STATISTICAL AND MACHINE LEARNING APPROACHES

INDIVIDUAL PROJECT

Hina Hussain

31-03-2022

This report aims to introduce five machine learning algorithms in detail and then sets up a benchmark experiments to compare them. The five selected models include Logistic Regression, Linear Discriminant Analysis, Decision Tree, Random Forest, and Support Vector Machines. The first part of the report will explain how these algorithms work, what is the mathematical objective function that runs in the background, and the advantages and disadvantage of each method. The second part of the report will run these models in R on a credit card default dataset. The models will be used for a classification problem of whether a customer will default or not. The accuracy of the predictions made on a test set by these models will be compared in this benchmarking experiment.

1 Logistic Regression

1.1 How does the method work?

Logistic regression is a method of predictive analysis used normally when the dependent variable (Y) is categorical variable. It is used to explain the relationship between one dependent variable and one or more independent variables. Logistic regression models the probability that Y belongs to a particular category. In the case of this project the logistic regression models the probability of whether the person will default in payment or not.

Logistic regression is used for categorical problems as linear regression falls short in such cases. Assuming we have an independent variable X, the linear model will present the probability as shown in equation 1. If this probability function was applied on a problem with a binary dependent variable (default), it will pe possible to predict a negative probability of default for values of X close to zero. For very large values of X, the probability of value obtained could be bigger than 1. This problem arises because any time a straight line is fit into a binary response (1 or 0), there will be some values of X where it will be possible to predict p(X) < 0 or p(X) > 1. The range of X will have to be limited otherwise.

Equation 1

$$p(X) = \Pr(Y = 1|X)$$

$$p(X) = \beta_0 + \beta_1 X.$$

Equation 2

$$f(x) = \frac{e^x}{1 + e^x}$$

Equation 3

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

Equation 4

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0+\beta_1 X}$$

Equation 5

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X$$

Logistic regression solves this problem by modeling a S shape curve through the **sigmoid function**. This leads to the probability values always being between 0 and 1. The sigmoid function and the logistic function are defined as shown in equation 2 and 3.

Here the e is the Euler number, p(X) is the probability that observation X is in class 1. The beta are the coefficient of the logistic regression model. However, this probability will need to be manipulated into an odds ratio. **Odds** are defined as the ratio of the probability of success and the probability of failure. In the case of logistic regression, odds represent the constant effect of a predictor X on the **likelihood** that a particular outcome will occur. If probability is used to represent the effect of X on the likelihood of a categorical Y (which has a specific value) the effect will not be constant.

Therefore, to get constant effect, odds are used instead of probability. Odds ratio can be written as equation 4.

The next step is to take the logarithm of both sides. The range of odds goes from zero to positive infinity while the range of log odds is negative infinity to positive infinity. To remove the restriction in the range, the log off odds is taken. The log-odds/logit can be written as shown in equation 5.

After taking the exponent to solve for p(X), the sigmoid function discussed before is obtained. It means that increasing X by one unit changes the log odds by beta1.The next step now is to estimate the beta coefficients. For linear regression, least squares fit is used but in the case of logistic regression **maximum likelihood** is preferred due to its statistical properties. If a least square fit is used for logistic regression, the cost function will not be convex as it is non-linear and it will be difficult to find the global minimum. The main idea of maximum likelihood is that the beta coefficients are estimated such that the predicted probability is as close as possible to the actual observed status. It is used to fit many non-linear models. This idea is written in the form of a likelihood function as shown in equation 6.

Equation 6

$$L(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p(x_i))$$

Equation 7

$$\ell(\beta) = \log \mathrm{L}(\beta) = \sum_{i=1}^{N} \{y_i \log p(x_i;\beta) + (1 - y_i) \log(1 - p(x_i;\beta))\}$$

$$= \sum_{i=1}^{N} \{y_i \beta^T x_i - \log(1 + e^{\beta^T x_i})\}$$

- $y_i = \{0; 1\}$
- $\beta = \{\beta_0, \beta_1\}$
- x_i : vector input, including constant term 1 for the intercept β_0

Beta coefficients are chosen to maximize this likelihood function i.e. maximize the likelihood of the observed data. Likelihood can be unstable and difficult to differentiate so to simplify our calculations both sides are multiplied by log to obtain the **log-likelihood function**. Log-likelihood and the original likelihood are both monotonically increasing functions, so it is okay to do so. The likelihood function is displayed in equation 7.

Since conventional optimization minimizes the error through gradient descent, we will take the negative of the function above to maximize the likelihood. Maximum of a log function is the same as minimum of a negative log function. By taking the negative of the log-likelihood, the cost function is obtained. This cost function is minimized using **gradient descent** optimization. Plotting the cost function, we obtain a convex curve with only one local minimum. Gradient descent works by finding the optimal weights which minimize the cost function. It starts at a random point on the curve and then iteratively moves by finding the slope of the curve at the point and moving to reach the bottom. The step size at each iteration is determined by the learning rate. The lower the rate the better so that the minimum point is not missed. This way the descent converges to the minimum point and the optimal beta coefficients which minimize the cost function and maximize the likelihood are obtained.

