose $X_1, \ldots, X_n$ are comonotonic, we sample by uniformly generating a random number u from 0 to 1 so that $x_i = F_{X_i}^{-1}(u) \quad \forall i \in \{1, \ldots, n\}$. In this sense, not all combinations of values for $X_1, \ldots, X_n$ are feasible since we generate all of them using the same u.

Theoretically, for datasets with a large feature space n, assume for each feature there are m possible values on average. Then, a fully general machine learning algorithm which attempts to learn all possible dependencies among the features has to deal with m^n possible combinations of all features, and thus needs $O(m^n)$ many data. But for Naive Bayes, due to the "naive" assumption on mutual independence, it only needs $O(mn)$ many data. Meanwhile, for comonotonicity, within a group of k comonotonic features, the feature value combination is $O(km)$ and for all features the feature value combination is $O((km)^{\frac{n}{k}})$. For aggregated comonotonic classifier, $k = 1$. Therefore, in terms of the size of datasets, Naive Bayes < Comonotonic classifier < general machine learning algorithms.

Probability Measure

As for the probability measure, according to both Hoeffding (1940) and Frechét (1951) **[2] [3]**, for any n-dimensional random vector $\mathbf{X} = (X_1, \ldots, X_n)$ with multivariate CDF $F_{\mathbf{X}}$ and marginal univariate CDF's $F_{X_1}, \ldots, F_{X_n}$ and for any $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbf{R}^n$, we have the following inequality: $F_{\mathbf{X}}(\mathbf{x}) \leq min(F_{X_1}(x_1), \ldots, F_{X_n}(x_n))$. According to this inequality, we derive that $Pr(\mathbf{X} = \mathbf{x}) = min(F_{X_1}(x_1), \ldots, F_{X_n}(x_n)) - max(F_{X_1}(x_1 - t_1), \ldots, F_{X_n}(x_n - t_n))$ where $t_i = 1$ if X_i is discrete but rankable and $t_i \to 0^+$ if X_i is continuous. For discrete but rankable variables, $Pr(X_i = x_i)$ can be represented by an interval $(max\{0, \sum_{k=0}^{k=x_i-1} Pr(X_i = k)\}, \sum_{k=0}^{k=x_i} Pr(X_i = k)]$. However, there is no probability mass for continuously distributed variables. In order to get an interval for $X_i = x_i$, we must consider an error term t_i. However, the problem is that the selection of t_i will affect the interval significantly. Thus, we need to discretize the variables before applying comonotonic method. The method for discretization is of great importance for the performance of the model, I will provide more details in the experiment part of this report.

In terms of marginal distribution, when we draw a sample from the distribution of X_i, we just need to uniformly generate a random number u between 0 and 1 and get $X_i = x_i$ if and only if u is in this interval. In this way, we can transform the calculation of probability mass into finding the intersection of n intervals. Intuitively, when evaluating the joint probability of dependent variables, multiplication of marginal probabilities will make the joint probability smaller. However, when it comes to the intersection of intervals, as long as the comonotonicity condition holds, such error can be reduced.

Aggregated Comonotonic Classifier

Without the loss of generality, we assume $X_0, \ldots, X_{n-1}$ are continuous or discrete but rankable features. In this approach, we treat all of them as comonotonic features. This is actually a very naive assumption and it is very likely that the intersection of n intervals is empty, which triggers the Laplacian correction and exacerbates the performance of the model.

Clustered Comonotonic Classifier

In this approach, we use the same clustering algorithm as stated in the previous section for Gaussian Copula. Now that the variables are all discrete, there are two types of variables: rankable and unrankable. For the rankable variables, we compute the probability mass of each cluster. Within the cluster, say, we have variables $X_0, \ldots, X_5$, we assume that they are comonotonic. We treat X_0 as the basic variable and for all other variables X_i's, if $corr(X_0, X_i) < 0$, we take the negative of X_i in order to keep a positive correlation with the

basic variable. Otherwise, keep X_i unchanged. Suppose in an observation in the test set, $X_i = x_i$ for i between 0 and 5, we compute 6 intervals by the formula:
$(max\{0, \sum_{k=0}^{k=x_i-1} Pr(X_i = k)\}, \sum_{k=0}^{k=x_i} Pr(X_i = k)]$, and get the intersection. The length of intersection is the probability mass of this cluster conditional to a certain class c.

Applying the method above to every observation in the test set, we can get the probability distribution of each test data over all possible classes. In this case, we choose the largest one to be the predicted class.

Experiments

The selection of dataset is nontrivial for the verification of a new algorithm. Since the objective of this research is to improve Naive Bayes classifier, at least on some specific datasets, the performance of Naive Bayes should not be good on these datasets. Otherwise, no significant improvement can be made anymore. Referring to the methodology of Naive Bayes, it ignores the dependence structure of features. Thus, based on intuition, we may consider datasets with strong dependency among features. For each dataset in this section, I apply Naive Bayes Classifier, Aggregated Comonotonic Classifier, Clustered Comonotonic Classifier separately. For the binary classification task, I use ROC Curve and AUC value to evaluate the performance.

Before describing the experiments, I provide the details for discretisation first.

Discretisation

Transforming a continuous feature into a discrete feature is of vital importance for comonotonic classifier. Traditionally, we can discretise the value into equal-width bins based on the range or discretise into equal-sized buckets based on rank or sample quantiles. Both of them requires the specification of number of categories, which is nontrivial as well. I have tried both of these discretization methods and also tried to optimize the combination of number of classes for all continuous features. Unfortunately, these trials are unsuccessful and I will describe them in the section of Discussion.

