QF634 APPLIED QUANTITATIVE RESEARCH METHODS
LECTURE 4

Lecturer: Prof Emeritus Lim Kian Guan

Naïve Bayes, kNN, SVM Algorithms

Naïve Bayes Algorithm

The Naïve Bayes classifier (or classification algorithm) is based on Bayes' Rule or Bayes' Theorem. The theorem states that the conditional probability of event $\{Z=1\}$ given event Y has occurred is

$$P(\{Z=1\} \mid Y) = P(\{Z=1\} \cap Y) / P(Y)$$

where probability of event Y in the context of a universe set Ω in which Y resides is P(Y), and $P(\{Z=1\}\cap Y)$ denotes the probability of event that $\{Z=1\}$ and Y occurred jointly.

Example. Consider the bag with the same 100 balls. 10 of the 30 blue balls are painted with a black dot, 20 of the 30 red balls are painted with a black dot, and 30 of the 40 green balls are painted with a black dot. Let $\{Z=1\}$ represent the event that the ball drawn contains a black dot. $P(\{Z=1\}) = 0.6$ since altogether 60 balls have black dots.

Let Y represent the event that a blue ball is drawn. Suppose in a drawn ball, we are given the information that it is blue, then the probability that it also carries a black dot is $P(\{Z=1\}|Y) = 1/3$ or 10 out of 30 blue balls. $P(\{Z=1\}|Y)$ can also be computed as $P(\{Z=1\} \cap Y) / P(Y) = 0.1/0.3 = 1/3$ since $P(\{Z=1\} \cap Y) = 10/100$ and $P(Y) = 30/100$.

Naïve Bayes Algorithm

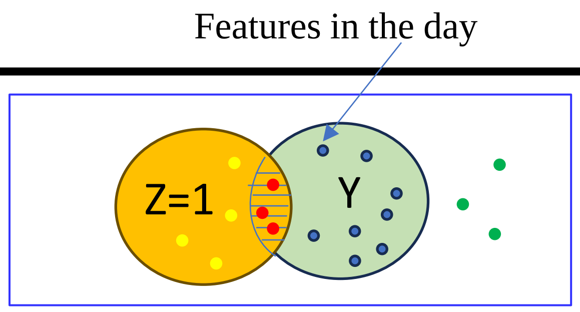
- Think of event $\{Z=1\}$ as a class/type, e.g., those who are marksmen, and event $\{Z=0\}$ as the rest of the universe, i.e., those who are not marksmen.
- Z is a random variable that can take the value 1 or else 0, i.e., there is a binary classification. Bayes' Theorem can be expressed as:

$$P(Z|Y) = P(Y|Z) \times P(Z) / P(Y) = P(Y|Z) \times P(Z) / [P(Y|Z) \times P(Z) + P(Y|Z^c) \times P(Z^c)]$$

where Z^c is complement of event Z, i.e., $Z^c = \Omega \setminus Z$ (Ω less Z), and $P(Y) = P(Y \cap Z) + P(Y \cap Z^c)$. In our notations, $Z^c \equiv \{Z=0\}$. This latter version of Bayes' Theorem is useful in the context of updating prior information or probability of Z, $P(Z)$.

- Example. Medical testing, where a patient has apriori (without any testing for further information) a probability $P(Z) = x$ of having the medical condition Z. x (%) could be obtained from some statistical averages over reports of such medical conditions in various countries. A medical test on the patient could produce information Y, e.g., a positive test case. It may not be easy to obtain total information across all countries of the % of all tests producing Z and Y, i.e., $P(Z \cap Y)$. However, it may be easier to obtain summary reports on $P(Y|Z)$ and $P(Y|Z^c)$, e.g., percentage of cases that have Z and tested positive, and percentage of non-Z and tested positive (false positive).

Naïve Bayes Algorithm



- The Naïve Bayes classification algorithm works using the Bayes Theorem:

$$P(Z|Y) = P(Y|Z) \times P(Z) / P(Y)$$

where Z =1 or else 0 are the two class types we seek for a case under testing and Y is some information/features or evidence we can use to estimate/predict Z.

- Suppose it is a prediction of next stock price index change employing sentiment analysis based on density of mood words prevalent in stock market reports, assuming words like “optimistic”, “growth”, “boom”, etc. would more likely lead to stock index rises, while “pessimistic”, “poor”, “cautious”, etc. would likely lead to stock index drops.
- Let {Z = 1} be the event of next stock price index increase. Therefore {Z = 0} is the event of next stock price index decrease. (Assume next stock price index remaining constant has zero probability or let {Z=0} include case of no change.)
- Let features $F_1, F_2, \ldots, F_n$ be mood words in a particular day. These are used for predicting if the stock price index will rise or fall the next day.
- All the features found in a day constitute event $\{F_1, F_2, \ldots, F_n\} = Y$, the evidence or signal for the day. More convenient to denote elements F_k collected in set as in Venn Diagram.
- A feature or mood word can occur many times on a day due to many reports or appearing many times in a single report, but we could just count it as one occurrence for that day. Red dots are mood words that coincide with Z=1 in that day. Blue dots are mood words that coincide with Z=0 in that day. Yellow dots and green dots are words not captured in the Y information vocabulary.

Naïve Bayes Algorithm

$$P(Z|F_1, F_2, \ldots, F_n) = P(F_1, F_2, \ldots, F_n\,|Z) \times P(Z) \,/\, P(Y) \quad \text{where } Y = \{F_1, F_2, \ldots, F_n\}$$

- Naïve Bayes typically naively makes a simplifying assumption. The conditional probability of $P(F_j\,|Z)$ is assumed to be independent of any other feature F_k, $k \neq j$. In other words, for any j,

$$P(F_j\,|Z, F_1, F_2, \ldots, F_{j-1}, F_{j+1}, \ldots, F_n) = P(F_j\,|Z).$$

- By Bayes' Theorem,

$$\begin{aligned}
P(F_1, F_2, \ldots, F_n\,|Z) &= P(F_1|\, F_2, \ldots, F_n, Z) \times P(F_2, F_3, \ldots, F_n\,|Z) \\
&= P(F_1|\, F_2, \ldots, F_n, Z) \times P(F_2|\, F_3, \ldots, F_n, Z) \times P(F_3, F_4, \ldots, F_n\,|Z) \\
&= P(F_1|\, F_2, \ldots, F_n, Z) \times P(F_2|\, F_3, \ldots, F_n, Z) \times P(F_3|\, F_4, \ldots, F_n, Z) \times P(F_4, \ldots, F_n\,|Z) \\
&= \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
&= P(F_1|\, F_2, \ldots, F_n, Z) \times P(F_2|\, F_3, \ldots, F_n, Z) \times P(F_3|\, F_4, \ldots, F_n, Z) \times P(F_4|\, F_5, \ldots, F_n, Z) \\
&\qquad \times P(F_5|\, F_6, \ldots, F_n, Z) \times \ldots\ldots \times P(F_{n-1}|\, F_n, Z) \times P(F_n\,|Z)
\end{aligned}$$

Naïve Bayes Algorithm

- Using the simplifying assumption on the last line,

$$P(F_1, F_2, \ldots, F_n \mid Z) = P(F_1 \mid Z) \times P(F_2 \mid Z) \times P(F_3 \mid Z) \times P(F_4 \mid Z) \times P(F_5 \mid Z) \times \ldots\ldots \times P(F_{n-1} \mid Z) \times P(F_n \mid Z)$$

$$= \prod_{j=1}^{n} P(F_j \mid Z)$$

Then, $P(Z \mid F_1, F_2, \ldots, F_n) = \prod_{j=1}^{n} P(F_j \mid Z) \times P(Z) / P(Y)$ (4.1)

- Since there are only two classes/types {Z=1} and {Z=0}, we can use the training data set to estimate $\hat{P}(Z = 1)$ and $\hat{P}(Z = 0)$ respectively, where $\hat{P}(Z = 1) + \hat{P}(Z = 0) = 1$.
- Similarly, for all features $\{F_1, F_2, \ldots, F_n\}$ in the training data set, we can estimate $\hat{P}(F_j \mid Z = 1)$ (percentage of occurrences of F_j when next day stock price index increases) and $\hat{P}(F_j \mid Z = 0)$ (percentage of occurrences of F_j when next day stock price index decreases or stays constant).
- Now, suppose a sample point $Y^* = \{F_1, F_3, \ldots, F_m\}$ ($m < n$) from the test sample is chosen. We suppose we do not know the test point's Z value (the type) and want to use Naïve Bayes algorithm to predict its Z value. Then

$$P(\{Z=1\} \mid F_1, F_3, \ldots, F_m) = \hat{P}(F_1 \mid Z = 1) \times \hat{P}(F_3 \mid Z = 1) \ldots \times \hat{P}(F_m \mid Z = 1) \times \hat{P}(Z = 1) / P(Y^*)$$

and $$P(\{Z=0\} \mid F_1, F_3, \ldots, F_m) = \hat{P}(F_1 \mid Z = 0) \times \hat{P}(F_3 \mid Z = 0) \ldots \times \hat{P}(F_m \mid Z = 0) \times \hat{P}(Z = 0) / P(Y^*).$$

Naïve Bayes Algorithm

- P(Y*) does not need to be computed above as it is the same for both the quantities above. We find

$$\text{Max } \{ P(\{Z=1\} \mid F_1, F_3, \ldots, F_m) , P(\{Z=0\} \mid F_1, F_3, \ldots, F_m) \}$$

  given Y*. The **predictor (of class Z, whether 1 or 0)** for this test sample point with Y* is

$$\arg\max_{Z=1,0} P(\{Z\}|F_1, F_3, \ldots, F_m) = \arg\max_{Z=1,0} \prod_j \hat{P}(F_j|Z) \times \hat{P}(Z) \qquad (4.2)$$

  Accuracy of the algorithm is evaluated based on the performance of predictors on each test data point.

- There are several methods for estimating $\hat{P}(F_j|Z=1)$ and $\hat{P}(F_j|Z=0)$ for every j in the training data set, depending on what we assume is the probability distribution of the features F_j.
- Some common distribution assumptions are the multinomial, the categorical, and Gaussian/normal distributions. The Naïve Bayes predictor derived from each of these distributions of the features are in turn called the Categorical NB, the Gaussian NB, and the Multinomial NB.

Naïve Bayes Algorithm

- In the sentiment analyses using mood words as features, it is natural to assume that the features F_j, j=1,2,…,n follow the **categorical distribution**. This means that for any sample point (day), given Z, the probability of a feature F_j occurring is $P(F_j|Z)$ and the probability of the feature not occurring is $1 - P(F_j|Z)$. Thus, any feature F_1 through F_n can either occur or not occur. The categorical variable of F_j or no F_j can be hot-encoded as dummy variable 1 or 0. If there is only one feature in the dataset, then this is also a Bernoulli Naïve Bayes. In any one day, different categorical variables F_1, F_2, etc. can occur together.
- For example, suppose we have 700 days in our study for the training set and 300 other days for the test set. Suppose the dictionary contains 100 mood words. If in the training set of 700 days, 500 have Z=1 or index rise the day after, and 200 have Z=0 which represents index decrease or staying constant the day after. Of the 500, 200 days have reports containing "optimistic", 100 days have reports containing "growth", and so on.
- For estimating $\hat{P}(F_j|Z = 1)$, we can use their frequency counts of number of days of occurrence of F_j /total number of days given Z in the training sample. Here, $\hat{P}(F_j = \text{"optimistic"}\ |Z = 1) = 200/500 = 0.4$. $\hat{P}(F_j = \text{"growth"}\ |Z = 1) = 100/500 = 0.2$. Of the other 200 days, 4 days have reports containing "optimistic", and 10 days have reports containing "growth", and so on. Then, $\hat{P}(F_j = \text{"optimistic"}\ |Z = 0) = 4/200 = 0.02$. $\hat{P}(F_j = \text{"growth"}\ |Z = 0) = 10/200 = 0.05$. Note that both "optimistic" and "growth" can occur together in any one day.

Naïve Bayes Algorithm

- Sometimes, the probability estimates may be too small or close to zero and can make the prediction in Eq. (4.2) difficult because of the nearly zero probability on the right-side. To make the numerical computation more robust, a smoothing parameter $\alpha > 0$ (usually ≤ 1) is added in the estimation of $\hat{P}\left(F_j \mid Z = 1\right)$ and $\hat{P}\left(F_j \mid Z = 0\right)$. For example, $\hat{P}\left(F_j = \text{"boom"} \mid Z = 0\right)$ = $(1+\alpha)/(200+\alpha n)$ where n is the number of features. This adjustment can be non-negligible when n is small.