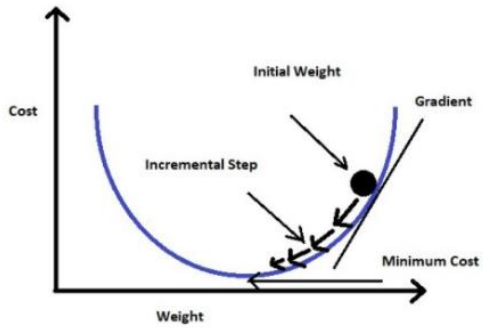


Image Source: www.analyticsvidhya.com/

1.2 How to read the output

When a logistic regression model is fit on a dataset in R, the output shows the coefficient estimates, standard error, z-statistic, the p-value, and some other metrics. For example, for this project if a value for beta(x) is 0.05, it will be read as a one unit increase in x is associated with an increase the log odds of default by 0.05 units. It should be noted that if input variables are in different units, then the coefficients cannot be compared with each other. The standard error shows the accuracy of the coefficient estimates. The z-value is calculated as the beta coefficient estimate divided by its standard error. Therefore, a large absolute value of the z statistic indicates that the null hypothesis can be rejected. The null hypothesis states that the there is no association between the probability of Y and X. Rejecting it means that an association exists. The p-value of each term also tests the null hypothesis. A low p-value (e.g., less than 0.05 depending on the significance level) means that the null hypothesis can be rejected.

1.3 Advantages and disadvantages

- ✓ Easy to implement and interpret the coefficients
- ✓ It can be applied to multinomial regression problems as well
- ✓ Results in good accuracy for simple datasets especially when the data is linearly separable
- ✓ Model coefficients can indicate feature importance
- ✓ It outputs well calibrated probabilities and has a close relationship with neural networks
- × Can lead to overfitting when the number of observations is less than the number of predictors
- × The assumption of a linear relationship between the independent and dependent variables is a limitation
- × Cannot solve non-linear problems because of its linear decision surface
- × Hard to capture complex relationship in data and underperforms compared to complex models
- × It requires there to be little or no multicollinearity between independent variables

2 Linear Discriminant Analysis (LDA)

2.1 How does the method work?

LDA is an alternative approach to logistic regression for classification problems. It is also used as a dimension reduction method, but the focus of this paper will be on LDA's use a classification algorithm. It works by modeling the distribution of the predictors separately in each of the categories of Y. After this it uses Bayes' theorem to flip these around into estimates for the probability of Y given X. The main reasons why LDA is used are:

- When the classes are well-separated, logistic regression model is unstable
- LDA is more stable if n is small and the distribution of the predictors X is approximately normal in each of the classes
- LDA is popular when there are more than two response classes

Bayes' theorem states that the conditional probability of event A based on the occurrence of event B is equal to the likelihood of event B happening given the event A multiplied by the probability of event A. It can be stated as shown in equation 8. The following explanation will be based on the supposition that we need to classify an observation into one K classes where K is greater than two. Applying the Bayes theorem on such a problem will result in the transformations shown in equation in 9 and 10.

Equation 8

$$Pr(A|B) = \frac{Pr(B|A) * Pr(A)}{Pr(B)}$$

Equation 9

$$Pr(Y = k|X = x) = \frac{Pr(X = x|Y = k) * Pr(Y = k)}{Pr(X = x)}$$

Equation 10

$$Pr(Y = k|X = x) = \frac{\pi_k * f_k(x)}{\sum_{l=1}^{K} \pi_k * f_l(x)}$$

Equation 11

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{1}{2}(\frac{x-\mu_k}{\sigma_k})^2}$$

Equation 12

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}(\frac{x-\mu_k}{\sigma})^2}}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}(\frac{x-\mu_l}{\sigma})^2}}$$

Equation 13

$$\delta_k(x) = x.\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

In equation 10, πk is the prior probability that a randomly chosen observation belongs to the kth class i.e. kth category of the response variable Y. fk(x) ≡ Pr(X = x|Y = k) shows the density of X for an observation from the kth class. This means that fk(x) which is the density of X in k will be large if there is a high probability that an observation in the kth class has X ≈ x, and vice versa.

Equation 10 says that we can calculate the probability that Y belongs to class k given X if we plug in the values of πk and fk(x). πk can be estimated by taking a random sample of Y from the population and calculating the fraction on population that belong to the kth class. However, estimating fk(X) is not straightforward and some **assumptions** need to be made for simplicity. We assume that fk(X) follows a **normal/Gaussian distribution**. The normal density function is stated in equation 11 where μk and σ^2k are the mean and variance of x in class k. It is also assumed the variance term is the same across all k classes is the same and is denoted by σ^2. After plugging in this equation of fk(X) into the Bayes' theorem equation, equation 12 is obtained. For simplicity, equation 12 is simplified by taking the log and doing rearrangement. The resulting equation is the **discriminant function** shown in equation 13. It tells us how likely x is from each class. An observation will be assigned to the class for which this classifier is the largest.