In practice, I discretize the continuous features by mean and standard deviation. For each continuous feature and each class in the training set, the bins should be stored for the sake of data discretization for continuous features in the test set. Here I discretize each continuous variable given the class number into 8 categories. The bins are: $(-\infty, mean-3*std]$, $(mean-3*std, mean-2std]$, $(mean-2*std, mean-std]$, $(mean-std, mean]$, $(mean, mean+std]$, $(mean+std, mean+2*std]$, $(mean+2*std\ mean+3*std]$, $(mean+3*std, \infty)$. Then for each rankable variable X_i, we store the value of $Pr(X_j = k|Y = Y_j)$ for any possible value k. Note that we should use Laplacian correction to avoid zero probability.

Overall Performance on MNIST Dataset

Inspired by the application of logistic regression on MNIST dataset, I discovered the potential of comonotonic classifier on image processing. The MNIST dataset consists of handwritten digits from 0 to 9 and thus, there are 10 classes. When we train a logistic regression model on MNIST dataset, basically, we should reshape each image of size $28 * 28$ in which each entry is real-valued to a vector of size $1 * 784$. Then we randomly initialize a matrix of size $784 * 10$ and use stochastic gradient descent to optimize the cross entropy error function. Using the idea from this, I also reshape each image in MNIST dataset and conduct discretization to each of the 784 features. The results are shown below.

Naive Bayes

	precision	recall	f1-score	support
0	0.79	0.89	0.84	980
1	0.85	0.95	0.90	1135
2	0.90	0.26	0.40	1032
3	0.71	0.35	0.47	1010
4	0.88	0.17	0.29	982
5	0.55	0.05	0.09	892
6	0.65	0.93	0.77	958
7	0.88	0.27	0.42	1028
8	0.28	0.67	0.40	974
9	0.37	0.95	0.53	1009
accuracy			0.56	10000
macro avg	0.69	0.55	0.51	10000
weighted avg	0.69	0.56	0.52	10000

Aggregated Comonotonic

	precision	recall	f1-score	support
0	0.00	0.00	0.00	980
1	0.03	0.23	0.05	1135
2	0.00	0.00	0.00	1032
3	0.00	0.00	0.00	1010
4	0.00	0.00	0.00	982
5	0.00	0.00	0.00	892
6	0.00	0.00	0.00	958
7	0.01	0.00	0.01	1028
8	0.00	0.00	0.00	974
9	0.00	0.00	0.00	1009
accuracy			0.03	10000
macro avg	0.00	0.02	0.01	10000
weighted avg	0.00	0.03	0.01	10000

Clustered Comonotonic

	precision	recall	f1-score	support
0	0.91	0.90	0.90	980
1	0.90	0.96	0.93	1135
2	0.89	0.82	0.85	1032
3	0.79	0.84	0.81	1010
4	0.83	0.81	0.82	982
5	0.82	0.73	0.77	892
6	0.89	0.89	0.89	958
7	0.93	0.85	0.89	1028
8	0.76	0.79	0.77	974
9	0.75	0.85	0.80	1009
accuracy			0.85	10000
macro avg	0.85	0.84	0.84	10000
weighted avg	0.85	0.85	0.85	10000

Analysis of results

From the results, we can see that the clustered comonotonic classifier demonstrates significant improvement in precision, recall, F1-score and accuracy rate compared with the Naive Bayes classifier. However, The aggregated comonotonic classifier performed extremely poor when dealing with high dimensional dataset. Literally, we can attribute the poor performance of aggregated comonotonic classifier to the naive assumption of comonotonicity among all rankable features, which leads to empty intersection of intervals and triggered the Laplacian correction and exacerbated the result.

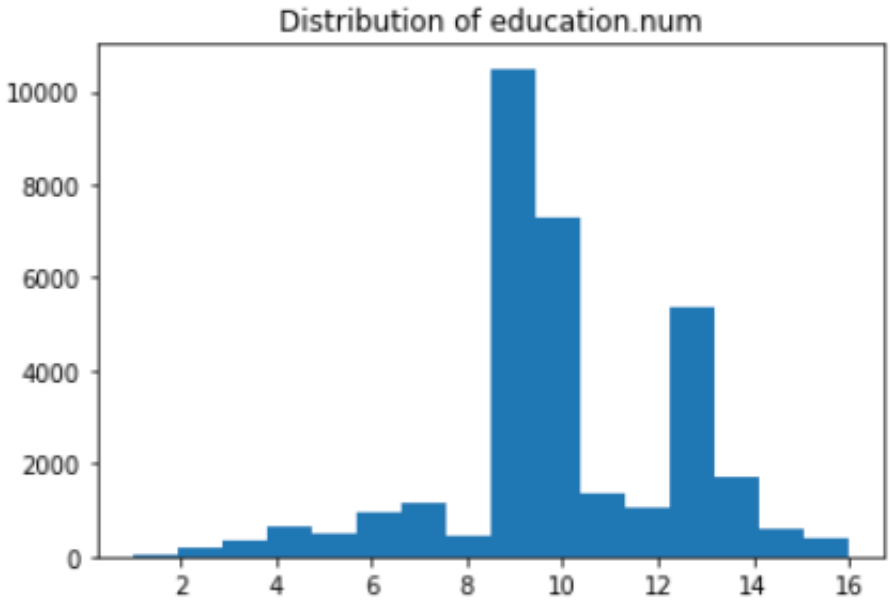
As for Naive Bayes, we can see that the precision and recall is not stable on some classes. For example, the precision on classes 5, 8, 9 and the recall on classes 2, 3, 4, 5, 7 are even lower than 0.6. But for clustered comonotonic classifier, the precision and recall on each class is at least 0.7 and the testing accuracy significantly surpasses that of Naive Bayes.