- In the general execution of Eq. (4.2), we assume that the training set is large enough so that $\hat{P}\left(F_j \mid Z\right) > 0$ for every possible F_j (in the dictionary) in each of condition Z = 1 or Z = 0. We also assume that in the test data set (and foreseeable generalized data set), observed Y** for one day or observed dictionary mood words for that day consisting of F_5, F_7 , F_8,…, F_m for example have computed $\hat{P}(F_5 \mid Z = 1)$, $\hat{P}(F_7 \mid Z = 1)$, $\hat{P}(F_8 \mid Z = 1)$,…, $\hat{P}(F_m \mid Z = 1)$, and also $\hat{P}(F_5 \mid Z = 0)$, $\hat{P}(F_7 \mid Z = 0)$, $\hat{P}(F_8 \mid Z = 0)$,…, $\hat{P}(F_m \mid Z = 0)$ obtained from the training set computations. $\hat{P}(Z = 1)$ and $\hat{P}(Z = 0)$ are also obtained from the training data set. Then (4.2) can be used to predict the test set Z (whether 1 or 0) based on these test set features Y**.

$$\arg\max_{Z=1,0} P(\{Z\} \mid Y^{**}) = \arg\max_{Z=1,0} \prod_j \hat{P}\left(F_j \mid Z\right) \times \hat{P}(Z) \qquad (4.2)$$

Naïve Bayes Algorithm

- Suppose there are other non-natural language features G_j that are continuous, such as volatility (j=1), volume traded on index futures (j=2), lagged index change (j=3), etc., so that it is not feasible to count category frequency to estimate $\hat{P}(G_j|Z)$. In such a situation, we can use the Gaussian Naïve Bayes (GNB) method which employs the Gaussian or normal probability distribution. GNB assumes that $P(G_j|Z)$ (Z=1 or Z=0) follows a Gaussian/normal probability distribution for features j=1,2,…,n :

$$P(G_j|Z) = \frac{1}{\sqrt{2\pi\sigma_{zj}^2}} \exp\left(-\frac{1}{2}\left[\frac{G_j - \mu_{zj}}{\sigma_{zj}}\right]^2\right)$$

where $\mu_{zj} = E(G_j|Z)$ and $\sigma_{zj} = \sqrt{var(G_j|Z)}$.

- If it is further assumed that these parameters are independent of Z, then $\mu_{zj} = \mu_j = E(G_j)$ and $\sigma_{zj} = \sigma_j = \sqrt{var(G_j)}$, and these can be estimated using the unconditional sample mean and sample standard deviation of the feature j values. And if the assumption is that these parameters are independent of both Z and j, then $\mu_{zj} = \mu = E(G)$ and $\sigma_{zj} = \sigma = \sqrt{var(G)}$ where no subscripts for G indicate that the sampling mean and variance are taken over all features, across all j.

Naïve Bayes Algorithm

- Then, the predictor (of class Z, whether 1 or 0) for each test sample case with Y (with its associated features $\{G_j\}$) is

$$\arg\max_{Z=1,0} P(\{Z\}|G_1, G_2, \ldots, G_n) = \arg\max_{Z=1,0} \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi\hat{\sigma}_{zj}^2}} \exp\left(-\frac{1}{2}\left[\frac{G_j - \hat{\mu}_{zj}}{\hat{\sigma}_{zj}}\right]^2\right) \times \hat{P}(Z) \qquad (4.3)$$

- Note that with GNB, there is no need to first standardize/normalize the features first (using all of sample data) before training if the test data features also have the same mean and variance as those in the training data since $\hat{\mu}_{zj}, \hat{\sigma}_{zj}$ estimates from the training data would suitably represent parameters from the test data set. However, if the mean and variance from the test data set are different from those in the training data set (even if features from both sets are Gaussian) for example when we are dealing with time-sequenced data, it may be more robust to first standardize the training and the test data features separately before applying the GNB method. After standardizing, the $\hat{\mu}_{zj}, \hat{\sigma}_{zj}$ estimates of the standardized features are 0 and 1 respectively, for both the training and test set data.

- For mixed type of data in features, e.g., some features F_j are categorical while others G_j are continuous variables, we can apply the NB method as follows. Following Eq. (4.1):

$$P(Z|\ F_1, F_2, \ldots, F_u, G_1, G_2, \ldots, G_v) = \prod_{j=1}^{u} P(F_j|Z) \prod_{j=1}^{v} P(G_j|Z) \times P(Z) \,/\, P(Y)$$

  where $Y \equiv \{F_1, F_2, \ldots, F_u, G_1, G_2, \ldots, G_v\}$. The independence of all features conditional on Z is assumed.

- Then the predictor (of class Z, whether 1 or 0) for each test sample case with Y is

$$\underset{Z=1,0}{\arg max}\, P(\{Z\}|F_1, F_2, \ldots, F_u, G_1, G_2, \ldots, G_V) = \arg\max_{Z=1,0} \prod_{j=1}^{u} \hat{P}(F_j|Z) \prod_{j=1}^{v} \frac{1}{\sqrt{2\pi\hat{\sigma}_{zj}^2}} \exp\left(-\frac{1}{2}\left[\frac{G_j - \hat{\mu}_{zj}}{\hat{\sigma}_{zj}}\right]^2\right) \times \hat{P}(Z)$$

Naïve Bayes Algorithm

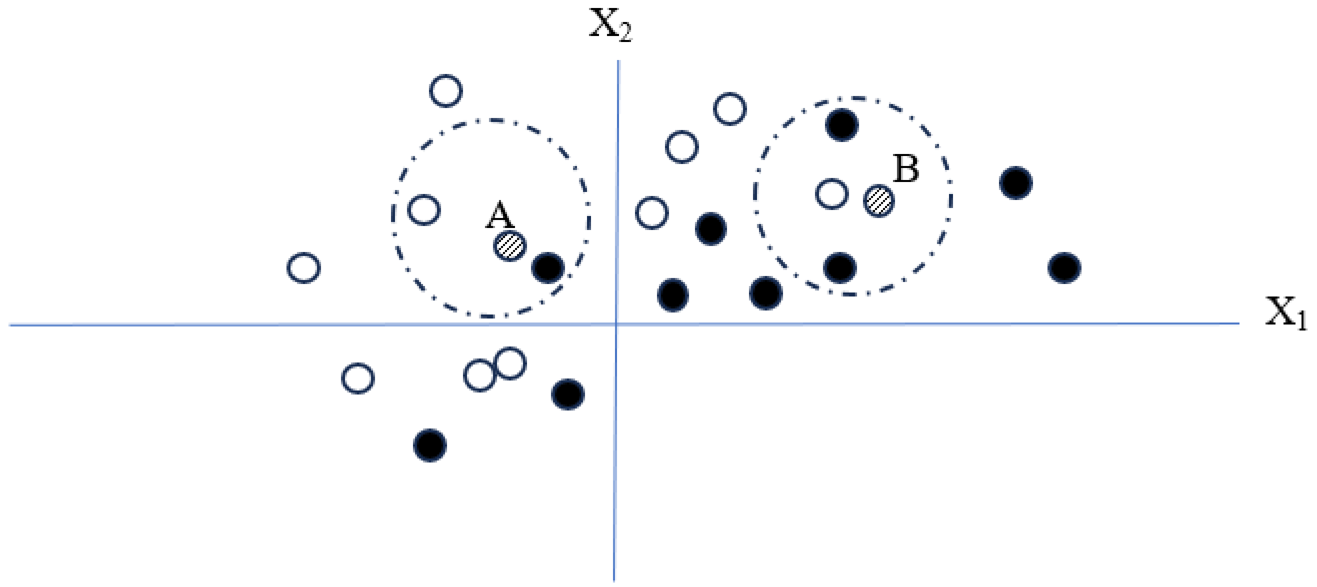
When sample data is split into training and test data sets, the Naïve Bayes algorithm is first applied to the training data set to compute the classification accuracy.

More features can be added or selected so that the training set can be adjusted to optimize the accuracy of a balanced data set.

In GNB, the training data set possibly also provides estimation in time-sequenced data of the feature parameters of means and variances in order to compute the conditional probabilities for predicting the labels or targets.

If there are no competing models or no hyperparameters to optimize, the optimized and estimated parameters are then applied to predict cases from the test features. The predictions are compared with the test labels to find the various performance metrics as well as the ROC AUC.

k-Nearest Neighbors Algorithm or k-NN (kNN)



In the diagram, the training data are 20 sample points represented by the black circles (target value/label 1) and the white circles (target value/label 0). Each sample point has two features represented by values in (X_1, X_2).

The feature values X_k are obtained by transforming the raw features using the standard scalar (where all transformed values X_k have mean 0 and standard deviation 1) or else by MinMaxScaler (where all transformed values X_k lie within [0,1]). This transformation is done over the whole sample for cross-sectional data.

The transformation is useful in k-NN algorithm so that outliers or some features with much larger values on a larger scale would not create bias when the distances are measured.

k-Nearest Neighbor Algorithm

- Sometimes categorical variables are found in available features, e.g., X_3 = male or else X_3 = female. While categorical variables can be a problem in k-NN since they do not operate on a quantitative basis with ratio or else relative comparisons, they may be usable if we provide a suitable dummy.
- In this case, we can expand gender to two dummy features, where $X_{31} = \alpha$ if male and 0 if female, and $X_{32} = 0$ if male and α if female. α should be chosen to be of the same order of magnitude as the rest of the standardized/scaled features. Too small an α close to zero may render the categorical/qualitative feature ineffective. Too large an α may unintentionally outsize the effect of categorical/qualitative feature.
- In the diagram, the test set data are entered one at a time. Sample point A from the test set is shown with the given features in X_{A1} and X_{A2}. The distance between any pair of points u and v is measured typically using the Euclidean distance which is

$$\sqrt{\sum_{k=1}^{2}(X_{uk} - X_{vk})^2}$$

where (X_{u1}, X_{u2}) are the feature values of point u, and (X_{v1}, X_{v2}) are the feature values of point v.

- Sometimes the square of this distance is used as the distance metric. Other distance metrics include the Manhattan distance $\sum_{k=1}^{2}|X_{uk} - X_{vk}|$, the Minkowski distance $\left(\sum_{k=1}^{2}|X_{uk} - X_{vk}|^p\right)^{1/p}$ for some $p \geq 1$, and so on.

k-Nearest Neighbor Algorithm

- If k=1 or using 1 Nearest-Neighbor algorithm, then only the nearest neighbor (of the training sample points) to A, i.e., shortest distance from A, will be picked. In this case, it is a black circle. The prediction for test case A is therefore black or target value/label 1.
- If k=3 or using 3 Nearest-Neighbor algorithm, then only the nearest 3 neighbors to A (of the training sample points) will be picked. In this case, the 3 neighbors are within the dotted circle showing 2 white and 1 black circle. The probability of white circle for the test point is 2/3 and probability of black circle for the test point is 1/3. Since the majority is white (or the higher probability > 0.5), the 3-NN algorithm for test case A is white or target value/label of 0.
- Note that in k-NN algorithm, there is de facto no active computations done on the training data, so the training is sometimes called lazy learning or fitting. The use of the training data is delayed until a test query is made.
- For test sample point B, using 1 Nearest-Neighbor, the prediction of B is white or label 0 since the white circle (of the training sample points) is nearest. But for 3-Nearest-Neighbor, the prediction of B is black or label 1 since within the dotted circle of the 3 closest neighbors (of the training sample points), there is a majority of black circles (or probability 2/3 > 0.5 of black circles).

k-Nearest Neighbor Algorithm

- In using the k-NN algorithm, one important and about the only hyperparameter (there may be other hyperparameters if some other more complicated distance metric is used) to choose is k.
- In general, for k-NN algorithm, k is chosen to be odd to avoid a tie. We may have to re-run the algorithm several times for different hyperparameter values of k (tuning k) before we reach a satisfactory prediction or classification accuracy and good AUC.
- When k is decreased toward 1, the prediction for each case becomes unstable as prediction is based only on what happens to be the nearest neighbor of the test sample point. The prediction is myopic and may miss the correct affiliation as it may happen that there are more of the incorrect categories that are closer.
- But as k is increased, the prediction becomes more stable due to majority voting/averaging phenomenon, but this prediction can worsen as k becomes too large when the expanding neighborhood gets in more of both categories or types of the sample points so that it can flip sides.
- The hyperparameter of k could be selected using a validation data set. Another way of tuning k is to implement cross-validation, e.g., splitting the data set into 5 equal-sized groups and using 4 for training k while keeping 1 as the validation set.

k-Nearest Neighbor Algorithm

- The k-NN algorithm becomes slower as the number of features used for computing distances increases.
- In the case of regression where k-NN is used to predict a target value based on the test sample point's features, the average of the values of the training cases in the nearest-neighbor dotted circle will be used as the prediction.
- Note that k-NN is a non-parametric algorithm. There is no need for any assumption of probability distributions on the features, and therefore avoids distributional misspecification errors.
- If a test point sits in the k-neighborhood of m number of white circles and k-m of black circles, then the predicted probability of a white circle of the test point may be construed as m/k while the probability of a black circle as predicted value is 1 – m/k. Typically m/k > 0.5 (a hyperparameter threshold) implies prediction of white, otherwise black.
- But for constructing the Receiver Operating Characteristics (ROC) Curve in a binary classification, the hyperparameter threshold of white circles can be varied between 0 and 1 so that only when the predicted probability of white is higher than the threshold, the prediction is white, otherwise black. Thus, ROC can be constructed for the k-NN algorithm.