The discriminant function is a linear function in x. This means that the **decision boundary** between any pair of classes is also a linear function in x. An example can be used to explain the decision boundary. if K = 2 and π1 = π2, then the Bayes classifier assigns an observation to class 1 if 2x ($\mu_1 - \mu_2$) > $\mu^2_1 - \mu^2_2$, and to class 2 otherwise. This is the decision boundary.

The next step is to estimate the parameters in the equation above to obtain the **discriminant score**. The parameters are estimated so that the LDA approximates the Bayes Classifier. An approximation is used as in real life it is not possible calculate the Bayes Classifier. The following estimates are used for the parameters.

$$\hat{\pi}_k = \frac{n_k}{n} \qquad \hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i \qquad \hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2$$

n is the number of observations and n_k is the number observations in class k. μ_k estimate is the average of all the observations from class k. σ^2 estimate is the weighted average of the sample variances for each of the K classes.

The LDA model then plugs in these parameter estimates into the discriminant function. It assigns an observation to the class for which $\hat{\delta}_k(x)$ is the largest.

The discussion above was based on the case where there is only one predictor (X). The method for cases where the predictors are more than 1 is similar. It assumes that each individual predictor follows a one-dimensional normal distribution with a class-specific multivariate mean vector and a common covariance matrix. This concept is known as **multivariate Gaussian**. This density function for p dimensions is given by:

$$f(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

μ_k is a class-specific mean vector and Σ is a covariance matrix. If we plug this multivariate density function into our Bayes' theorem equation and do some as before transformation, the following discriminant function is obtained for when the predictors are more than 1:

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \log(\pi_k)$$

This function assigns an observation to the class for which $\delta_k(x)$ is the largest. The similar method as discussed before is to be used to estimate the parameters to obtain the discriminant score.

2.2 How to read the output

The section above walks through the steps that the LDA uses to classify an observation. This section discusses how to read and evaluate the output produced. In R, when an LDA model is fit on the train set, it uses the data to calculate the prior probabilities of groups, the group means, and the coefficients of linear discriminants. This model is then used to make predictions on a test set. The predictions can be evaluated conveniently using a **confusion matrix**. The matrix compares the predictions with the actual status. The following shows what the structure of a confusion matrix looks like. + can be considered as the defaulter and – as the non-defaulter. The matrix can be used to calculate the overall error rate by looking at the misclassified observations compared to the total number of observations.

		Predicted class		
		− or Null	+ or Non-null	Total
True class	− or Null	True Neg. (TN)	False Pos. (FP)	N
	+ or Non-null	False Neg. (FN)	True Pos. (TP)	P
	Total	N*	P*	

Source: James, G., Witten, D., Hastie, T., & Tibshirani, R. (2014). An Introduction to Statistical Learning: With Applications in R.

However, it is also important to consider the **class specific error**. According to the business problem at hand class specific error can be crucial. For example, since the bank wants to identify people who default, it is important to have a low error rate for people who defaulted as compared to people who did not. Concepts of **sensitivity** and **specificity** come are important here. Sensitivity is the percentage of true

defaulters that are identified as defaulters and specificity and specificity is the percentage of non-defaulters that are correctly identified as non-defaulters. The Bayes Classifer/LDA does not look at each separate class but tries to obtain the smallest total number of misclassified observations possible. It does so by assigning an observation to the class for which the **posterior probability** is the largest. In our binary example, it assigns an observation to default class if the posterior probability is greater than 50%. This posterior threshold can be manipulated to get a more conservative classification according to the business problem at hand. The bank could lower it to 25% so that the error of misclassifying defaulters reduces as more observations will now be classified as default.

Results using different thresholds for the posterior probability can be plotted using the **ROC** curve (discussed in benchmark experiment section). The true positive rate will be the sensitivity and the false positive rate will be 1-specificity. The AUC can then be calculated to see the performance of the classifier.