Our data processing method for the MNIST dataset is to reshape the images, or in other words, $2d$ arrays to $1d$ arrays. Although such reshaping operation commits spacial information loss from the original image, the accuracy is over 80%. In the next section, I will apply the three algorithms on a low dimensional dataset and compare the performances again.

Overall Performance on Adult Census Income dataset

Here we apply the three algorithms on the Adult Census Income dataset. This dataset was extracted from the 1994 Census bureau database by Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics) **[4]**. It includes numerical and categorical features. They are **age**, **workclass**, **fnlwgt**, **education**, **education. num**, **marital. status**, **occupation**, **relationship**, **race**, **sex**, **capital. gain**, **capital. loss**, **hours. per. week**, **native. country**. The objective is to predict whether an adult has income over $\$50k$ or not based on the 14 features. Intuitively, we may find that there would be strong correlation between **education** and **education. num**. Indeed, in the original dataset, **education** is a feature of text value specifying the actual highest level of institution that this person has ever taken while **education. num** is a natural number ranking the level of institution in the **education** column.

In order to increase the level of dependence among the features while not just simply replacing one feature with another, I did some data transformation. Referring to the distribution of **education.num** (as shown in the plot below), most of the people are between level 9 and level 14. Thus, I encode the feature **education** according to the value in **education.num**. The correspondence is: $\leq 9 \rightarrow 0$, $10 \rightarrow 1$, $11\&12 \rightarrow 2$, $\geq 13 \rightarrow 3$. In this way, the two columns are highly correlated but they are not strictly equal.



For the description of the feature **fnlwgt**, as stated on Kaggle:

"The weights on the Current Population Survey (CPS) files are controlled to independent estimates of the civilian noninstitutional population of the US. These are prepared monthly for us by Population Division here at the Census Bureau. We use 3 sets of controls. These are:

1. A single cell estimate of the population 16+ for each state.
2. Controls for Hispanic Origin by age and sex.
3. Controls by Race, age and sex.

We use all three sets of controls in our weighting program and "rake" through them 6 times so that by the end we come back to all the controls we used. The term estimate refers to population totals derived from CPS by creating "weighted tallies" of any specified socio-economic characteristics of the population. People with similar demographic characteristics should have similar weights. There is one important caveat to remember about this statement. That is that since the CPS sample is actually a collection of 51 state samples, each with its own probability of selection, the statement only applies within state."

Now after some preprocessing, the continuous features are **age**, **fnlwgt**, **education.num**, **capital.gain**, **capital.loss**, **hours.per.week**. The unrankable features are **workclass**, **marital.status**, **occupation**, **relationship**, **race**, **sex**, **native.country**. The discrete but rankable feature is **education**. I apply the three models to this preprocessed dataset.

The results are as follows:

Naive Bayes

	precision	recall	f1-score	support
0	0.80	0.95	0.87	4623
1	0.66	0.29	0.41	1521
accuracy			0.79	6144
macro avg	0.73	0.62	0.64	6144
weighted avg	0.77	0.79	0.76	6144

Aggregated Comonotonic

	precision	recall	f1-score	support
0	0.89	0.77	0.83	4623
1	0.51	0.72	0.60	1521
accuracy			0.76	6144
macro avg	0.70	0.75	0.71	6144
weighted avg	0.80	0.76	0.77	6144

Clustered Comonotonic

	precision	recall	f1-score	support
0	0.91	0.85	0.88	4623
1	0.63	0.74	0.68	1521
accuracy			0.83	6144
macro avg	0.77	0.80	0.78	6144
weighted avg	0.84	0.83	0.83	6144

AUC-ROC Curve

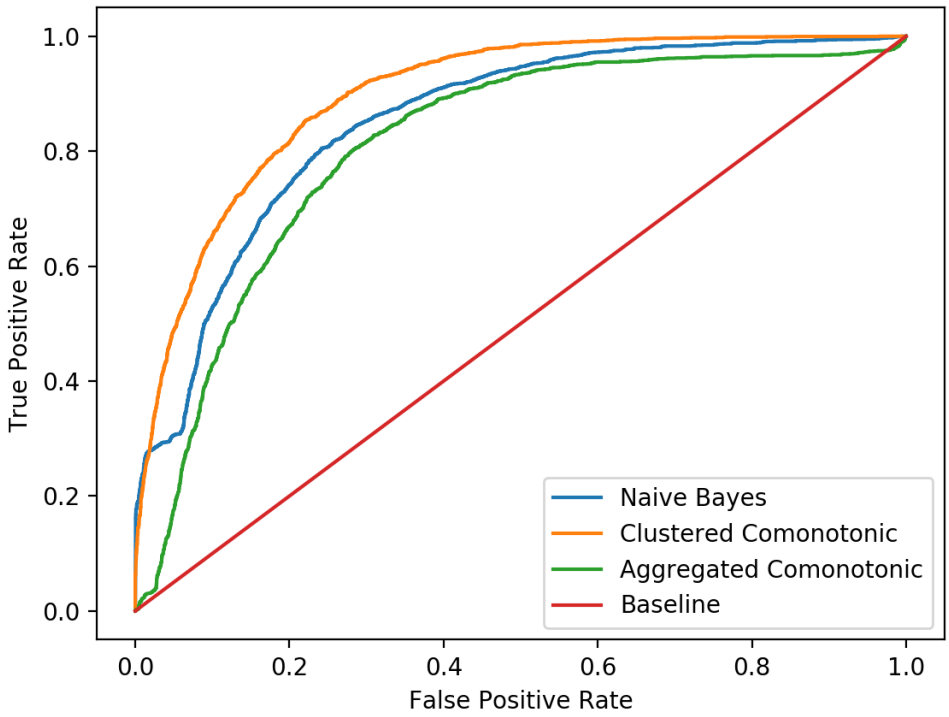
When it comes to the evaluation of a binary classification model, we can count on **AUC-ROC Curve** where AUC stands for Area Under the Curve and ROC stands for Receiver Operating Characteristics. **[5]**