Support Vector Machine

We employ SVM for the loan classification – a popular ML algo, although it is computationally intensive when data set gets large

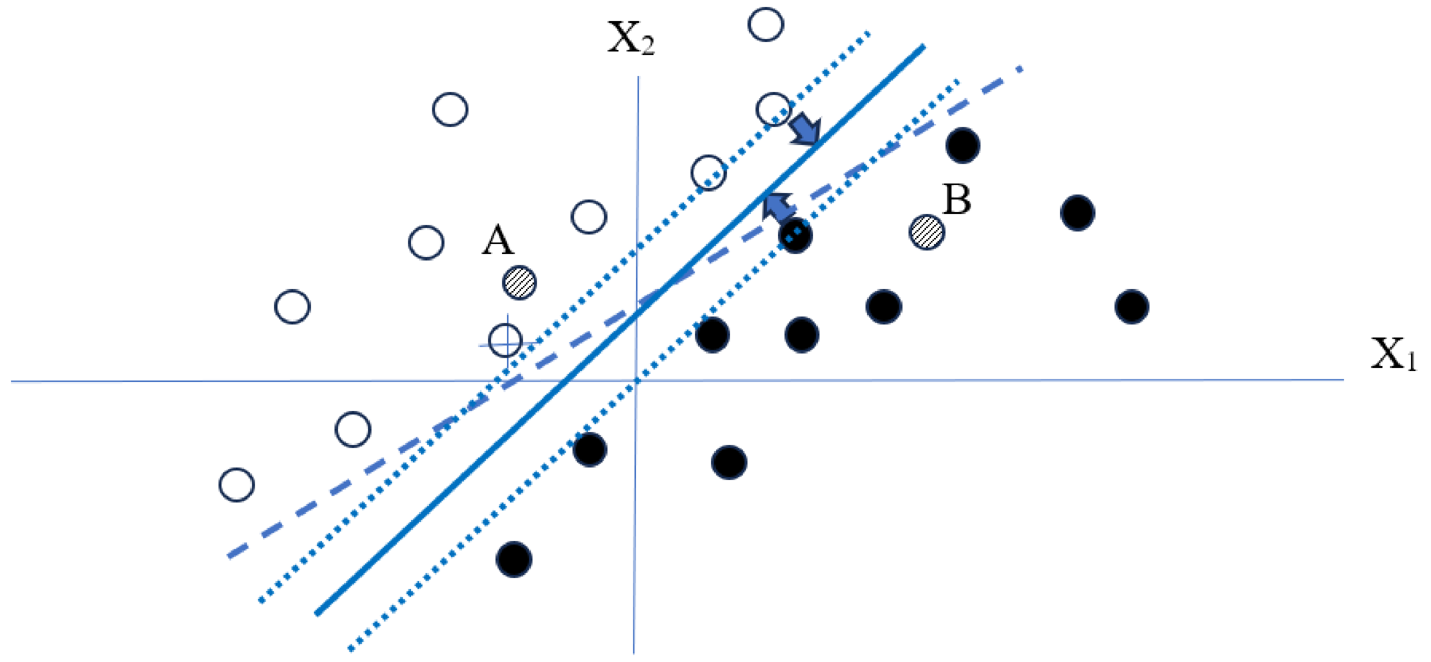


Figure 4.2

As shown in Figure 4.2, the maximum separation is attained via the solid boundary line with positive slope. Both the colored circles are clearly separated.

There, the nearest white circle to the boundary and the nearest black circle to the boundary have equal distances represented by the arrows.

The linear SVM provides for a solid line boundary with the maximum margin (sum of perpendicular distances from the nearest feature points on either class to the line).

Any shift in the boundary (see dashed line) would end up having at least one colored circle being closer to the boundary, e.g., the white circle shown with a cross.

The latter means that the margin or gap has shrunk and is not an optimal boundary.

Support Vector Machine

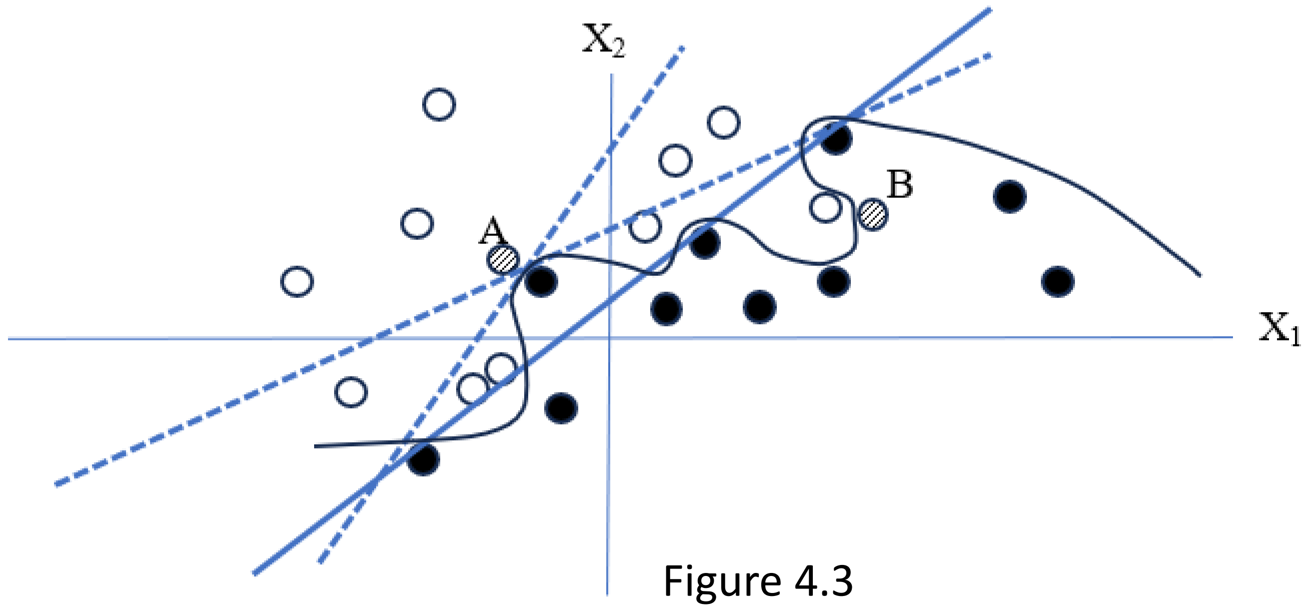


Figure 4.3

Two ways when outright linear separation is not possible.

- Firstly, we could still apply linear boundary as in the solid line but allow some classification errors in training.
- Secondly, we may be able to find a polynomial function (shown in the curved line) to separate the colored circles.

Support Vector Machine

Often there are complicated grouping patterns such as the following whereby a transformation must be applied to the features to separate the classes. This transformation involves a higher-dimensional feature space. Such a transformed feature space with a higher dimension than the original one for the purpose of separating the classes/types is often called a "kernel trick".

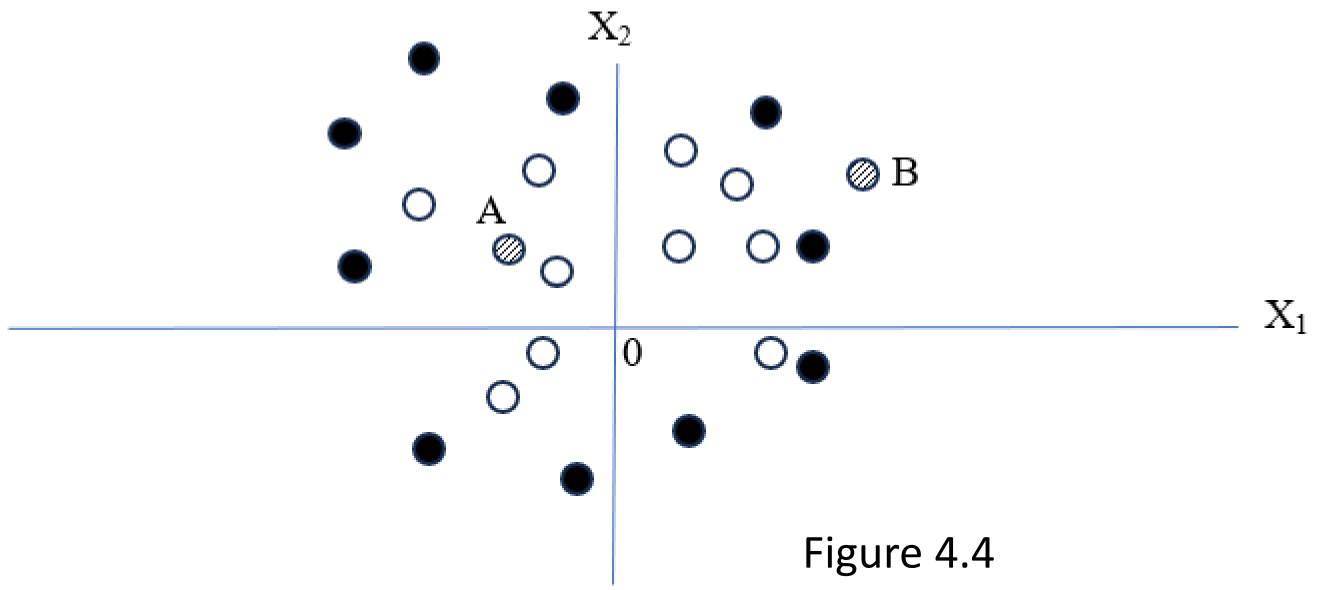
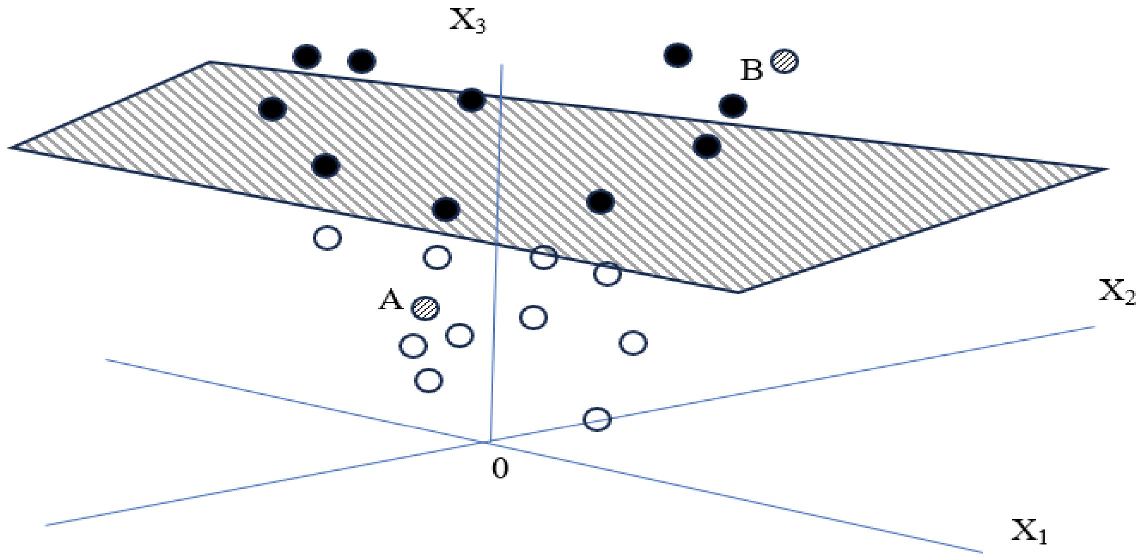


Figure 4.4

Support Vector Machine

As a specific (need not be optimal) solution, we could use the transformation $\phi(X_1^{(j)}, X_2^{(j)}) = (X_1^{(j)}, X_2^{(j)}, X_3^{(j)})$ for the jth training data set sample point, where the third coordinate is $X_3 = X_1^2 + X_2^2$. By extending the feature space, we can obtain the following 3-dimensional representation of the circles. Now the black circles will have higher X_3 values since they lie on an outer circle in the X_1-X_2 axes. (Drawing may not be exactly to scale.)



In the 3-dimensional representation, a plane (shaded) can be seen to separate the black circles from the white circles below. Clearly now A and B from the test sample can be easily predicted as belonging to the white and black circle groups respectively.