2.3 Advantages and Disadvantages

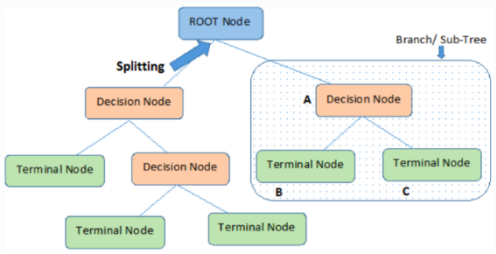
- ✓ LDA is a simple and fast algorithm which beats logistic regression when its assumptions are met
- ✓ It works with a linear decision boundary
- ✓ Good for non-binary classification
- × It requires assumption that the predictors are normally distributed
- × Does not work for non-linear problems
- × Does not work for some types of categorical variables

3 Decision Tree

3.1 How does the method work?

Decision Tree is a supervised learning method used for classification and regression. It builds models in the form of tree-like structures by breaking down data into smaller pieces. The resulting tree has decision nodes, leaf nodes, root node, branches etc. Following are the main elements of a decision tree model:

- Root Node is the entire population
- Splitting is the process of dividing a node into sub-nodes
- Internal/decision nodes are the points at which the predictor space is split
- Leaves are terminal nodes that do not split
- Branches are the segments of the tree that connect the nodes i.e., subsection of a tree



Source: Decision tree algorithm; www.kdnuggets.com

Decision trees can be used for regression and classification both. This report will explain the basic concepts of decision tree **regression** and then extend them to explain **classification**. The process of building a decision tree to make predictions has two main steps:

i. Divide the predictor space (X_1, X_2, ..., X_p) into J distinct and non-overlapping regions, R_1, R_2, ..., R_J.
ii. For every observation falling into the region R_j, its predicted value is the mean of the response values of training observations in region R_j.

The predictor space is divided into high dimensional rectangles for ease of interpretation. The aim is to find rectangles that minimize the residual sum of squares (**RSS**) which is the sum of differences between the response of a training observation in a particular region and the mean response of the training observations within that region. Since it is computationally infeasible to scan through every possible division of the predictor space, a **top-down**, **greedy** approach known as **recursive binary splitting** is used. It is called top-down as it starts at the top (root node) and then splits the predictor space step by step through two new branches. It is called binary as it splits the predictor space into two at each step. It is called greedy because at each step it only looks at the best split at that particular step and does not consider impact on possible future steps. Best split means that it leads to greatest possible reduction in the RSS. This process can be presented mathematically as below:

$$R_1(j,s) = \{X|X_j < s\} \; and \; R_2(j,s) = \{X|X_j \geq s\}$$

This is the pair of half-planes for any j and s. For example, R_1 means the region in which X_j takes on a value less than s. Recursive binary splitting considers all predictors and all possible values of the **cutpoint** s for each of the predictors. It then chooses the predictor and cutpoint that make the resulting tree have the lowest RSS. This means that it looks for the pair of j and s that minimize the following equation:

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

$\hat{y}R_1$ is the mean response for the training observations in R_1 (j,s) and $\hat{y}R_2$ for R_2 (j,s). This splitting process of looking for best predictor and cutpoint is repeated for subsequent regions until a stopping criterion is reached. Once the regions have been created, the response for a specific test observation can be predicted using the mean of the training observations in the region in which the test observation falls.

The process described above can lead to complex trees being made which can cause overfitting. This problem can be addresses through **pruning**. Pruning works by removing subtrees that are not useful and replacing by a leaf node. Pruning can be of two types – pre-pruning and post-pruning. Under pre-pruning, the construction of a subtree is stopped at a specific node based on a measure of error such as the RSS. Post-pruning works in the opposite way. An entire tree is built and then the nodes are inspected in a bottom-up manner and the decision is made whether to keep the node or replace it based on a measure of error.

This report will now discuss the **classification tree**. While the regression tree was used to predict a numerical response, the classification tree will be used to predict a qualitative response (e.g. default, not-default). Instead of predicting a response by using the mean response of the training observations, the classification tree predicts that each test observation belongs to the **most commonly occurring class** of training observations in the region which the test observation belongs. Classification tree also uses recursive binary splitting but since RSS cannot be used as the splitting criterion, the following alternative ways to calculate the error can be used.

- **Classification error rate** – This is the proportion of misclassifications. Classification tree assigns an observation to the most commonly occurring class of training observations of its region. Therefore, the classification error rate is calculated as the fraction of the training observations in that region that do not belong to the most common class. It can be written as the equation below where $\hat{p}_{mk}$ is the fraction of training observations in the mth region that belong to class k.

$$E = 1 - \max_k(\hat{p}_{mk})$$

- **Gini Index** – This is a measure of equality and uses the total variance across K classes. It takes on values between 0 and 1 where a large value shows higher inequality. It can be written as the equation below. If $\hat{p}_{mk}$ is small, then the Gini index value will be small meaning that a node mainly contains observations from a single class. This is also called the purity of a node.

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

- **Cross-entropy** – This is a measure of randomness or uncertainty and is similar to the Gini Index. Rearrangement shows that since $\hat{p}_{mk}$ can only fall between 0 and 1, negative $\hat{p}_{mk}$ multiplied by the log of itself will have to be greater than equal to 0. This can be shown by the entropy equation below. In short, for extreme values of $\hat{p}_{mk,}$ the entropy will be close to 0. This means that entropy will have a small value if the mth node is pure.