An ROC Curve is a probability curve which shows the performance of a classification model at different classification thresholds. A classification threshold is also called a decision threshold, which outputs "positive" if a value (between 0 and 1) is above the threshold. The $x-axis$ of the curve is the False Positive Rate (FPR) while the $y-axis$ is the True Positive Rate (TPR). TPR is a synonym of recall, sensitivity or probability of detection **[6]**. The formula is $TPR = \frac{TP}{TP+FN}$. FPR is also called probability of false alarm **[6]**. The formula is $FPR = \frac{FP}{FP+TN}$. Before plotting the ROC curve, we train the model and compute the probability distribution over negative (class 0) and positive (class 1) for each item in the test set. Then we pass in the labels for the test set (i.e. y_{test} which is a vector of 0 and 1) and the probability of "positive" for each item. When reducing the classification threshold, more items will be classified as positive. In this way, both FP and TP increase. Therefore, the ROC curve starts from $(0,0)$ and ends at $(1,1)$. We also set the baseline

which is the line segment connecting $(0, 0)$ and $(1, 1)$. This can be achieved by a random classifier. The higher the ROC curve is above the baseline, the better the model's performance.

When it comes to AUC, as the name suggests, it uses numerical methods to measure the area underneath the ROC curve. It provides a measure of aggregate performance over all classification thresholds. We can interpret it as the probability that the model ranks a random positive item higher than a random negative item. Thus, an excellent model has AUC near to 1. On the contrary, if a model has AUC close to 0, then it just runs in the opposite direction since it predicts 0s and 1s reciprocally. The worst case is when the AUC is 0.5 and the ROC curve approximately coincides with the baseline, indicating that the model has no class separation capacity at all.

Since the Adult Census Income is a binary classification task, I use AUC-ROC Curve to check the performance of the models. Using the same set of hyper-parameters, I get the following plot:



The AUC values for the three models are: 0.851 for Naive Bayes, 0.897 for Clustered Comonotonic, 0.808 for Aggregated Comonotonic.

Analysis of results

All of the three methods performed bad on those items of class 1. Such a large gap between the precision on each class should be attributed to the unbalanced dataset in which items of class 0 are over three times the items of class 1. It is worth noting that the recall of Naive Bayes classifier on class 1 is only 0.29, which badly influenced the F1 score on class 1. In terms of the overall accuracy, Clustered Comonotonic > Naive Bayes > Aggregated Comonotonic. This ranking can be verified by the AUC-ROC Curve since the better the model performs, the quicker the curve rises, and hence the larger the area under the curve. Therefore, we conclude that clustered comonotonic model outperforms Naive Bayes on this dataset.