Support Vector Machine

- The theory of SVM can be explained as follows. Suppose a training sample consists of N number of data or sample points. Each sample point has p features and the jth sample point can be characterized in Euclidean space as a p-dimensional vector ($X_1^{(j)}$, $X_2^{(j)}$, $X_3^{(j)}$, …., $X_p^{(j)}$).
- Each point belongs to one of two classes. In the categorical nature of the class/type, there is no loss of generality in assigning numerical value +1 or -1 to the jth sample point category/type, i.e., let Y_j = +1 or –1 depending on which on the binary class/type.

Consider again the two-dimensional features in an earlier example. We want to draw a hyperplane (a straight line in 2-dimension) to separate the black circles (target/label Y_j = –1) from the white circles (target/label Y_j = +1).

Moreover, we want a straight line with the widest margin (wide "street") with the two margin lines (shown as bold lines) touching the support vectors or the points on the lines.

The double-arrowed line in Figure 4.6 shows the extent of the margins.

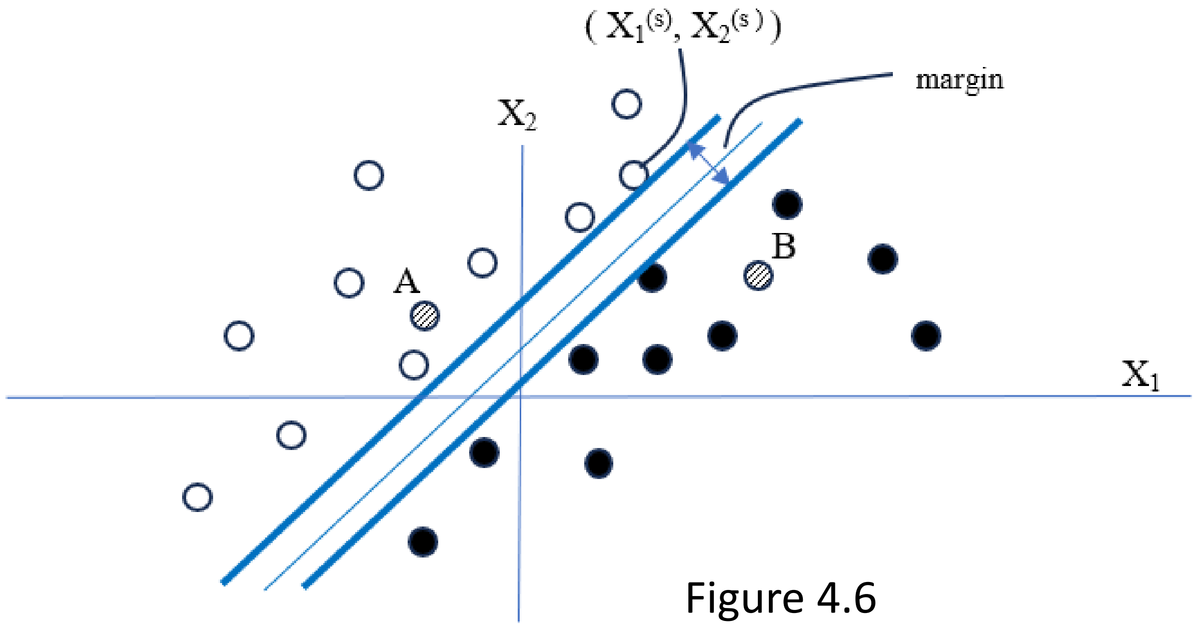


Figure 4.6

Support Vector Machine

- Suppose a training sample consists of N number of data or sample points. Each sample point has p features and the jth sample point can be characterized in Euclidean space as a p-dimensional vector $X_j \equiv (X_1^{(j)}, X_2^{(j)}, X_3^{(j)}, …., X_p^{(j)}) \in R^p$.
- Each point belongs to one of two classes denoted by $Y_j = +1$ or $Y_j = -1$.
- Let $W \in R^P$ be a p-dimensional vector representing coefficients of X, and W_0 denote a constant vector. Suppose the separating hyperplane is $W^T X + W_0 = 0$. Then a unit vector perpendicular to the separating hyperplane is $W/||W||$.
- Now, any two points X_a and X_b on the separating hyperplane obeys $W^T X_a + W_0 = 0$ and $W^T X_b + W_0 = 0$, so $W^T (X_a - X_b) = 0$. But $(X_a - X_b)$ is a vector lying on the hyperplane. Hence W is a perpendicular vector to $(X_a - X_b)$. Thus W is perpendicular to the separating hyperplane. W has length $\sqrt{}(W \bullet W) = ||W||$. Hence $W/||W||$ has unit length.

Cosine law states that in a triangle with sides having lengths a, b, c, with c the largest, then $c^2 = a^2 + b^2 - 2ab \cos \theta$ where θ is the angle between the sides having lengths a and b. But vectors u and v are such that the vector v – u forms the third side of a triangle. Then $||v - u||^2 = ||u||^2 + ||v||^2 - 2\ ||u||\ ||v|| \cos \theta$ by the cosine law. Or, $(v-u) \bullet (v-u) = u \bullet u + v \bullet v = 0$ if and only if (iff) u and v are perpendicular, i.e., $\cos \theta = 0$. Hence $u \bullet v = 0$ iff u and v are perpendicular.

Support Vector Machine

- If X_j is a point in R^p and X_0 is its perpendicular projection onto the separating hyperplane (X_0 is on the separating hyperplane), then $X_j = X_0 + \gamma_j W/||W||$ where $\gamma_j > 0$ is the distance of the perpendicular projection of X_j. Let X_j be a sample point or vector on the positive side or class $Y_j = +1$.
- Now $X_0 = X_j - \gamma_j W/||W||$. Since X_0 is on the separating hyperplane, $W^T X_0 + W_0 = 0$. Thus, $W^T (X_j - \gamma_j W/||W||) + W_0 = 0$. Or, $W^T X_j + W_0 = \gamma_j W^T W/||W|| = \gamma_j ||W||$.

Hence, for the points on the positive side with class $Y_j = +1$,

$$\gamma_j = \frac{W^T X_j + W_0}{\|W\|}$$

- If, however, X_j is a sample point or vector on the negative side or class $Y_j = -1$. Then $X_j = X_0 - \gamma_j W/||W||$ where $\gamma_j > 0$ is the distance of the perpendicular projection of X_j. Now $X_0 = X_j + \gamma_j W/||W||$. Since X_0 is on the separating hyperplane, $W^T X_0 + W_0 = 0$. Thus, $W^T (X_j + \gamma_j W/||W||) + W_0 = 0$. Or, $W^T X_j + W_0 = -\gamma_j W^T W/||W|| = -\gamma_j ||W||$.
- Hence, for the points on the negative side with class $Y_j = -1$,

$$\gamma_j = -\frac{W^T X_j + W_0}{\|W\|}$$

Support Vector Machine

- For all sample points j in the training data set, their absolute perpendicular distances $\gamma_j = \left|\frac{W^T X_j + W_0}{\|W\|}\right|$ from the separating hyperplane should be above or equal to a minimum $\gamma > 0$. Twice γ is the margin of the SVM. Hence,

$$Y_j \frac{W^T X_j + W_0}{\|W\|} \geq \gamma \quad \text{for all j.}$$

- But a constant c times W, W_0 gives $(cW^T X_j + cW_0)/||cW|| = (W^T X_j + W_0)/||W||$. Hence there is no unique solution (W, W_0) linking a support vector X_s to minimum projection distance γ. Suppose the solution γ exists and is unique given the sample data set. We can find a unique solution by fixing $||W|| = 1/\gamma$ relative to the sample data set. This is fixing $||W||\, \gamma = 1$.
- Suppose the solution to a SVM separating hyperplane $W^T X + W_0 = 0$ is

$$\max_{W, W_0, \gamma} \gamma \text{ subject to } Y_j \frac{W^T X_j + W_0}{\|W\|} \geq \gamma \quad \text{for all j.}$$

Support Vector Machine

- For unique solution (W, W_0), the fixing of $||W|| \gamma = 1$ allows the program to be re-written as:

$$\max_{W,W_0} 1/\|W\| \qquad \text{subject to } Y_j\left(W^T X_j + W_0\right) \geq 1 \text{ for all j.}$$

- This is equivalent to a quadratic program:

$$\min_{W,W_0} \frac{1}{2}\|W\|^2 \qquad \text{subject to } Y_j\left(W^T X_j + W_0\right) \geq 1 \text{ for all j.} \tag{4.5}$$

- To solve the constrained optimization shown above more readily, we make use of the Lagrange duality. This involves the Lagrange primal form and the dual Lagrange form to facilitate numerical computations of a solution. The Lagrange duality concept and solution is explained in the following sub-section.

Lagrange Duality Solution

- A numerical solution of W, W_0 (each is p-dimensional) may be too cumbersome as the inequality constraints $\mathrm{Y_j}\left(\mathrm{W^T X_j} + \mathrm{W_0}\right) \geq 1$ needs to be checked for each j (with $p \times 1$ X_j and 1×1 Y_j) given a potential solution (W, W_0).
- A neater solution is to make use of the Lagrangian. Recall program (4.5) is to maximize the margin of the SVM with the separating hyperplane as $W^T X + W_0 = 0$ for some optimal W and W_0. The primal representation of program (4.5) is

$$\min_{W,W_0} \max_{\lambda \geq 0} \frac{1}{2}\|\mathrm{W}\|^2 + \sum_{j=1}^{N} \lambda_j \left(1 - \mathrm{Y_j}\left(\mathrm{W^T X_j} + \mathrm{W_0}\right)\right) \tag{4.6}$$

  where $\lambda_j \geq 0$ (defined to be positive) is a Lagrange multiplier on the inequality constraint $1 - \mathrm{Y_j}\left(\mathrm{W^T X_j} + \mathrm{W_0}\right) \leq 0$. The Lagrange multiplier acts like a penalty if the constraint is violated such as when $1 - \mathrm{Y_j}\left(\mathrm{W^T X_j} + \mathrm{W_0}\right) > 0$.
- The dual Lagrange form of (4.6) is

$$\max_{\lambda \geq 0} \min_{W,W_0} \frac{1}{2}\|\mathrm{W}\|^2 + \sum_{j=1}^{N} \lambda_j \left(1 - \mathrm{Y_j}\left(\mathrm{W^T X_j} + \mathrm{W_0}\right)\right) \tag{4.7}$$

Lagrange Duality Solution

- The dual Lagrange form in (4.7) has a finite solution where
$$\min_{W,W_0\,|\lambda} \quad \frac{1}{2}\|\mathrm{W}\|^2 + \sum_{j=1}^{N} \lambda_j \left(1 - \mathrm{Y_j}\left(\mathrm{W^T X_j} + \mathrm{W_0}\right)\right) \qquad (4.8)$$
is performed with a given λ as a kind of first stage optimization before the second stage optimization over $\lambda \geq 0$ (some elements of λ can be = 0).
- In the first stage solution in (4.8), the first order conditions for minimum are:
$$\frac{\partial L}{\partial W} = W - \sum_{j=1}^{N} \lambda_j \mathrm{Y_j X_j} = 0 \quad \text{and} \quad \frac{\partial L}{\partial W_0} = \sum_{j=1}^{N} \lambda_j \mathrm{Y_j} = 0$$
- Fixing these optimal values for W* in (4.8), we have the second stage optimization in $\lambda \geq 0$:
$$\max_{\lambda \geq 0 | W^*} \quad \sum_{j=1}^{N} \lambda_j - \frac{1}{2} \sum_{j=1}^{N} \sum_{k=1}^{N} \lambda_j \lambda_k \mathrm{Y_j Y_k X_j^T X_k} \qquad (4.9)$$
subject to $\sum_{j=1}^{N} \lambda_j \mathrm{Y_j} = 0$ and $\lambda_j \geq 0$ for all j.