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} log \hat{p}_{mk}$$

3.2 Advantages and Disadvantages

- ✓ Very intuitive and easy to understand
- ✓ The result can be shown in the form of a graph and is easy to interpret
- ✓ Trees can be constructed even when the predictors are categorical variables
- × Can easily overfit in case of numerous X variables (methods like pruning can be used for this)
- × Do not have as good a predictive performance as other methods
- × Results can have high variance. A small change in data can lead to large change in result

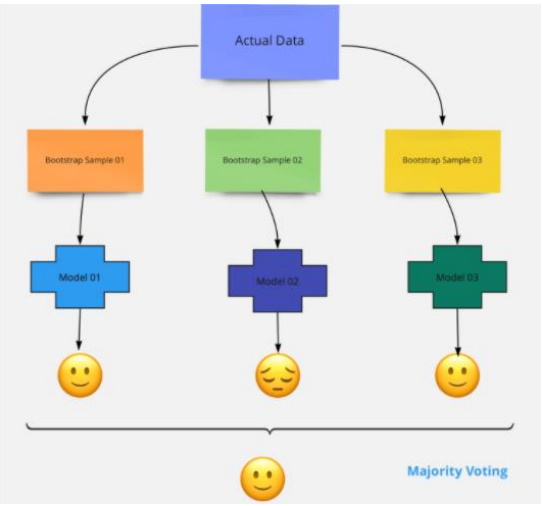
4 Random Forest

4.1 How does the method work?

Decision trees suffer from the problem of high variance and low predictive performance which can be improved by using aggregating methods such as bagging, random forests, boosting. This section focuses on Random Forests. Random forest is a supervised algorithm used for classification and regression and can handle both continuous and categorical variables. It builds a number of decision trees on bootstrapped training samples and takes their majority vote for classification and average in case of regression. Random forest gets its name from the fact that it builds multiple trees (**Forest**) and each tree in the forest is built by **randomly** selecting a sample of the data.

As an example, consider a person who has to decide between two jobs based on his/her skillset. He/she consults various people e.g. teachers, family, other colleagues and asks various questions such as why should he/she choose, the benefit, the drawbacks etc. After consulting, he/she decides to choose the job voted by most people. This is the main idea by which random forest operates.

Random forest is considered an ensemble machine learning model and is based on the concept of **Bootstrap aggregation** or **bagging**. Ensemble means combining multiple models to make predictions. Bagging creates different subsets from sample training data with replacement. The main goal is to reduce the variance of the sampling as averaging a set of observations reduces variance. Bagging works by choosing a random sample from the dataset with replacement called row sampling. This means that the data is not split into smaller chunks, rather if the data has a size on N, each subset will also have a size on N through row replacement. Bagging then trains models independently on each subset and the resulting trees have high variance but low bias. It combines the results of all the models and produces prediction through **majority voting** for classification and averaging for regression.



Understanding Random Forest; www.analyticsvidhya.com

Random forest takes bagging one step ahead and add the concept for **decorrelation**. It builds various decision trees on bootstrapped samples but at the time of splitting the tree, a random sample of m predictors is chosen as split candidates from the full set of p predictors. The split then is based only on those m predictors. A new sample of m predictors is considered for each split. Usually, the value of m is reached by taking the square root of the total number of predictors p. This is done to reduce the influence of a very strong predictor. If there is a strong predictor and all the available variables are used almost all the trees will use this strong predictor in the top split. This would make all the bagged trees similar to each other resulting in highly correlated predictions. Averaging these correlated results will not lead to the reduction in variance that we are looking for as compared to a single tree. This issue is solved by random forest when it forces each split to consider only m predictors. Some splits will have the strong predictor and some will not. This process results in decorrelating the trees and makes the average of the resulting trees less variable and more reliable.

Once the predictions are made, it is easy to estimate the test error in bagged Random Forests. Random forest repeatedly fits trees to bootstrapped subsets the data and each tree uses approximately two-thirds of the observations. The remaining one-third of data which are not used by the tree are called **out-of-bag (OOB)** observations. Prediction can be made for a specific observation using each of trees which did not use that observation i.e. OOB. This will result in numerous (one-third of the number of bootstrapped subsets used) predictions for that observation. The average of these predictions can be used for regression, or the majority vote can be taken for classification. This process can be applied to all the observations and the MSE or classification error can be obtained depending on the type of model.

Another interesting element of random forest is **variable importance**. Random forest is difficult to interpret as many trees are built but it gives the possibility to see the overall summary of the importance of each predictor. For regression, **RSS** is used. The decrease in RSS resulting from splitting over a particular predictor can be calculated and then averaged over all the trees. If the decrease is large, then the predictor is considered important. For classification problem, **Gini Index** is used. The amount that the Gini index is decreased by because of splits over a particular predictor can be averaged across all trees.