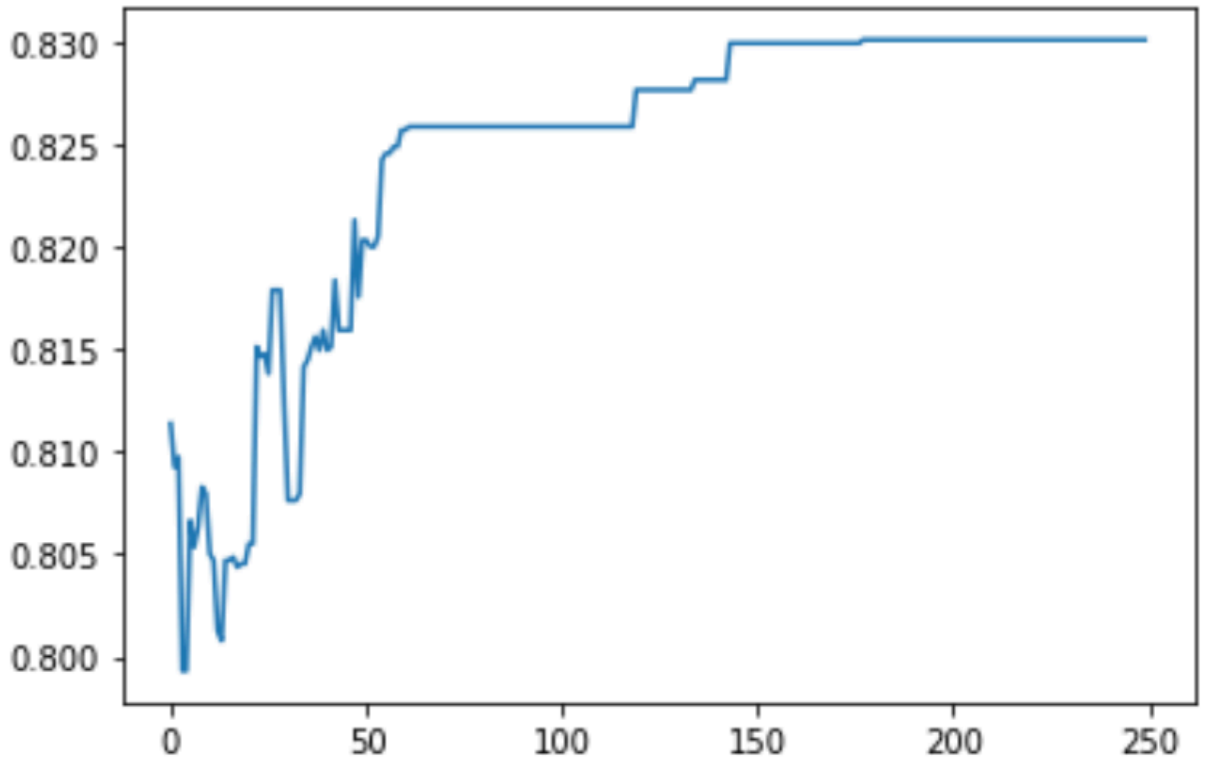
Discussion for Discretisation

In the experiments above, I use the mean and standard deviation to discretise the data into 8 categories and it works fine for the features which approximately follow normal distribution. However, we may come up with some questions. What if we discretise each continuous feature into different number of bins? To what extent would the discretisation method influence the model's performance? Of course, applying grid search to test the accuracy of all possible combinations of number of bins will definitely find the optimal one. However, it is of $O(k^m)$ where k is the maximum number of bins and m is the number of continuous features. Thus, grid search is not applicable for high dimensional datasets. Another good method is random search proposed by Bengio **[7]** In this part, I make some "directional" modifications to random search which incorporate the ideology of Simulated Annealing algorithm to find an optimal combination of the number of buckets for all continuous features. For simplicity, I conduct two sets of experiments. In the first set, the bins are of equal *length*, just as what we do when creating a histogram. In the second set, the bins are of equal *size*, or in other words, quantile-based discretisation. In each set, assume each continuous feature X_i can be discretised into i_k categories where i_k is a natural number between 2 and 10. I initialise them randomly. For the sake of reducing the demand for computing power, I use the traditional method of transition in each iteration. In other words, in the $i's$ iteration, suppose the accuracy in the previous iteration is acc_{i-1}, we change the number of bins for one continuous feature, and then apply clustered comonotonic model to the preprocessed data and compute a new accuracy acc_i on the test set. If $acc_i > acc_{i-1}$, then just switch to this new combination of bins. Otherwise, the transition probability is $exp(-\frac{acc_{i-1}-acc_i}{T})$ where T is a temperature and it should be gradually lowered during the process. Here are the results of the application of Simulated Annealing algorithm on the discretisation optimisation task.

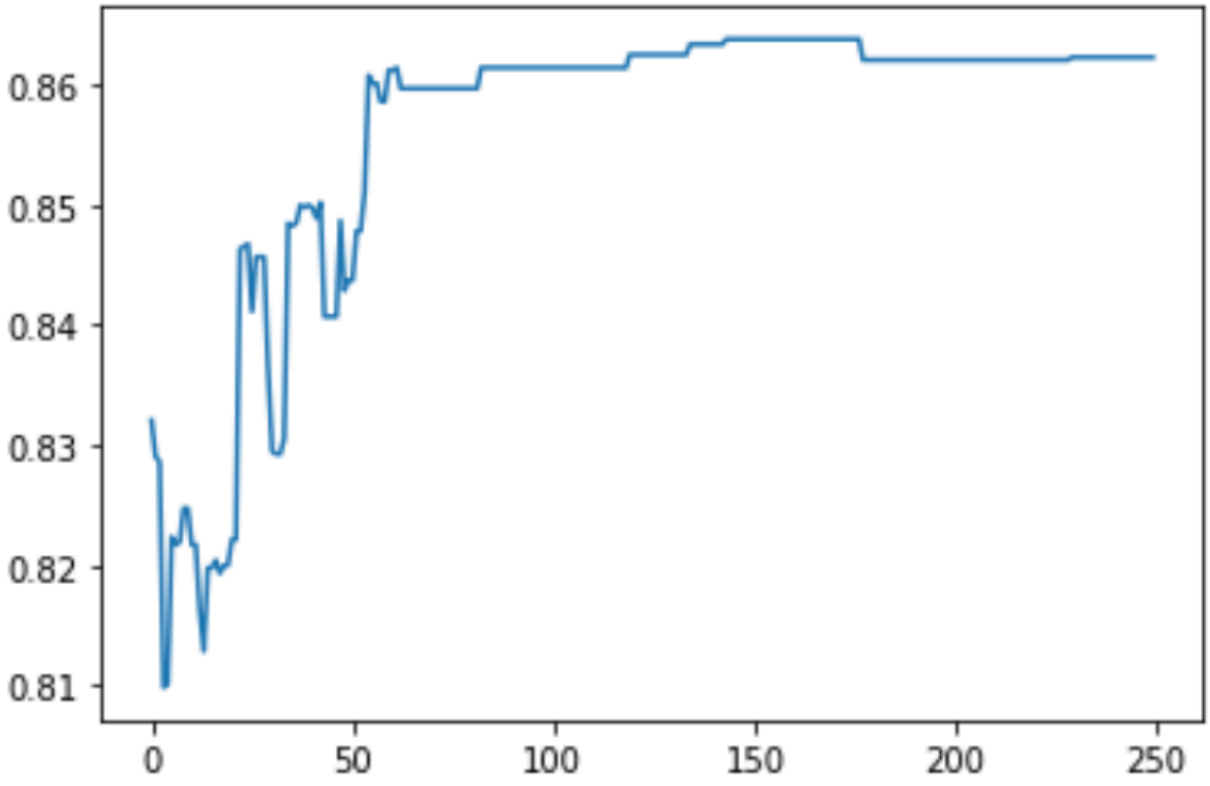
Bins of Equal Length

I use the function $pandas.cut()$ in Python to discretise the data into bins of equal size. The hyper-parameters are: $maximum\ iteration: 250; initial\ temperature: 100; annealing\ schedule: 10$. I reduce the temperature to $1/10$ after every 10 iterations. The overall accuracy and class-wise accuracies are shown below.