Lagrange Duality Solution

- Obtain W* = $\sum_{j=1}^{N} \lambda_j^* \mathrm{Y_j X_j}$. Note that W* is solution to the separating hyperplane $W^{*T}X + W_0^* = 0$. Note that for uniqueness of solution in W*, W_0*, with setting $\gamma \, \|W^*\| = 1$, and $\mathrm{Y_j}\left(\mathrm{W^{*T}X_j + W_0^*}\right) \geq 1$ for all j, we have support vector(s) on the positive side (Y_j = +1) "margin" hyperplane as $\mathrm{W^{*T}X_j + W_0^*}$ = 1 or $\min\left(\mathrm{W^{*T}X_j + W_0^*} \mid \mathrm{Y}_j = +1\right) = 1$, or $\min\left(\mathrm{W^{*T}X_j} \mid \mathrm{Y}_j = +1\right) = 1 - \mathrm{W_0^*}$.
- Support vector(s) on the negative side (Y_j = –1) "margin" hyperplane are $\mathrm{W^{*T}X_j + W_0^*}$= –1, or $\max\left(\mathrm{W^{*T}X_j + W_0^*} \mid \mathrm{Y}_j = -1\right) = -1$, or $\max\left(\mathrm{W^{*T}X_j} \mid \mathrm{Y}_j = -1\right) = -1 - \mathrm{W_0^*}$. Then $\mathrm{W_0^*}$ can be found as

$$\mathrm{W_0^*} = -\left[\min\left(\mathrm{W^{*T}X_j} \mid \mathrm{Y}_j = +1\right) + \max\left(\mathrm{W^{*T}X_j} \mid \mathrm{Y}_j = -1\right)\right]/2.$$

- The score of any sample point can be measured as $\mathrm{W^{*T}X_j + W_0^*}$ which is the distance of $\gamma_\mathrm{j} = \frac{\mathrm{W^{*T}X_j + W_0^*}}{\|\mathrm{W^*}\|}$ in terms of γ = 1/$\|\mathrm{W^*}\|$, hence γ_j/γ = $\mathrm{W^{*T}X_j + W_0^*}$. This score can be positive (on the positive side of the separating hyperplane) or negative (on the negative side of the separating hyperplane).
- When it comes to prediction of a class given a generalized data point with features X_g (including predictions on test data), the SVM score of the generalized data point is measured using the trained SVM, i.e., score = $\mathrm{W^{*T}X_g + W_0^*}$. If the score is ≥ +1, the point or case is predicted to be in the class Y_g = +1. If the score is ≤ –1, the point or case is predicted to be in the class Y_g = –1.

Non-Linear Kernels in SVM

- So far we have used a linear separating hyperplane, $W^T X_0 + W_0 = 0$ based on the p-dimensional feature vector $X_k \equiv (X_1^{(k)}, X_2^{(k)}, X_3^{(k)}, \ldots, X_p^{(k)}) \in R^p$. Suppose the optimization in (4.9) fails as a linear separating hyperplane cannot be found. In (4.9), the term $X_i^T X_j$ involving the dot product of X_i and X_j is called a linear kernel on X_i and X_j .

  • The polynomial kernel $K(X_i,X_j) = (\gamma X_i \bullet X_j + c)^d$ where the degree of the polynomial d, gamma γ, and c are hyperparameters. Sometimes a simpler version of $\gamma = 1$ or $\gamma = 1/N$ is used as a default case in many programs. As an illustration, suppose point X_i is $(x_1^{(i)}, x_2^{(i)})$ with two dimensions. Let $K(X_i,X_j) = (X_i \bullet X_j + 1)^2$ where d = 2. Then $K(X_i,X_j) = (x_1^{(i)}x_1^{(j)} + x_2^{(i)}x_2^{(j)} + 1)^2 = 2\, x_1^{(i)}x_1^{(j)} + 2\, x_2^{(i)}x_2^{(j)} + 2\, x_1^{(i)}x_1^{(j)}x_2^{(i)}x_2^{(j)} + (x_1^{(i)}x_1^{(j)})^2 + (x_2^{(i)}x_2^{(j)})^2 + 1 = (\sqrt{2}\, x_1^{(i)}, \sqrt{2}\, x_2^{(i)}, \sqrt{2}\, x_1^{(i)}x_2^{(i)}, x_1^{(i)\,2}, x_2^{(i)\,2}, 1) \bullet (\sqrt{2}\, x_1^{(j)}, \sqrt{2}\, x_2^{(j)}, \sqrt{2}\, x_1^{(j)}x_2^{(j)}, x_1^{(j)\,2}, x_2^{(j)\,2}, 1)$. Clearly, the polynomial kernel now expands X_i from two to six dimensions. The higher dimension allows for more space in-between sample point features and hence easier linear separation; this is called the “kernel trick”.

  • We can let the dimension expansion (enlarging feature space) be transformation function $\phi(x_1^{(i)},x_2^{(i)}) = (\sqrt{2}\, x_1^{(i)}, \sqrt{2}\, x_2^{(i)}, \sqrt{2}\, x_1^{(i)}x_2^{(i)}, x_1^{(i)\,2}, x_2^{(i)\,2}, 1)$. Hence polynomial kernel on 2-dimensional X_i, X_j , $K(X_i,X_j) = \phi(x_1^{(i)},x_2^{(i)}) \bullet \phi(x_1^{(j)},x_2^{(j)}) = \phi(X_i) \bullet \phi(X_j)$. The latter is a linear kernel on $\phi(X_i)$, $\phi(X_j)$.

  • Mercer’s Theorem: For symmetrical real-valued $K(X_i,X_j)$, a representation $K(X_i,X_j) = \phi(X_i) \bullet \phi(X_j)$ for function $\phi(.)$ can be found. Mercer, J. (1909), "Functions of positive and negative type and their connection with the theory of integral equations", *Philosophical Transactions of the Royal Society A*, **209** (441–458): 415–446.

Non-Linear Kernels in SVM

Using a non-linear kernel $K(X_i,X_j)$, we are de facto trying to solve for a (linear) separating hyperplane on linear $\phi(X_i)$, $\phi(X_j)$ for some function $\phi(.)$ associated with the non-linear kernel. See (4.10).

$$\max_{\lambda \geq 0 | W^*} \quad \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j \mathrm{Y_i Y_j} \, \phi(X_i) \bullet \phi(X_j) \qquad (4.10)$$

- Another popular non-linear kernel is the Gaussian radial basis function (RBF) $K(X_i,X_j) = \exp(-\gamma \, ||X_i - X_j||^2)$ or $\exp(-\gamma \, (X_i - X_j) \bullet (X_i - X_j))$ where $\gamma = 1/(2\sigma^2)$. Here γ (gamma) > 0 is a hyperparameter and σ need not be constrained to be standard deviation of the standardized features.
- Another non-linear kernel is the sigmoid or hyperbolic tangent kernel $K(X_i,X_j) = \tanh(\gamma X_i \bullet X_j + c)$ where gamma γ and c are hyperparameters. Sometimes a simpler version of $\gamma = 1$ or $\gamma = 1/N$ is used as a default case in many programs.
- In general, when we apply a particular kernel function as in the constrained optimization in Eq. (4.10), we check for the performance results in prediction accuracy and the classification errors on the test sample data set. There is no need to explicitly determine or check the transformation function of $\phi(.)$ in $K(X_i,X_j) = \phi(X_i) \bullet \phi(X_j)$.
- The choice of what type of kernel depends on the nature of the data and the problem. Linear kernel is used when the data is approximately linearly separable. Polynomial kernel is used when the data has a curved border. Gaussian kernel is used when the data appear to have complicated overlaps and no clear boundaries.

Linear Non-Separability SVM

Suppose the optimization in (4.9) still fails as a linear separating hyperplane cannot be found despite suitable transformations of features or feature vector X_i into $\phi(X_i)$ with perhaps a higher dimension. We can see this in Figure 4.7 whereby any transformations $\phi(X_i)$ of a sample point into two dimensions (Z_1, Z_2) could not allow linear separation.

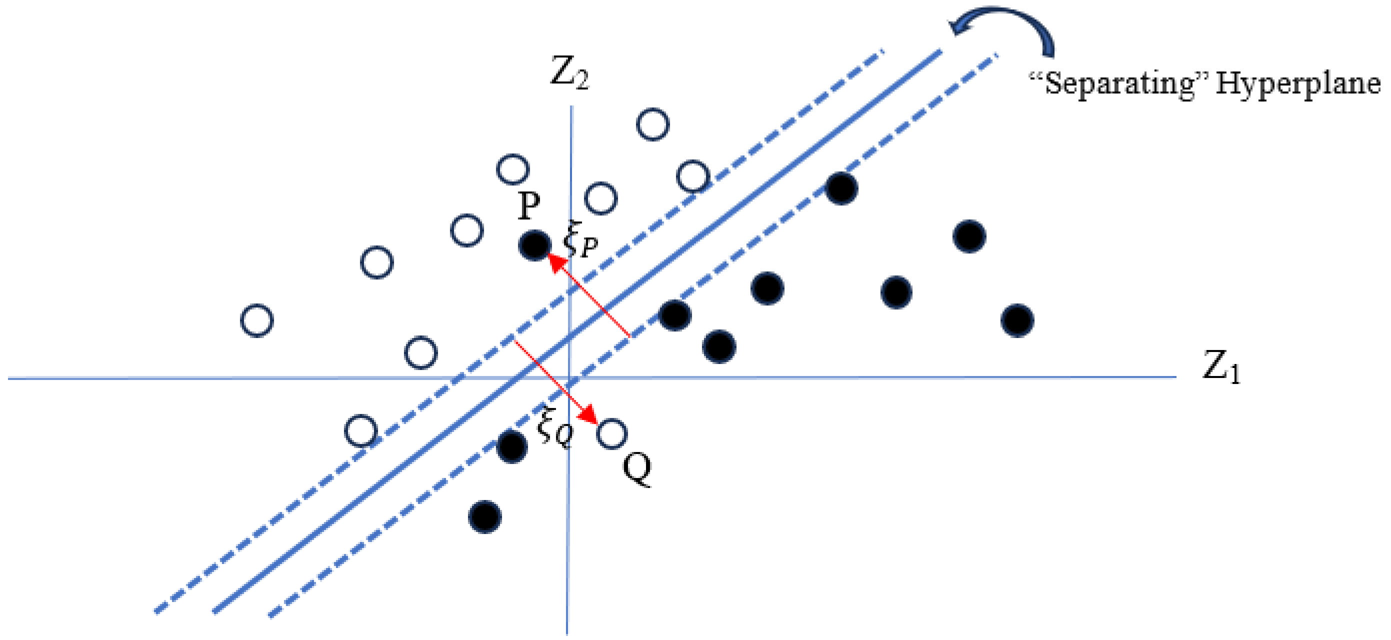


Figure 4.7

Most sample points would have $Y_j(W^{*T}Z_j + W_0^*) \geq 1$. However, as seen in Figure 4.7, there would be points such as P, Q that violate the constraints. In particular, the distance of P away from its "margin" into the opposite side is ξ_P, and the distance of Q away from its "margin" into the opposite side is ξ_Q.

Linear Non-Separability in SVM

- The dual program to solve in the context of allowing some points to have $\xi_j > 0$, similar to (4.8), is

$$\min_{W,W_0,\xi|\lambda,\mu} \frac{1}{2}\|W\|^2 + \alpha\sum_{j=1}^{N}\xi_j + \sum_{j=1}^{N}\lambda_j\left(1-\xi_j - Y_j\left(W^T Z_j + W_0\right)\right) - \sum_{j=1}^{N}\mu_j\xi_j \qquad (4.11)$$

given λ and μ, where $\alpha > 0$ is a hyperparameter denoting the penalty to put on $\sum_{j=1}^{N}\xi_j > 0$. $\lambda_j \geq 0$ (defined to be positive) is a Lagrange multiplier on the inequality constraint $Y_j\left(W^T Z_j + W_0\right) \geq 1 - \xi_j$ for all j. $\mu_j \geq 0$ (defined to be positive) is a Lagrange multiplier on the inequality constraint $\xi_j \geq 0$ for all j.

- A large penalty α tends to reduce $\sum_{j=1}^{N}\xi_j$ as a choice, i.e., small error rates of training misclassifications, while a small penalty α and large $\sum_{j=1}^{N}\xi_j$ results in larger margin and large error rates of training misclassifications. α is to be fine-tuned for optimal performance in a validation data set before final testing is done.
- Fixing the optimal values for W*, we have the second stage optimization in $\lambda \geq 0$:

$$\max_{\lambda\geq 0|W^*} \sum_{j=1}^{N}\lambda_j - \frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N}\lambda_j\lambda_k Y_j Y_k Z_j^T Z_k \qquad (4.13)$$

subject to $\sum_{j=1}^{N}\lambda_j Y_j = 0$, $\alpha = \lambda_j + \mu_j$, $\lambda_j \geq 0$ and $\mu_j \geq 0$ for all j.

The constraints here can be simplified to $\sum_{j=1}^{N}\lambda_j Y_j = 0$ and $\alpha \geq \lambda_j \geq 0$ for all j.