4.2 Advantages and Disadvantages

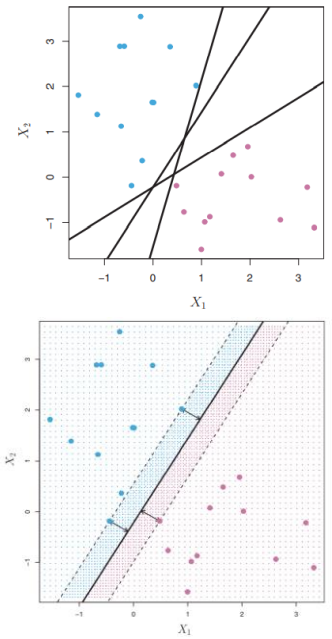
- ✓ Building multiple trees and combining the output reduces overfitting and variance
- ✓ It works well with continuous and categorical variables to do regression and classification
- ✓ It is not affected by the **curse of dimensionality** since each tree does not consider all variables. This helps to maintain **diversity**.
- ✓ Each tree is created in **parallel** independently so full CPU memory can be used
- ✓ It can automatically handle missing values and outliers
- ✓ Data does not need to be normalized or scaled to build random forests
- ✓ Large number of trees ensure stable predictions and a small change does not cause variance
- ✓ It is able to calculate the variable importance of the predictors
- × Random forests are complex as they build multiple trees
- × Less interpretable as compared to a single decision tree and difficult to visualize
- × Training the model is a time-consuming task as each decision tree has to generate output

5 Support Vector Machine

5.1 How does the method work?

Support vector machines (SVM) are a supervised machine learning model used mostly for classification problems but can be used for regression too. This section first discusses some elements used by SVM and the moves on to discussing the SVM Classifier.

SVM is based on using hyperplanes. A **hyperplane** is a tool that separates the dataspace into one less dimension for classification E.g., in two dimensions, a hyperplane is a flat one-dimensional subspace. If we want to construct a hyperplane that separates the hyperplane training observations into two classes (blue and purple), the possible **separating hyperplanes** will look like the first figure on the right. Since there is more than one possible separating hyperplane, the question arises which one to use. A possible answer is to use **maximal margin hyperplane** / **optimal separating hyperplane** which is the separating hyperplane that has the biggest distance from the training data. Perpendicular distance from each observation to the hyperplane is calculated, and the smallest of these is called the **margin**. The optimal separating hyperplane is the one which has the highest margin (largest minimum distance). After this step, a test observation can be classified depending on which side of the hyperplane it falls. This concept is known as the **maximal margin classifier**. In the second figure on the right, the dashed line shows the width of the margin. The observations that lie on the dashed line are called **support vectors**. The maximal margin hyperplane depends on these observations because if they move, the hyperplane will move as well.



Source: James, G., Witten, D., Hast T., & Tibshirani, R. (2014). , Introduction to Statistical Learning

Maximal margin classifier has the tendency to overfit on train set if the number of dimensions is large. A small change in the data can lead to a big shift in the separating hyperplane. It is also infeasible in cases where the separating hyperplane does not exist. However, the concept of a separating hyperplane can be extended to include a **soft margin** which helps to find a hyperplane that **almost** separates classes. This extended concept is called **support vector classifier (SVC)**. SVC allows the margin to be **violated** by some of the training observation which is why the margin is called soft. SVC does not look for the largest possible margin that ensures that each observation is on the correct side of the hyperplane and also on the correct side of the margin. Instead, it allows some observations to be on the incorrect side of the margin, or even the incorrect side of the hyperplane. It allows misclassification of a few training observations so it can do a better job at classifying the rest. This is done through **slack variables**. Slack variables show where an observation is located in relation to the hyperplane and margin. It shows whether an observation is on the correct side of the hyperplane and if it is on the correct side of the margin. This slack variable is controlled by a **tuning parameter**. It bounds the sum of the slack variable for each observation, hence determines the tolerance for the number and severity of the violations to the margin and hyperplane. Tolerance to violations increases as the tuning parameter increases and vice versa. If the tuning parameter is small, narrow margins are obtained which result in a classifier that is tightly fit to the data and has low bias but high variance. If the tuning parameter is bigger, wider margins are obtained which result in a classifier that is not tightly fit and may have more bias and lower variance. Observations that lie on the margin, or on the incorrect side of the margin are called support vectors.

The SVC is mainly used for binary classification problems if the boundary between the two classes is linear. It doesn't work with **non-linear** decision boundaries as it looks for a linear boundary and performs poorly. **Feature expansion** is used to solve this issue and obtain non-linear decision boundaries. The feature space is enlarged using quadratic, cubic, and even higher-order polynomial functions of the predictors. For example, the number of features can be doubled by also including the square of each feature or by including interaction terms. In the expanded feature space, the decision boundary obtained is linear but when it is in the original space it becomes non-linear.

