**APPENDIX INDIVIDUAL STUDENT(ID 45539510)**

50%(22606 obs) of the total observations(45211 obs) were randomly selected then divided into train(13498 obs)[60%], evaluation(4555 obs)[20%] and Test(4553 obs)[20%] dataset. [obs - observations][16 input variables][1 response variable]

**SUPPORT VECTOR MACHINE**

**Methodology:** SVM(SUPPORT VECTOR MACHINE) is a robust regression and classification tool. SVM maximizes the estimation accuracy of a model avoiding the overfitting of the training data. SVM is suitable for analyzing a dataset with a very high number of independent variables(Predictor variables). SVM works by converting the data to high-dimensional feature space so that observations can be classified even observations are not otherwise linearly isolated. The line separating the categories is found then observations are transformed in such a way that the separating line could be represented or drawn as a hyperplane. The function which is used to transform the observations is known as the kernel function. There are many kernel functions available[Linear, Radial basis function, Sigmoid, Polynomial] and can be used to transform the data. Other than separating line between the different categories, a classification SVM also computes the marginal line the defined space between the two categories. The Observations lying on the margins are also known as support vectors. Wider the margin -> better the model will be at predicting the new observations. Narrow margin -> overfitted model.

**(A)Why SVM?:** Convex Optimization is one the characteristics of SVM, because of that Outcome is always a global minimum and not a local minimum(Guaranteed Optimality). It is used for both linearly and non-linearly separable data. Used in a dataset where some observations are labeled, and some are not. There are many kernel options available for feature mapping.

**(B) and (C):** The SVM was performed on IBM SPSS MODELER. No other data preprocessing was carried out especially for SVM. First default settings[simple] of the SVM node was used. Then for the experimental purposes and finding the best performance settings were altered. Kernels like Linear, RBF(Radial basis function), Sigmoid, and Polynomial were tried. As we didn’t have a deeper knowledge of how sigmoid and polynomial works, so they were not carried out furthermore. We also tried to change stopping criteria[1.0E-3 -- 1.0E-6]. Stopping criteria 1.0E-6 was taking so much of computation time and at the end giving the same results as other values, so we decided not to take into consideration. The regularization parameter (known as C/COST) took values like 10, 100, and 1000. Cost even > 100 was taking too much time and was overfitting the training model. RBF gamma took values like 0.1,0.01,0.001[We can’t go lower than this in IBM]. Regression epsilon setting was not considered as it is for regression purposes.**[Different values of parameters in table 1] [Further node setting table 10]**

**(D) & (E):** The results of all the models were then compares based on class accuracies[Table 1]. The best model had the setting Expert Mode, Stopping Criteria = 1.0E-3, Regularization Parameter(c) = 100, kernel type = Linear. There were other combinations of settings which produced the same result, but this was chosen because of faster results.The results from linear kernels seem consistent with different parameter[unlike RBF kernel], that’s why linear kernel with highlighted parameters mentioned above was chosen.

**NAÏVE BAYES CLASSIFICATION**

**(A)Why NAÏVE BAYES?:** It is a common-sense(logic) based technique which is simplest among all the algorithms you would encounter in the field of data science, yet it is so powerful that sometimes it overcomes the other complex algorithms. This modelling technique is all about the probability. It is easy to calculate the probability of an event[which is equal to the number of cases following an event divided by a total number of cases]. Naïve Bayes uses Bayes theorem, which is based on conditional probability and uses the formula P(A | B) = P(A) \* P(B | A) / P(B).

**(B) and (C):** We are carrying out this model in r using ‘e1071’ package. There is one problem with naïve Bayes, if it encounters a new observation totally odd from training data[occurrence of it in training data = 0], then it will assign probability 0 to it, which is not appropriate. That’s why Laplace smoothing is used where one small constant is added just to make sure the occurrence of a total odd event would not be 0 but it will be equal to that constant.