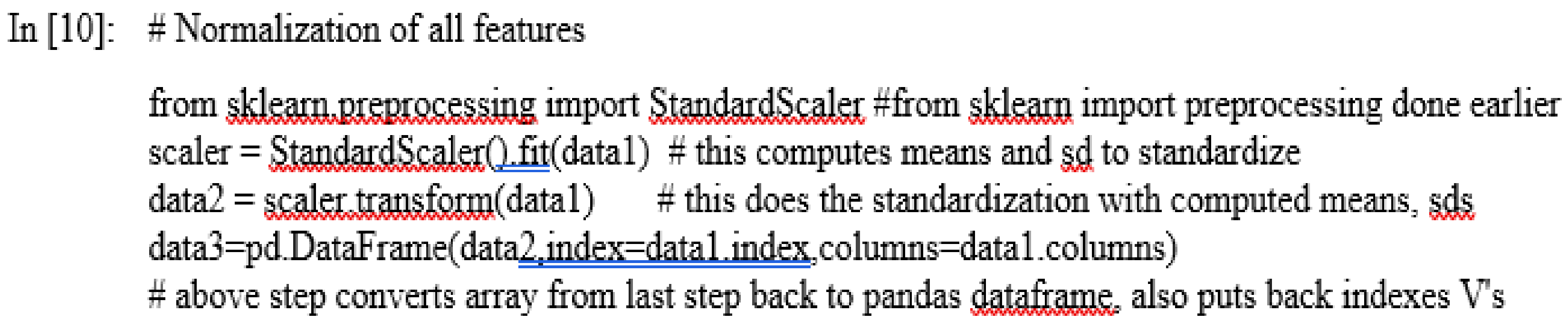
Worked Example

- The data set is found in public website: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud. It was collected by Worldline in collaboration with the Machine Learning Group of Université Libre de Bruxelles. The dataset shows European credit cardholders’ 284,807 transactions over two days in September 2013, out of which there were 492 fraudulent cases, accounting for about 0.17% of the transactions. Credit card fraudulent transactions occur when unauthorized person(s) somehow gain access to use the card to make purchases.
- According to the source, the data are numerical continuous variable features that are a PCA transformation from the original confidential client background information. There are 28 such features. In addition to these, two actual variables were added: “Time” variable indicates time in seconds that elapsed between each transaction and the first transaction in the dataset and “Amount” indicates dollar transaction amount. Eventually, “Amount” is also added as a feature. The target or class/type variable “Type” is a dummy variable that takes the value 1 if the case is fraudulent, and 0 otherwise.
- The “Time” variable is not included in the study as it does not appear to be a useful feature. The 28 PCA-transformed variables and the “Amount” are standardized via the StandardScaler and these 29
- Due to the huge type imbalance, it may be misleading just to consider Confusion Matrix accuracy, the AUC should also be considered.
- Other studies indicate client’s checking account history, pattern of credit card payments, abnormal use and amounts of credit card use, and client’s possession of property, car, and insurance could have predictive influence on the probability of a fraud (client is target of fraudulent card transactions due to lost or stolen card or hacked card details). These features are used to predict Type 1 (Fraud occurrence) or Type 0 (no Fraud occurrence).
- The prediction results are run in Chapter4-1.ipynb using Naïve Bayes, k-Nearest Neighbor, Support Vector Machines, and also the Logistic Regression algorithms. The main code lines are shown as follows.

Worked Example

Please upload Chapter4-1.ipynb and follow the computing steps in Jupyter Notebook

- There are 284,807 records/cases and 31 fields. Columns 2 to 30 will be used as features, while column 31 provides the target variable values or types. ‘Type’ = 1 indicates fraud case while ‘Type’ = 0 indicates no-fraud case. In [4], the percentage of fraud case is found to be very small at about 0.173%.
- Here we randomly select 492 sample points from the high density cases with Type 0 to match with the existing 492 sample points from the low density cases with Type 1.



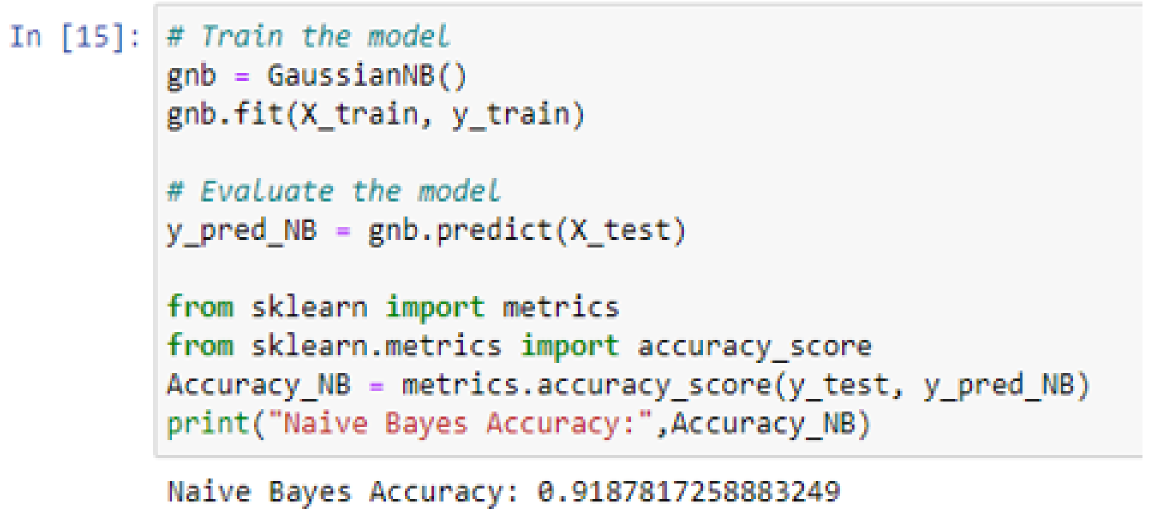
```
In [10]:  # Normalization of all features

          from sklearn.preprocessing import StandardScaler #from sklearn import preprocessing done earlier
          scaler = StandardScaler().fit(data1)  # this computes means and sd to standardize
          data2 = scaler.transform(data1)      # this does the standardization with computed means, sds
          data3=pd.DataFrame(data2,index=data1.index,columns=data1.columns)
          # above step converts array from last step back to pandas dataframe, also puts back indexes V's
```

Next, the data is split into 80% (787 cases) for training and remaining 20% (197 cases) for testing.

We skip validation and assume the test result is done using optimized hyperparameter(s), and that the performance would be similar to using a subset for validation.

Naïve Bayes Algorithm

Please upload Chapter4-1.ipynb and follow the computing steps in Jupyter Notebook

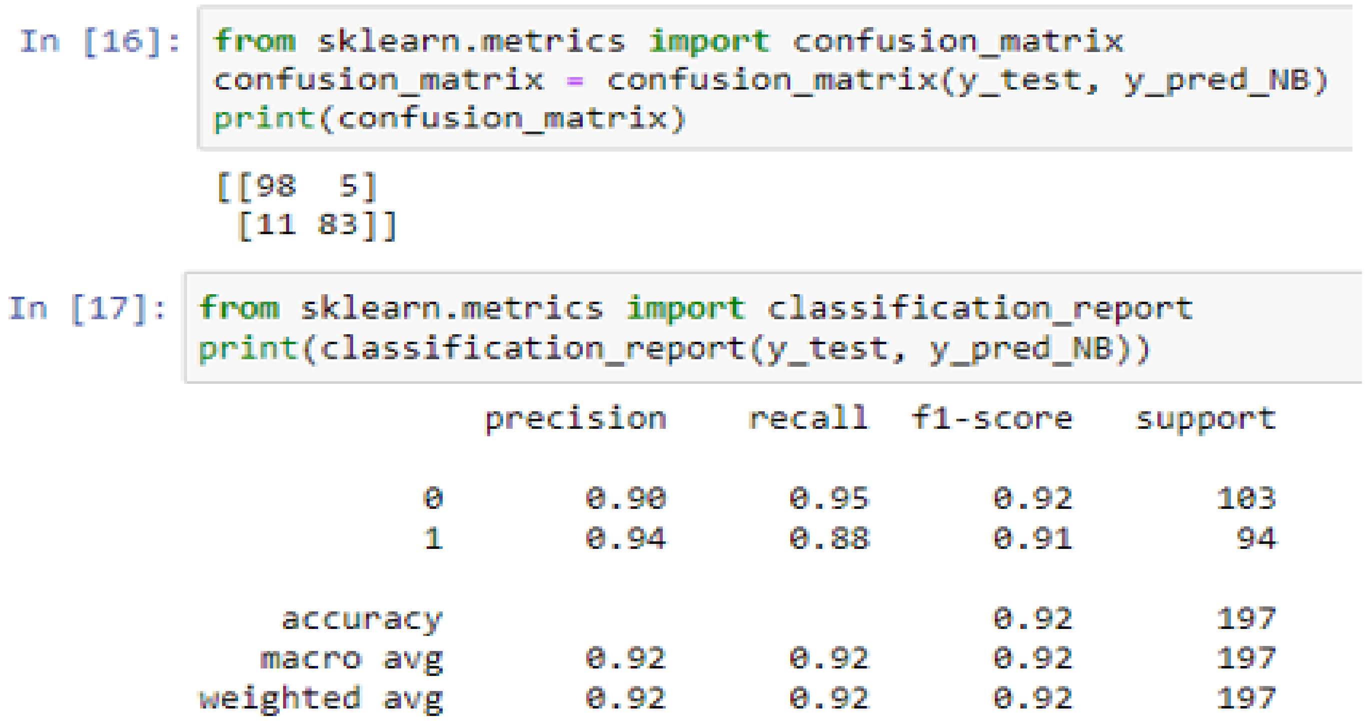


```
In [15]: # Train the model
         gnb = GaussianNB()
         gnb.fit(X_train, y_train)

         # Evaluate the model
         y_pred_NB = gnb.predict(X_test)

         from sklearn import metrics
         from sklearn.metrics import accuracy_score
         Accuracy_NB = metrics.accuracy_score(y_test, y_pred_NB)
         print("Naive Bayes Accuracy:",Accuracy_NB)

         Naive Bayes Accuracy: 0.9187817258883249
```

The confusion matrix, the classification report, and the ROC curve are obtained via code lines [16], [17], [18], [19]. ROC area is 96.075%.

Naïve Bayes Algorithm

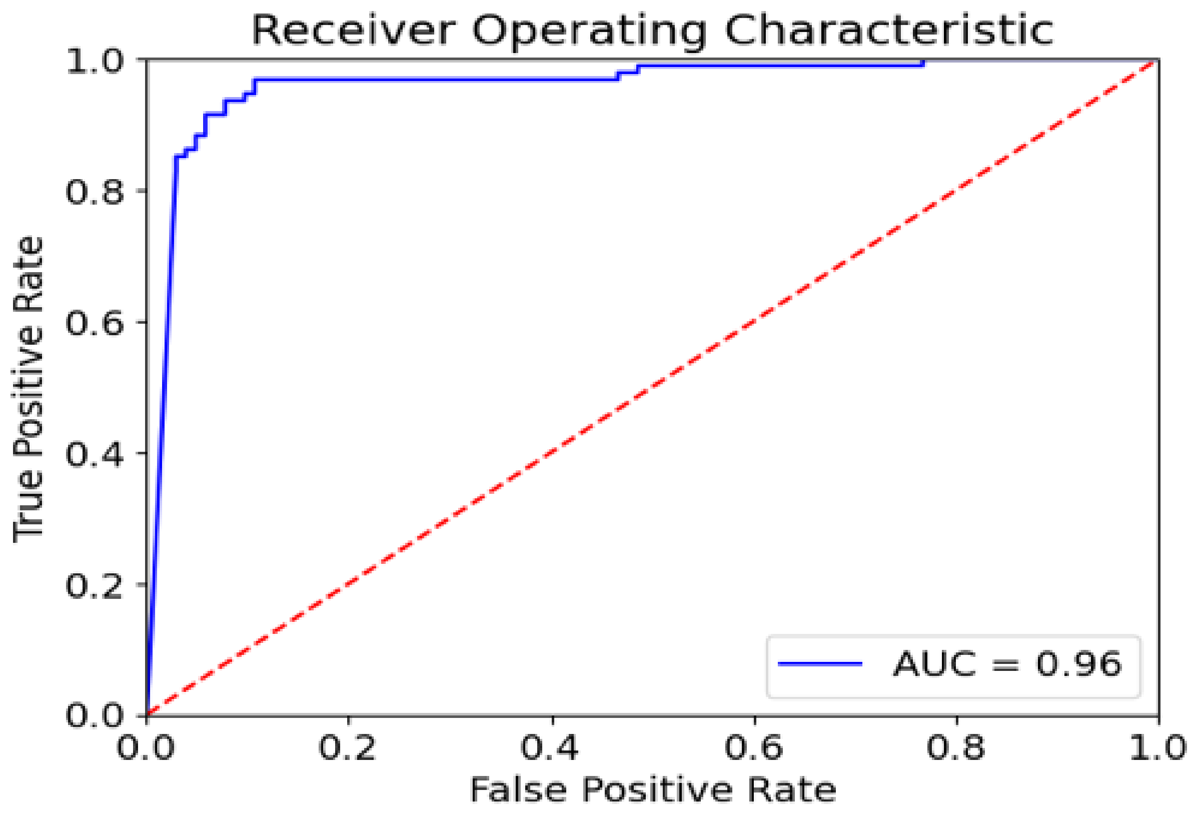
Please upload Chapter4-1.ipynb and follow the computing steps in Jupyter Notebook



```
In [16]: from sklearn.metrics import confusion_matrix
         confusion_matrix = confusion_matrix(y_test, y_pred_NB)
         print(confusion_matrix)

         [[98  5]
          [11 83]]

In [17]: from sklearn.metrics import classification_report
         print(classification_report(y_test, y_pred_NB))

                       precision    recall  f1-score   support

                    0       0.90      0.95      0.92       103
                    1       0.94      0.88      0.91        94

             accuracy                           0.92       197
            macro avg       0.92      0.92      0.92       197
         weighted avg       0.92      0.92      0.92       197
```