Support vector machines (SVM) are built upon SVCs and enlarge the feature space through **kernels**. Kernel is a computational approach to efficiently enlarge the feature space to obtain non-linear decision boundaries. It works by taking a low dimensional input space and transforming it into a high-dimensional space. Infinite number of dimensions can be obtained using kernels. The type of the kernel parameter can be set in the SVM code in different programming languages. The types can range from linear to polynomial to radial. When the SVC is combined with a non-linear kernel the resulting classifier is called the SVM.

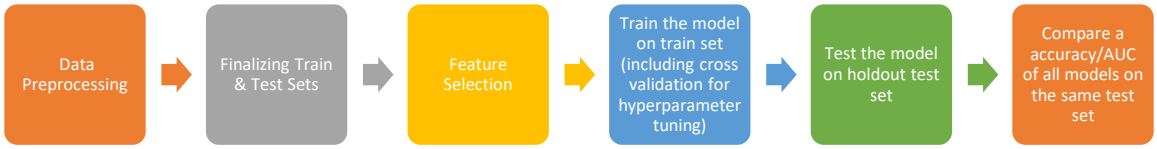
5.2 Advantages and Disadvantages

- ✓ SVMs are robust to changes in observations far away from the hyperplane
- ✓ Effective in high dimensional spaces
- ✓ Works well even if number of dimensions is greater than the observations
- ✓ It is memory efficient as it uses a subset of the training points
- ✓ It works very well with non-linearly separated data
- × Requires high training time if dataset is large
- × Does not perform well when data has noise or the classes in the target variable overlap
- × It doesn't produce probability estimates; other methods are needed to extract probabilities

6 Benchmark Experiment

6.1 Experimental Set-up

This section of the report focuses on conducting a benchmark experiment in R of the five models discussed in the section above. Benchmarking in machine learning means comparing different models on the same set of data and evaluating their performance.

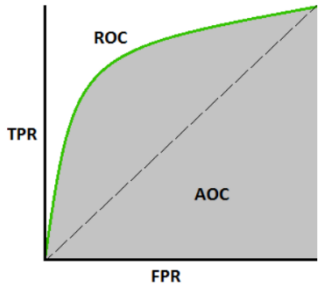


Data Preprocessing - As a first step, the dataset was split into train (70%) and test (30%) and each set was treated separately. The data set was cleaned to handle errors and missing values. New features and dummy variables were created. The missing values in the test set were imputed using information from the train set to avoid any kind of data leakage from test to train.

Feature Selection – Feature selection helps to select the variables which have high predictive power and drops redundant variables. This process improves the predictive performance of machine learning models. There are many ways to select features for different kinds of models and this project uses the **Fisher Score** method for this classification problem. The main idea of Fisher score is to find variables which assign similar values to instances in the same class and different values to instances from different classes. Fisher score is calculated by taking the absolute difference between mean value of defaulters and non-defaulters for each independent variable. This value is divided by the square root of the variance of a variable for respectively defaulters and non-defaulters. 15 variables with the highest Fisher score were selected and used for all the five models.

Cross Validation – The main method used in this project is **holdout cross-validation** in which the data is randomly split into train and test using a 70:30 split. The training data is used to fit the model and the test data is used to evaluate the performance of the model. The reason for doing so is to compare all the models using the same test set to arrive at a fair comparison.
To tune the hyperparameters of decision tree, random forest, and SVM models**, k-fold cross validation** was used on the train set to obtain the optimal parameters. K-fold cross validation is useful for assessing the performance of a model as it gives a range of accuracy scores using different section of the dataset to train and test. It splits the data into k (10 in this case) equal parts. K iterations of training and testing are done. At each iteration one of the k parts is used as the test set and the remaining as the train set. This is done k times so that each part has been used as test once. At the end, an accuracy measure (AUC) in this case is provided for each iteration and also the average over all the iterations.
Cross validation techniques help to see accuracy of the model and the stability (variance) of the model.

Evaluation metric - Predictions are made using the holdout test dataset and evaluated using Area Under the Curve (**AUC**) metric and **accuracy**. AUC is based on the Receiver Operating Characteristics curve (ROC). ROC shows the performance of the classification models across different **thresholds** (the cutoff above which the probability will be considered positive). Its plots the true positive rate against the false positive rate. For example, with a lower threshold, more items will be classified as positive thus increasing the false positive and true positive rate. AUC calculates the entire area under the ROC curve. It can be considered as the probability that the model ranks a random positive example more highly than a random negative example. A good model has an AUC closer to 1 and vice versa. When it is closer to one it means that the model is able to measure the separability of the classes well. AUC of 0.5 means the classifier performs no better than chance.