*bank\_train\_bayes <- naiveBayes(term\_deposit\_subscribed ~ ., laplace = 2, data = bank\_notscaled\_train)*

The laplace parameter was changed[1, 2, 3] just to see whether there are any totally odd events in the testing of evaluation data or not. But the results were the same for all the three values.

**(D) and (E):** The results of all the models were then compares based on class accuracies[Table 1]. The best model had the setting The Naïve Bayesian Model with Laplace = 2 was chosen. There were other combinations of settings which produced the same result.

The reason that it is called Naive is not because it is simple or easy to implement. It is because the modelling technique follows a strong assumption about the observations having features independent of each other while in real-time data, it may be not independent. It assumes that the occurrence of one feature in a category is not related to the occurrence of all other features. If this assumption is satisfied, it performs extremely well and better than other complex models. We know that Naïve Bayes performs well when there are categorical variables. When there are numerical data, we can use binning or discretization to convert them to a categorical variable but for this is assumes that continuous variable is coming from a normal distribution.

**Why the above model is not considered:** As we know it assumes normality for numerical variables. Here we have a bunch of numerical variables[balance, day, duration, campaign, pdays, previous]. We tried to apply different transformation like a log, sqrt, box-cox, min-max normalisation. But data was still right skewed. We also tried to use discretize function under ‘arules’ package to carry out binning. The outcome variables with type factor were not making any sense. We were not able to explain the different levels of the factor variables and the assumption of normality was not holding. So, we can’t rely on the integrity of this model. (**Fig 1**)

**Compare two models** [You can look at the whole information(table 4, 5, 6, 7, 8, 9)]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **MODEL** | **Overall Accuracy** | **Class Accuracy(n-‘no’/y-‘yes’)** | | | **Lift(n-‘no’/y-‘yes’)** | | |
| **Train** | **Evaluation** | **Test** | **Train** | **Evaluation** | **Test** |
| SVM | 87.964 | 53.107/92.247 | 54.797/92.250 | 53.947/92.465 | 1.06/4.05 | 1.06/4.11 | 1.06/4.16 |
| Naïve Bayes | 88.930 | 53.484/92.860 | 53.321/93.172 | 56.015/93.285 | 1.06/4.24 | 1.06/4.24 | 1.07/4.49 |

***Table 3 – comparison b/w models***

**Result: (A)**We will choose SVM as it is better at estimating both classes as well as there is a less variation in accuracies compared to Naïve Bayes. And also, for above-mentioned reasons that question the integrity of Naïve Bayes. You can check the gain chart(**Fig 3 & 4**) and also predictor importance(**Fig 2**) for SVM.

**(B) & (C)** The question was whether the client subscribes to the term deposit or not. And the problem was to accurately identify the less frequent category which is ‘yes’[Client subscribes term deposit?]. As we can see at the above table that SVM can clearly identify whether the client will subscribe for approx. 53% of the time with the misclassification error of approx. 47%. SVM was able to identify that the client will not subscribe accurately for approx. 92% of the time with misclassification error of approx. 8%. We can also look at the predictor importance(**Fig 2**), it says that ‘poutcome’ is the most importance variable, which makes sense as ‘poutcome’ represent the results of the previous campaign. Other predictors have low importance amongst all ‘balance’ has the least importance. When we look at the gain chart it says that we can identify class ‘no’ 1.1 times more accurately with the model[**Not that important**] and class ‘yes’ 4.2 times more accurately with the model compared to without the model.

**Conclusion: (A)**The most important finding is that most important predictor is ‘poutcome’, which stands for whether after the previous campaign client subscribed the term deposit or not? Which means the outcome of a campaign is crucial for us. We need to carry out the campaign and make sure to reach more clients. Also, the second important feature is ‘duration’ which indicates how long the client talked to the bank about the related campaign, higher the duration more chance that the client will subscribe. And also, last but not the least important feature ‘month’ which means the month of the year the bank carries out campaign also play an important role in getting more clients.