Naïve Bayes Algorithm

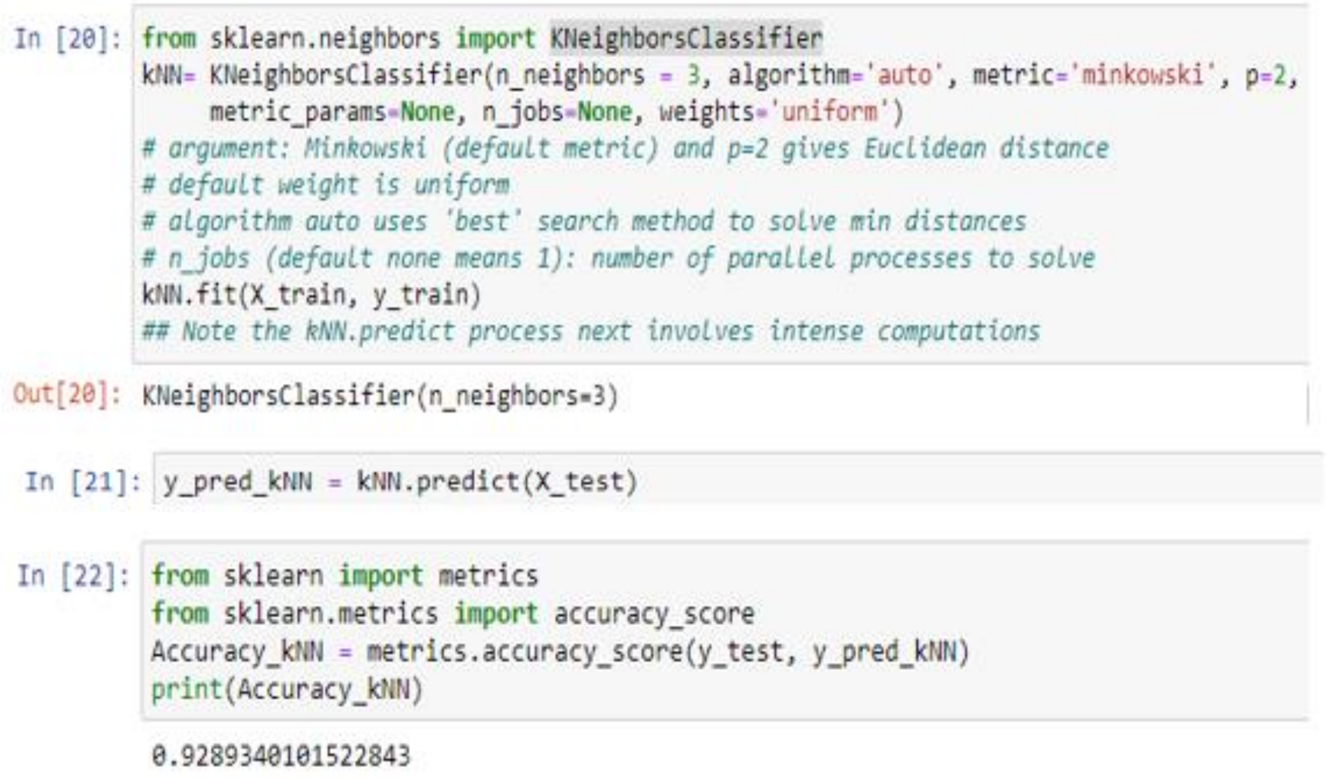
Please upload Chapter4-1.ipynb and follow the computing steps in Jupyter Notebook



K-Nearest Neighbor Algorithm

Please upload Chapter4-1.ipynb and follow the computing steps in Jupyter Notebook

Code lines [20], [21], [22] show the application of the sklearn KNeighborsClassifier employing k = 3 to fit the training set data, then perform the prediction on the test data (X_test). The accuracy is then reported as 92.893% using the sklearn metrics on accuracy_score.

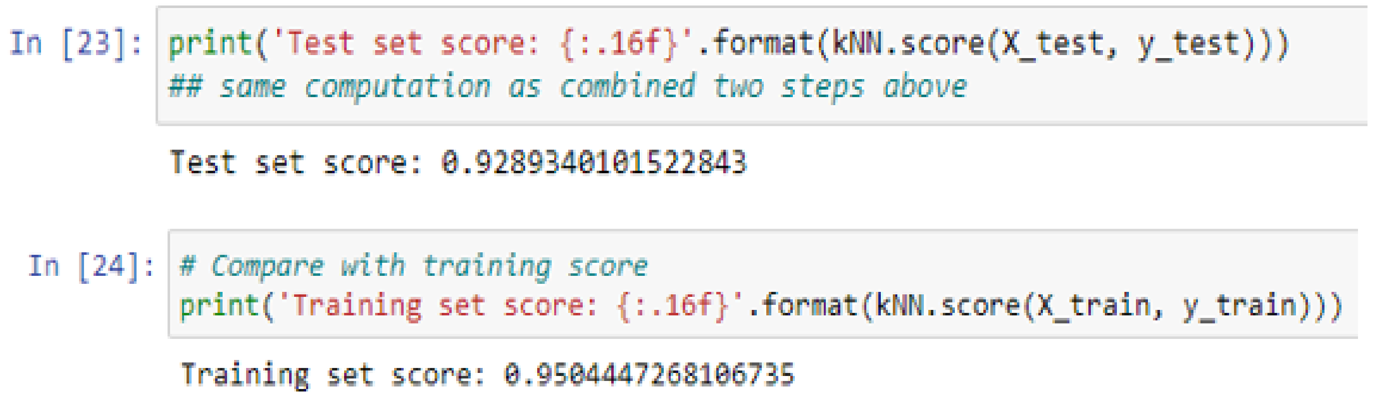


```
In [20]: from sklearn.neighbors import KNeighborsClassifier
         kNN= KNeighborsClassifier(n_neighbors = 3, algorithm='auto', metric='minkowski', p=2,
             metric_params=None, n_jobs=None, weights='uniform')
         # argument: Minkowski (default metric) and p=2 gives Euclidean distance
         # default weight is uniform
         # algorithm auto uses 'best' search method to solve min distances
         # n_jobs (default none means 1): number of parallel processes to solve
         kNN.fit(X_train, y_train)
         ## Note the kNN.predict process next involves intense computations

Out[20]: KNeighborsClassifier(n_neighbors=3)

In [21]: y_pred_kNN = kNN.predict(X_test)

In [22]: from sklearn import metrics
         from sklearn.metrics import accuracy_score
         Accuracy_kNN = metrics.accuracy_score(y_test, y_pred_kNN)
         print(Accuracy_kNN)

         0.9289340101522843
```

K-Nearest Neighbor Algorithm

Please upload Chapter4-1.ipynb and follow the computing steps in Jupyter Notebook

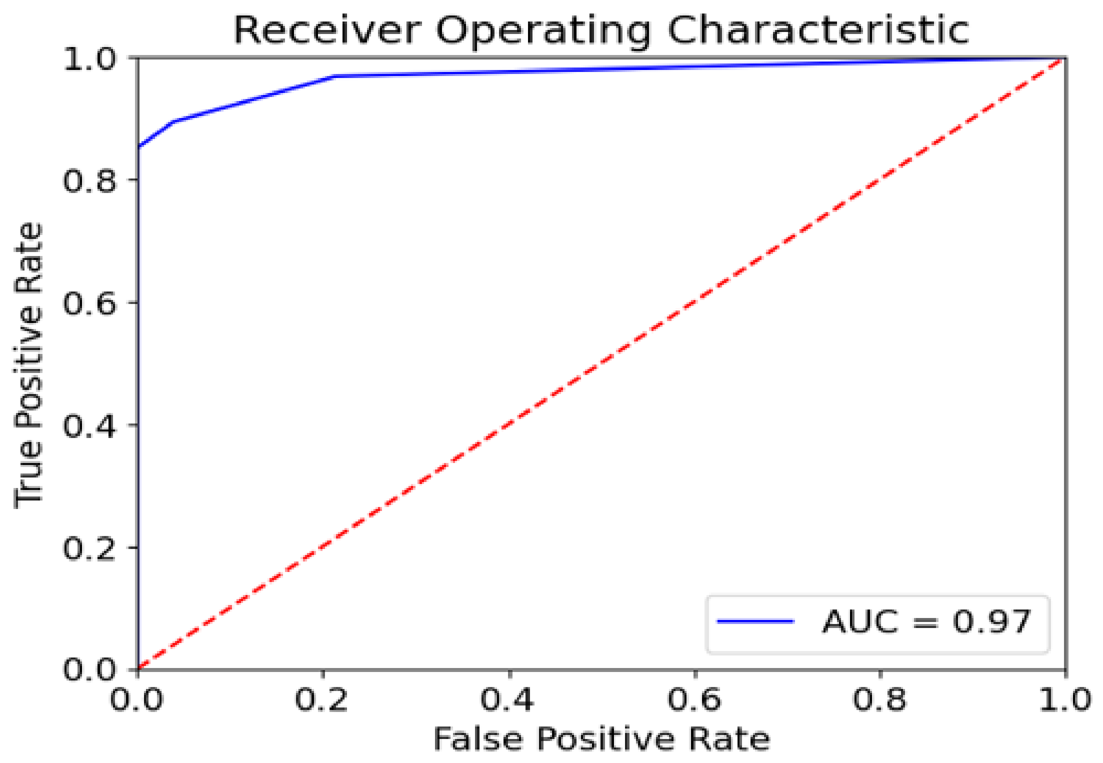
Code lines [23], [24] show that the training score is higher than the test score which is to be expected since the testing is done on generalized (out-of-sample) data. Both training fitting accuracy score and testing accuracy score are close so there is no overfitting in the training.



```
In [23]: print('Test set score: {:.16f}'.format(kNN.score(X_test, y_test)))
         ## same computation as combined two steps above

         Test set score: 0.9289340101522843

In [24]: # Compare with training score
         print('Training set score: {:.16f}'.format(kNN.score(X_train, y_train)))

         Training set score: 0.9504447268106735
```

K-Nearest Neighbor Algorithm

Please upload Chapter4-1.ipynb and follow the computing steps in Jupyter Notebook

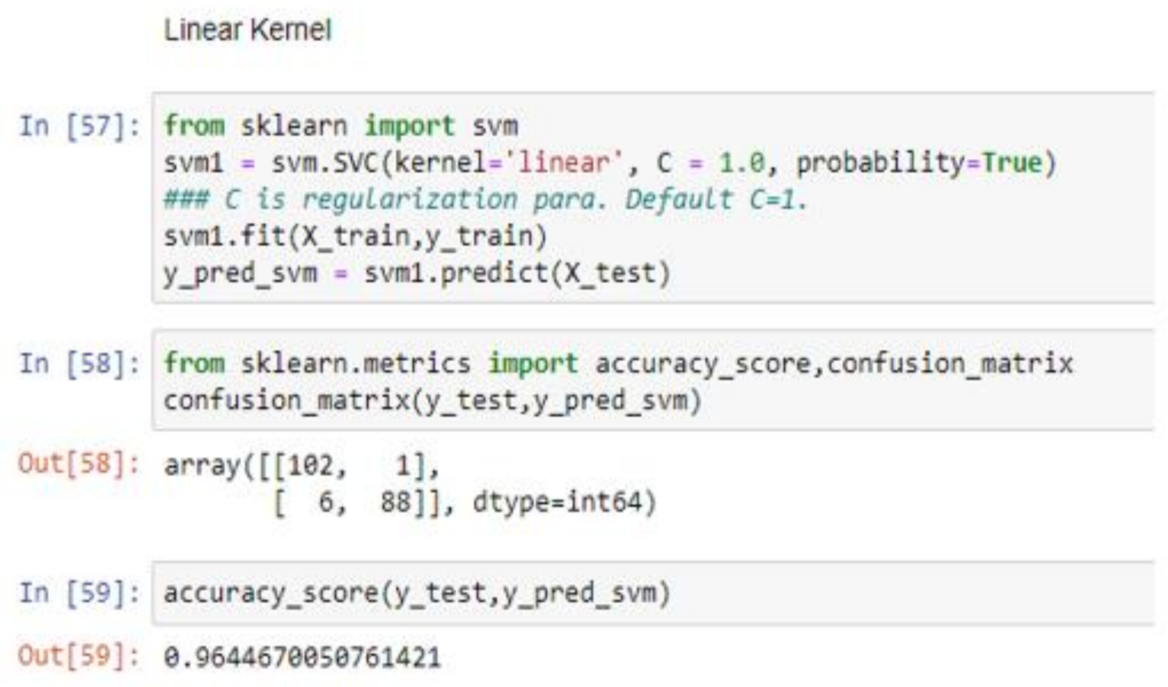


The area under ROC for k=3 is 97.041%, about 1% higher than that of Gaussian Naïve Bayes method. When k is increased to k=5, the area increases marginally to 97.707%, after which this area decreases when k is further increased. The accuracy also increases to 94.923% when k=5, after which it decreases as k increases. We assume that k=5 would have been the optimal hyperparameter to be used if we had used a validation subset.