AUC - ROC Curve [Image 2] (Image courtesy: My Photoshopped Collection)

Accuracy is the number of correctly predicted observations of all the observations. It is calculated as the number of true positives and true negatives divided by the number of true positives, true negatives, false positives, and false negatives

6.2 Modeling

Logistic Regression – The logistic regression algorithm was fit on the train dataset (filtered for only the variables selected through the Fisher score) using the glm function in R. The result showed the variable "PAY_0" to have the most predictive power with highest absolute z value. Predictions were made on both train and test sets, confusion matrices were built and the AUC calculated. The difference in the resulting AUCs was very low which shows that there was likely no overfitting.

Linear Discriminant Analysis (LDA) - The LDA algorithm was fit on the train dataset (filtered for only the variables selected through the Fisher score) using the lda function in R. The fitted LDA model produces a summary output containing **prior possibilities of groups** (proportion of the training observations in each group), **group means** of each variable, and **Coefficients** of linear discriminants (linear combination predictors used to form the decision rule of LDA). Predictions were made on both train and test sets, confusion matrices were built and the AUC calculated. The difference in the resulting AUCs was very low which shows that there was likely no overfitting. The **posterior** probability was also manipulated to see how the number of observations classified as default changes.

Decision Tree – For this model, first a k-fold cross validation of 10 folds was used for hyperparameter tuning of the decision tree classifier using the train set. Different values for the parameters "maxdepth" and "minsplit" were used. Depth is a measure of how many splits a tree can make before coming to a prediction and minsplit is the smallest number of observations in the parent node that could be split further. The cross validation used all the range of values given for these parameters to train and test on different folds of the train data to arrive at the optimal parameters. These optimal parameters were then used to fit a decision tree model on the train set and then predict on the test set. The summary of the trained model shows the variables with high feature importance and the number of nodes, primary splits and surrogate splits. Comparing the AUC on the train and test set shows that there was likely no overfitting.

Random Forest - For this model, first a k-fold cross validation of 10 folds was used for hyperparameter tu ning of the random forest classifier using the train set. Different values for the parameters “ntree”, “mtr y”, and “maxnodes” were used. Ntree is the number of trees to grow, mtry is the number of variables ra ndomly selected to be used at each split, and maxnodes is the Maximum number of terminal nodes trees in the forest can have. The cross validation used all the range of values given for these parameters to trai n and test on different folds of the train data to arrive at the optimal parameters. These optimal parame ters were then used to fit a random forest model on the train set and then predict on the test set. The O OB error rate of the fitted model is 18.52%. Important features were also extracted based on the Gini Ind ex. The large difference in the AUC on the train and test set shows that there is heavy overfitting of the m odel on the train set. This issue can be solved by debugging and using different hyperparameters.

Support Vector Machine - The e1071 library includes a built-in function called tune to perform cross validation. By default, it performs a ten-fold cross-validation on the described models. This project uses the tune function with a range of values for the cost and gamma parameters for a SVM with a radial kernel. The best model obtained from the tune function is then used to make predictions on the test set. The resulting AUCs on the train and test show that the model is highly overfitted.

6.3 Comparing Results

The following table displays the evaluation metrics of the models.

Model	AUC (Train)	AUC (Test)	Accuracy (Train)	Accuracy (Test)
Logistic Regression	0.7162	0.7144	0.809	0.805
LDA	0.7119	0.7089	0.809	0.806
Decision Tree	0.6399	0.6331	0.818	0.816
Random Forest	0.9930	0.7390	0.985	0.809
SVM	0.8728	0.6781	0.845	0.809

Based on the analysis above, logistic regression can be considered the best performing model for this case. Random forest does provide a higher AUC and accuracy but it also overfits on the training data. If this model were to be used to predict on another subset of data it might perform poorly as it has over learned the train set. Therefore, keeping all aspects in mind, logistic regression can be considered best suited for this case.

References

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2014). An Introduction to Statistical Learning: With Applications in R. Springer Publishing Company, Incorporated.

https://www.analyticsvidhya.com/blog/2021/08/conceptual-understanding-of-logistic-regression-for-data-science-beginners/

https://www.theanalysisfactor.com/why-use-odds-ratios/

https://iq.opengenus.org/advantages-and-disadvantages-of-logistic-regression/

https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

https://medium.com/analytics-vidhya/pros-and-cons-of-popular-supervised-learning-algorithms-d5b3b75d9218

https://chirag-sehra.medium.com/decision-trees-explained-easily-28f23241248#:~:text=A%20decision%20tree%20classifier%20is,belonngs%20to%20the%20same%20class.

https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html

https://www.geeksforgeeks.org/linear-discriminant-analysis-in-r-programming/#:~:text=LDA%20or%20Linear%20Discriminant%20Analysis,belongs%20to%20a%20different%20group.

https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

http://theprofessionalspoint.blogspot.com/2019/02/advantages-and-disadvantages-of-random.html

https://www.kdnuggets.com/2020/03/machine-learning-algorithm-svm-explained.html

https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

https://deepai.org/machine-learning-glossary-and-terms/accuracy-error-rate