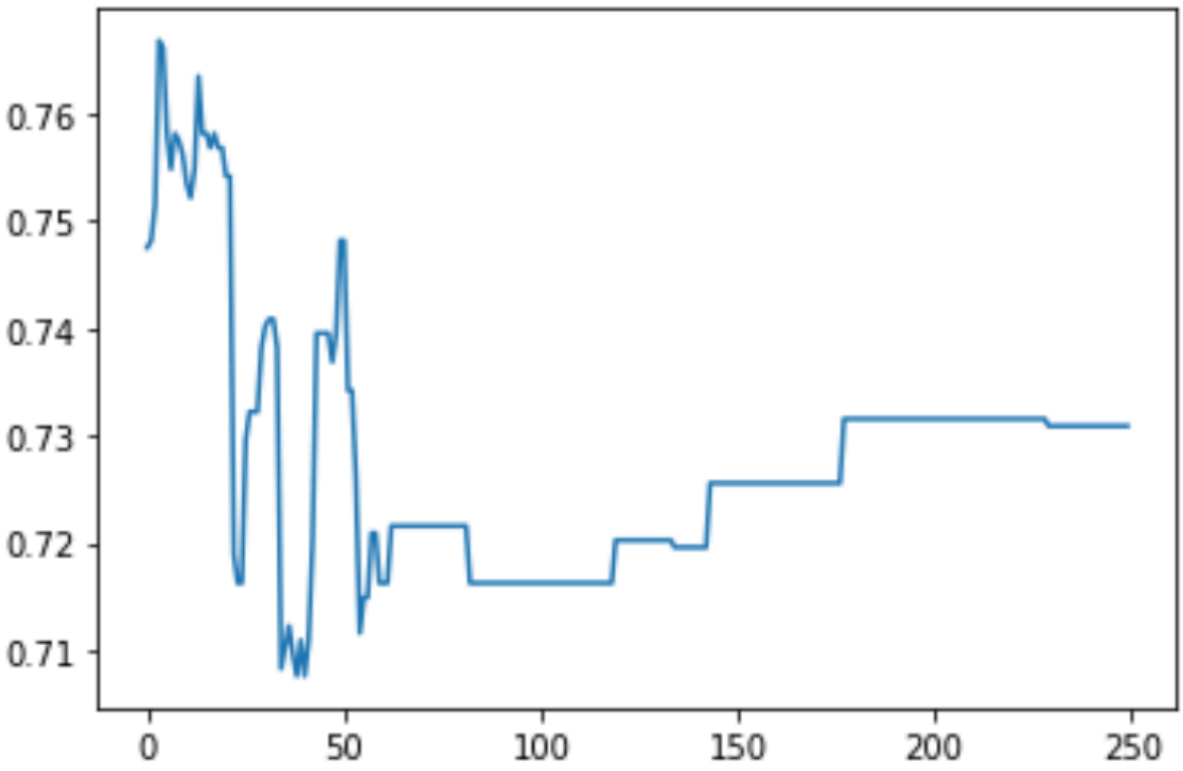
Overall Accuracy



Class 0 Accuracy



Class 1 Accuracy

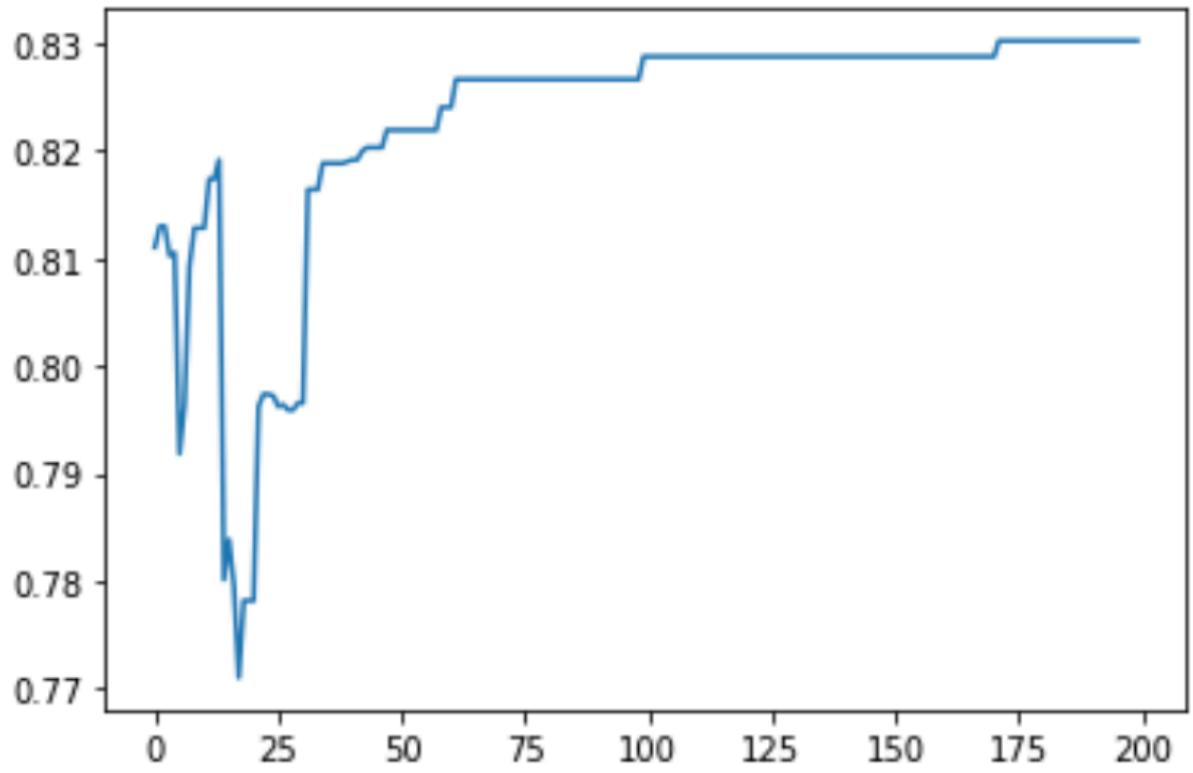


Bins of Equal Size

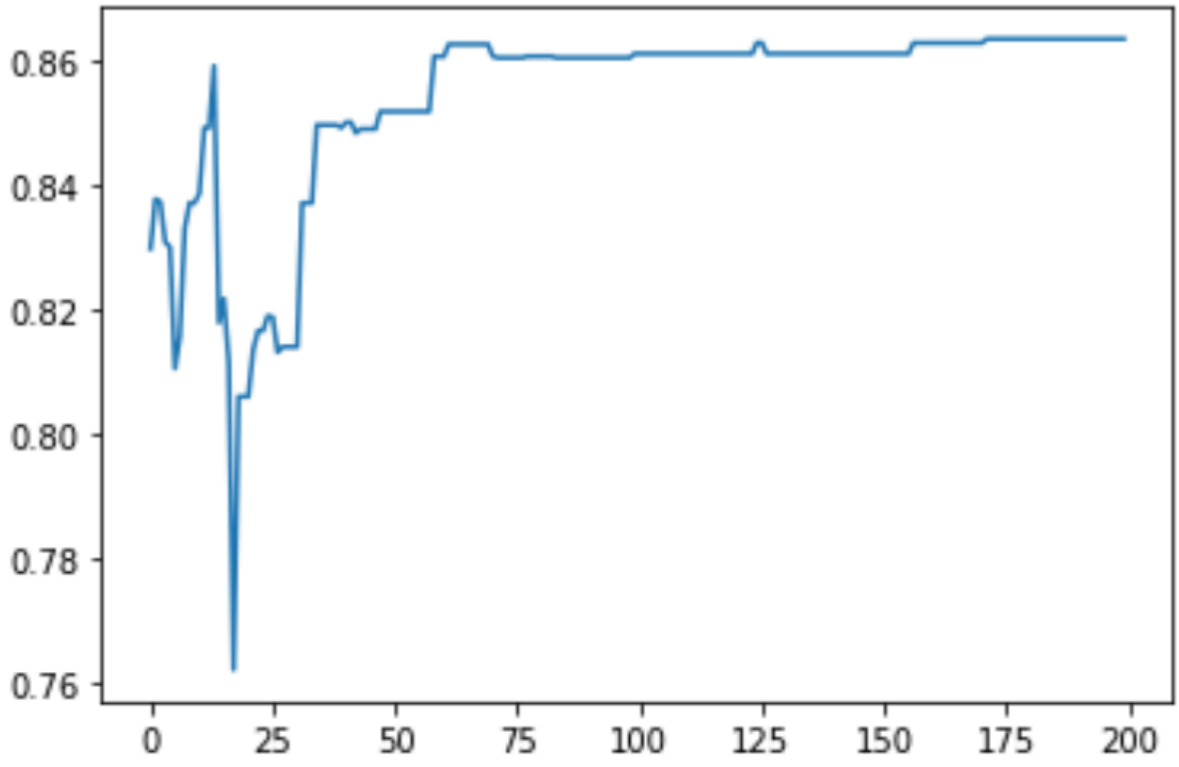
I use the function $pandas.qcut()$ in Python to discretise the data based on quantiles. The hyper-parameters are: $maximum\ iteration: 200$; $initial\ temperature: 100$; $annealing\ schedule: 5$. I reduce the temperature to $1/10$ of the original once every 5 iterations.

The plot of overall accuracy and class-wise accuracies are shown below.