**(B)** Our model can be used by any financial institute to make sure how they lure can clients in subscribing the term deposit for their scheme or not. For example, which communication mode would be useful, which profession mostly subscribe to the term deposit. They can know their target audience like is it females who are in the management field or retired males. They can also know which month of the year would be appropriate to carry out campaigns. They can also target the audience based on the economic characteristics of the clients.

**(C)** The features that explain the financial status of the clients like income, average debit/credit from a bank account, pattern in payment transactions can also be included. More demographics like the area they live in, the number of children can also be added. The features regarding asset information other than bank balance can also be added. The continuous data is heavily right skewed as well as data has categories like ‘unknown’, which diminishes the model accuracy.

**(D)** Further analysis in what were the reasons for clients not subscribing the term deposit in the previous campaign can be carried out by getting feedback from them. Also using those feedbacks creating a dedicated questionnaire for different target audience can be done.

***Table 1 Result of SVM***

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **MODEL** | **STOPPING CRITERIA** | **regularization parameter(c)** | **RBF gamma** | **kernel type** | **TRAIN\_ACCURACY**  **class(yes-y/no-n)** | **EVAL\_ACCURACY**  **class(yes-y/no-n)** | **TEST\_ACCURACY**  **class(yes-y/no-n)** |
| Simple | 1.0E-3 | 10 | 0.1 | RBF | 59.448/84.889 | 51.476/83.952 | 51.880/83.785 |
| Expert | 1.0E-3 | 10 | - | Linear | 51.161/92.751 | 52.768/93.122 | 51.692/92.912 |
| 100 | 53.107/92.247 | 54.797/92.250 | 53.947/92.465 |
| 1000 | 52.982/92.205 | 54.613/92.250 | 53.759/92.514 |
| 1.0E-4 | 10 | 51.161/92.743 | 52.768/93.122 | 51.692/92.887 |
| 100 | 52.982/92.272 | 54.613/92.225 | 53.759/92.465 |
| 1.0E-5 | 10 | 51.224/92.743 | 52.768/93.122 | 51.692/92.887 |
| 100 | 52.982/92.272 | 54.797/92.225 | 53.571/92.465 |
| 1.0E-3 | 10 | 0.1 | Sigmoid(bias = 0.0, gamma = 1.0) | 6.403/93.314 | 6.089/92.923 | 7.143/93.211 |
| 1.0E-3 | 10 | 0.1 | Polynomial(bias = 0.0, gamma = 1.0, degree = 3) | 64.093/77.925 | 49.077/76.452 | 50.188/75.230 |
| 1.0E-3 | 10 | 0.1 | RBF | 59.448/84.889 | 51.476/83.952 | 51.880/83.785 |
| 100 | 68.675/79.168 | 54.613/77.523 | 59.398/76.946 |
| 1000 | 71.877/75.674 | 59.225/74.134 | 62.218/73.191 |
| 10 | 0.01 | 18.267/98.597 | 18.635/98.904 | 18.985/98.607 |
| 100 | 52.982/90.743 | 52.030/90.057 | 51.128/90.251 |
| 1000 | 57.878/85.754 | 52.030/84.999 | 53.571/85.053 |
| 10 | 0.001 | 18.267/98.597 | 18.635/98.879 | 18.797/98.632 |
| 100 |
| 1000 |
| 1.0E-4 | 10 | 0.1 | RBF | 59.448/84.880 | 51.292/83.952 | 51.880/83.785 |
| 100 | 68.675/79.168 | 54.613/77.523 | 59.398/76.946 |
| 1000 | 71.877/75.674 | 59.225/74.134 | 62.218/73.191 |
| 10 | 0.01 | 18.267/98.597 | 18.635/98.904 | 18.985/98.607 |
| 100 |
| 1000 |
| 10 | 0.001 | 18.079/98.614 | 18.450/98.904 | 18.797/98.657 |
| 100 |
| 1000 |
| 1.0E-5 | 10 | 0.1 | RBF | 59.448/84.880 | 51.292/83.952 | 51.880/83.785 |
| 100 | 68.675/79.168 | 54.613/77.523 | 59.398/76.946 |
| 1000 | 71.877/75.674 | 59.225/74.134 | 62.218/73.191 |
| 10 | 0.01 | 18.267/98.597 | 18.635/98.904 | 18.985/98.607 |
| 100 | 52.982/90.743 | 52.030/90.057 | 51.128/90.251 |
| 1000 | 57.878/85.754 | 52.030/84.999 | 53.571/85.053 |
| 10 | 0.001 | 57.878/85.737 | 52.030/85.024 | 53.571/85.053 |
| 100 |
| 1000 |