Support Vector Machine

Please upload Chapter4-1.ipynb and follow the computing steps in Jupyter Notebook

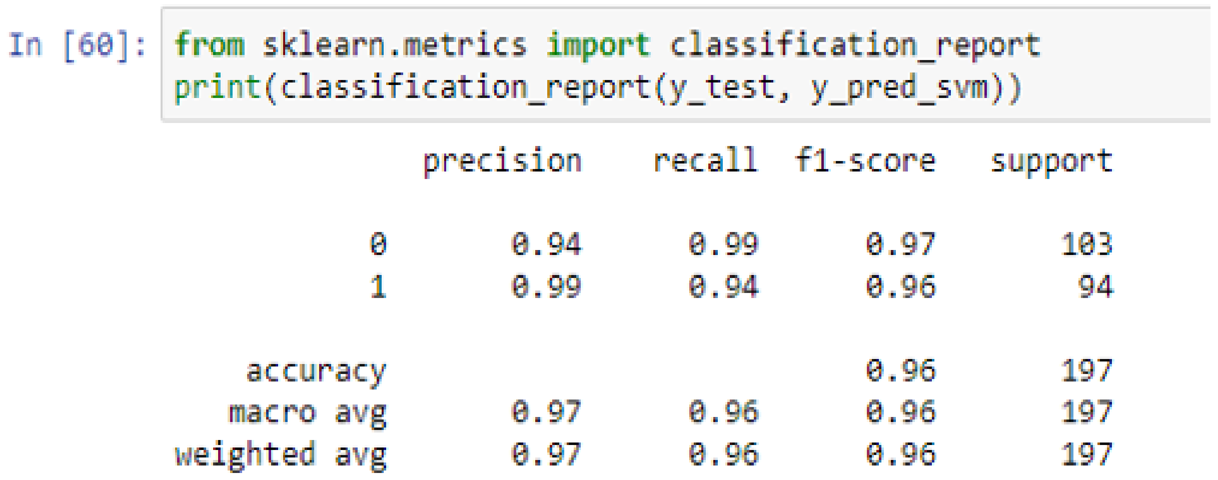
Code line [57] shows the application of the sklearn SVM employing the linear kernel with hyperparameter α = 1.0 (“C” in the ipynb file) to fit the training set data. These hyperparameters are assumed to be optimal from a validation. Then the model is for prediction on the X_test. Code lines [58], [59], [60] shows the confusion matrix, accuracy, and classification report. The accuracy is reported as 96.447%.


Linear Kernel

```
In [57]: from sklearn import svm
         svm1 = svm.SVC(kernel='linear', C = 1.0, probability=True)
         ### C is regularization para. Default C=1.
         svm1.fit(X_train,y_train)
         y_pred_svm = svm1.predict(X_test)

In [58]: from sklearn.metrics import accuracy_score,confusion_matrix
         confusion_matrix(y_test,y_pred_svm)

Out[58]: array([[102,   1],
                [  6,  88]], dtype=int64)

In [59]: accuracy_score(y_test,y_pred_svm)

Out[59]: 0.9644670050761421
```

Support Vector Machine

Please upload Chapter4-1.ipynb and follow the computing steps in Jupyter Notebook

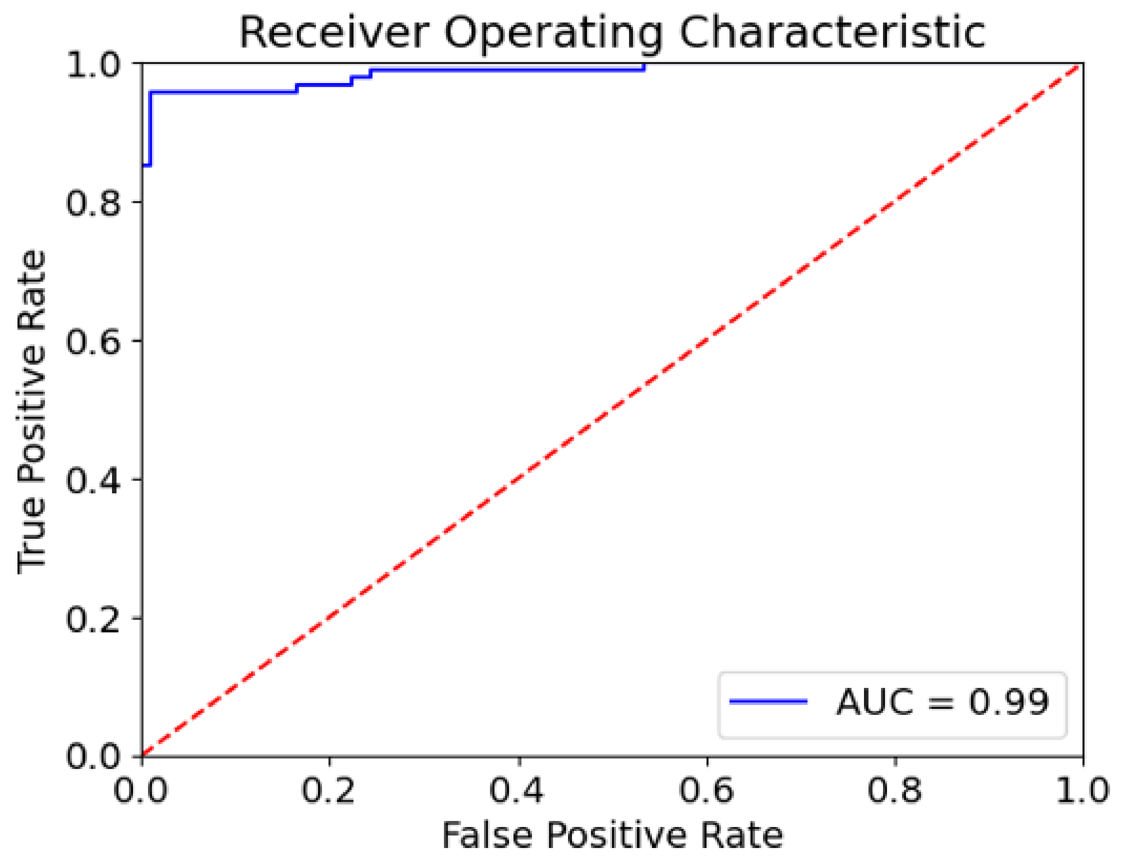


```
In [60]: from sklearn.metrics import classification_report
         print(classification_report(y_test, y_pred_svm))

                      precision    recall  f1-score   support

                   0       0.94      0.99      0.97       103
                   1       0.99      0.94      0.96        94

            accuracy                           0.96       197
           macro avg       0.97      0.96      0.96       197
        weighted avg       0.97      0.96      0.96       197
```

Code lines [61], [62] yield the ROC area and graph. The area is 98.657% which is the higher than those of the Naïve Bayes and the kNN methods.

Support Vector Machine

Please upload Chapter4-1.ipynb and follow the computing steps in Jupyter Notebook



Employing other kernels such polynomial, sigmoid, and rbf in the SVM produced lower accuracies and lower area under ROC curves.

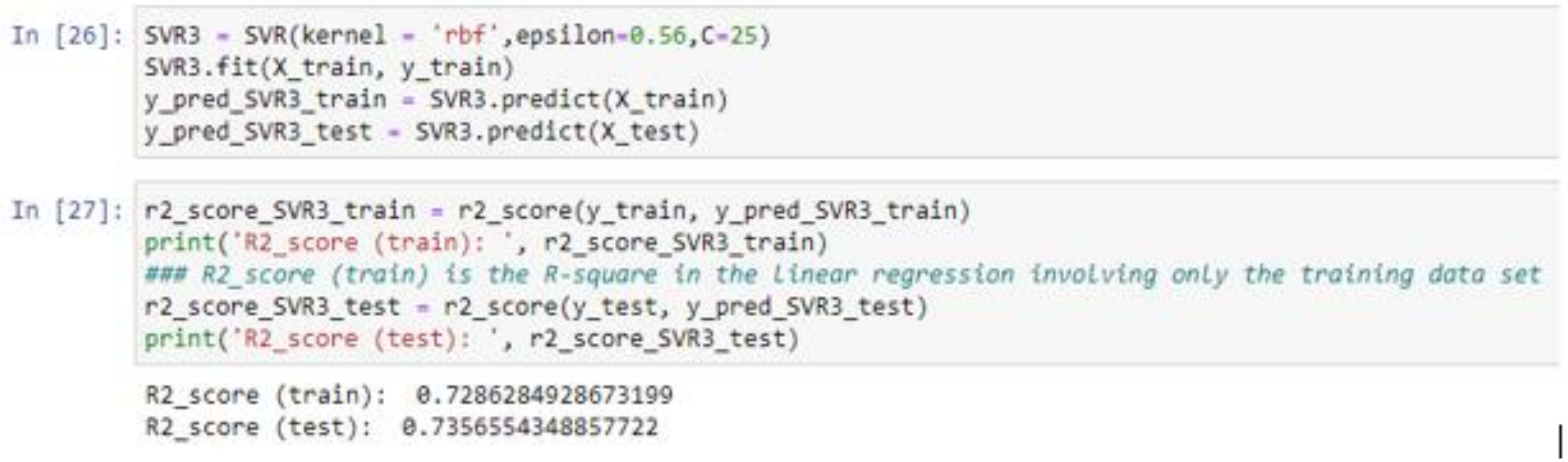
Support Vector Regression

Please upload Chapter4-2.ipynb and follow the computing steps in Jupyter Notebook

- Data set 'california_housing.frame.csv' . Predicting MedHouseVal from 8 features.
- In code line [17], the features are preprocesses and scaled to within (0,1).
- In code line [18], the data set is randomly split into 80% or 15,692 training data points and 3,923 test data points.

  In code lines [20], [21], linear kernel SVR (epsilon = 0.56, α or 'C' = 25) shows a R^2 training score of 61.97% and R^2 test score of 62.14%.

- Code lines [24] through [27] show the results of other nonlinear kernels. The best result is shown in code lines [26], [27] with the 'rbf' kernel.



```
In [26]: SVR3 = SVR(kernel = 'rbf',epsilon=0.56,C=25)
         SVR3.fit(X_train, y_train)
         y_pred_SVR3_train = SVR3.predict(X_train)
         y_pred_SVR3_test = SVR3.predict(X_test)

In [27]: r2_score_SVR3_train = r2_score(y_train, y_pred_SVR3_train)
         print('R2_score (train): ', r2_score_SVR3_train)
         ### R2_score (train) is the R-square in the Linear regression involving only the training data set
         r2_score_SVR3_test = r2_score(y_test, y_pred_SVR3_test)
         print('R2_score (test): ', r2_score_SVR3_test)

         R2_score (train):  0.7286284928673199
         R2_score (test):  0.7356554348857722
```

Summary

- Naïve Bayes
  - Lower computational time. Scales up with multi-class predictions. Works well with many features/dimensions.
  - Features independence assumption can cause poor performance. Not natural extension to numerical prediction.
- kNN
  - Non-parametric easy to understand. Lazy training. Can extend to both classification and numerical prediction. Extendible to multi-class problem.
  - Requires feature standardizations. Algorithm can be slow when dimension increases.
- SVM
  - Contrary to kNN, suited to handling many features or high dimension. Dimension can be larger than sample size. Nonlinear kernels allow good performance in complex data patterns.
  - May require high computational time for large data set. Overfitting can occur easily when number of features is large relative to sample size. Performance depends importantly on regularization (hyperparameter tuning).

In-Class Practice Exercise (not graded):

Chapter4-3.ipynb

Repeat the exercise done in Chapter4-1 except now with the total data set of imbalanced data

End of Class