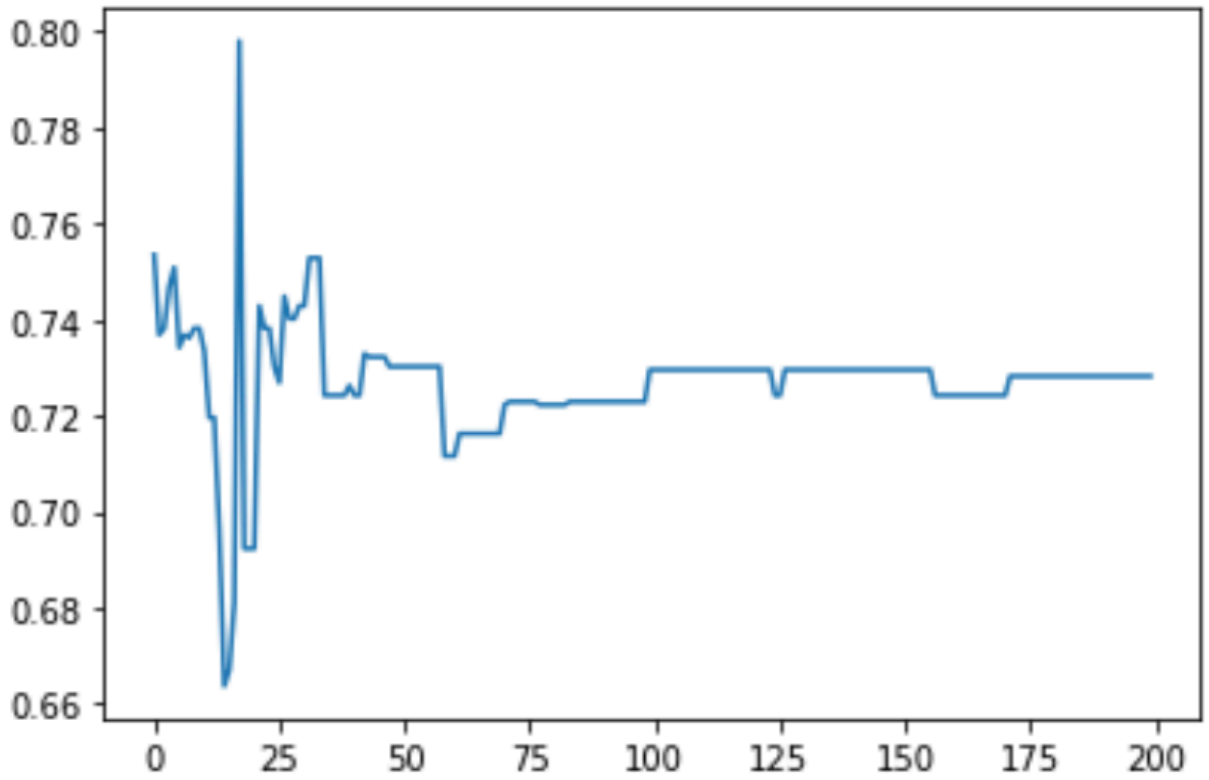
Overall Accuracy



Class 0 Accuracy



Class 1 Accuracy



Analysis of Results

In terms of overall accuracy, the plots show that no matter which type of discretisation method we choose, the overall accuracy oscillated in the first tens of iterations, then it gradually goes up. However, changing the combination of numbers of bins seems insignificant for the optimisation of overall accuracy since the range is even lower than 10%. But at least, we have verified the validity of simulated annealing algorithm in discrete optimisation once more. When it comes to the class-wise accuracy, the performance of boosting on class 0 is better than that on class 1.

The results shown above have provided a further indication on the lower importance of finding the most optimal combination of number of bins for continuous features compared with the importance of modifying the methodology itself. Therefore, even though the application of simulated annealing is an unsuccessful approach in performance boosting, it confirms that the naive approach to discretise based on mean and standard deviation is harmless to the final result.

Conclusion

In this research, I mainly discussed about the ideology of Naive Bayes classifier and proposed using the comonotonicity method to replace the "naive" assumption of features' mutually independence. In this way, we need to discretise the continuous features so that the marginal density for each feature can be represented as an interval. (i.e. $Pr(X_i = x_i)$ can be represented by $(max\{0, \sum_{k=0}^{k=x_i-1} Pr(X_i = k)\}, \sum_{k=0}^{k=x_i} Pr(X_i = k)]$.) The necessity of discretisation does not imply a significant performance boosting when trying to optimise the combination of the number of bins for each continuous feature. In practical applications, I just discretise them into 8

bins using the mean and standard deviation. Then we can transform the computation of joint density into finding the intersection of multiple intervals. Under this ideology, there are two affiliated approaches, namely Aggregated Comonotonic and Clustered Comonotonic. The former one conceives another naive assumption of mutually comonotonic among all features except those discrete and unrankable ones such as gender, race, religion etc. The latter one becomes more reasonable since it just treats those highly correlated features as comonotonic. I use agglomerative nested clustering algorithm to cluster the rankable features where the distance between any two features is $1 - |corr(X_i, X_j)|$. After clustering, the features within a cluster are treated as comonotonic while different clusters are treated as mutually independent.

The three methods are first tested on the MNIST dataset. The results of Aggregated Comonotonic revealed huge flaws when dealing with high dimensional datasets. This might be due to the naive assumption of mutually comonotonicity among all rankable features, which caused a large possibility to encounter an empty intersection of intervals. When empty intersection really happens, the probability is computed using Laplacian correction, which is invariant among any set of feature values with empty intersections. However, the performance of Clustered Comonotonic showed significant improvement to the Naive Bayes Classifier. This is reasonable since we have reshaped each image of size $28 * 28$ into a vector with size $1 * 784$, so there should be a probability distribution of each pixel to contain information of the handwritten digits. We can even regard the clustering process as a searching process to find those pixels with larger probabilities to contain information. The success of Clustered Comonotonic on MNIST indicates the potential of this method in image recognition.

I also tested the three methods on the Adult Census Income dataset. Both the overall accuracies and AUC-ROC curves showed the ranking of superiority to be: Clustered Comonotonic > Naive Bayes > Aggregated Comonotonic. This test is a verification of the advantage of Clustered Comonotonic on low dimensional datasets in which some features are highly dependent.

Bayesian statistics is worthwhile for classification model designing. Although Naive Bayes has been widely used in many applications such as spam filtering, multi-class prediction etc., it is meaningful to think of alternative methods to replace the naive assumption of mutually independence. Empirical evidences have shown that the Clustered Comonotonic method is a good way to solve this issue.

Reference

1. (find a source for the definition of comonotonicity)
2. Hoeffding 1940
3. Frechet 1951
4. find a source for the adult census income dataset
5. find a source for AUC-ROC Curve
6. "Detector Performance Analysis Using ROC Curves - MATLAB & Simulink Example". *www.mathworks.com*. Retrieved 11 August 2016.
7. random search by Yoshua Bengio: http://jmlr.csail.mit.edu/papers/volume13/bergstra12a/bergstra12a.pdf