***Table 2 Naïve Bayes Results***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MODEL** | **LAPLACE** | **TRAIN\_ACCURACY**  **class(yes-y/no-n)** | **EVAL\_ACCURACY**  **class(yes-y/no-n)** | **TEST\_ACCURACY**  **class(yes-y/no-n)** |
| Naïve Bayes Classification | 1 | 53.484/92.860 | 53.321/93.172 | 56.015/93.285 |
| 2 |
| 3 |

***Table 4* Training table for SVM**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **term\_deposit\_subscribed** |  | **no** | **yes** | **Total** | **Lift** |
| **no** | Count | 10982 | 923 | 11905 | 1.061599 |
|  | Row % | 92.247 | 7.753 | 100 |  |
|  | Column % | 93.631 | 52.176 | 88.198 |  |
|  | Total % | 81.36 | 6.838 | 88.198 |  |
| **yes** | Count | 747 | 846 | 1593 | 4.05225 |
|  | Row % | 46.893 | 53.107 | 100 |  |
|  | Column % | 6.369 | 47.824 | 11.802 |  |
|  | Total % | 5.534 | 6.268 | 11.802 |  |
| **Total** | Count | 11729 | 1769 | 13498 |  |
|  | Row % | 86.894 | 13.106 | 100 |  |
|  | Column % | 100 | 100 | 100 |  |
|  | Total % | 86.894 | 13.106 | 100 |  |

***Table 5* Evaluation table for SVM**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **term\_deposit\_subscribed** |  | **no** | **yes** | **Total** | **Lift** |
| **no** | Count | 3702 | 311 | 4013 | 1.064605 |
|  | Row % | 92.25 | 7.75 | 100 |  |
|  | Column % | 93.793 | 51.151 | 88.101 |  |
|  | Total % | 81.273 | 6.828 | 88.101 |  |
| **yes** | Count | 245 | 297 | 542 | 4.105272 |
|  | Row % | 45.203 | 54.797 | 100 |  |
|  | Column % | 6.207 | 48.849 | 11.899 |  |
|  | Total % | 5.379 | 6.52 | 11.899 |  |
| **Total** | Count | 3947 | 608 | 4555 |  |
|  | Row % | 86.652 | 13.348 | 100 |  |
|  | Column % | 100 | 100 | 100 |  |
|  | Total % | 86.652 | 13.348 | 100 |  |

***Table 6* Testing table for SVM**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **term\_deposit\_subscribed** |  | **no** | **yes** | **Total** | **Lift** |
| **no** | Count | 3718 | 303 | 4021 | 1.062304 |
|  | Row % | 92.465 | 7.535 | 100 |  |
|  | Column % | 93.818 | 51.356 | 88.315 |  |
|  | Total % | 81.66 | 6.655 | 88.315 |  |
| **yes** | Count | 245 | 287 | 532 | 4.163091 |
|  | Row % | 46.053 | 53.947 | 100 |  |
|  | Column % | 6.182 | 48.644 | 11.685 |  |
|  | Total % | 5.381 | 6.304 | 11.685 |  |
| **Total** | Count | 3963 | 590 | 4553 |  |
|  | Row % | 87.042 | 12.958 | 100 |  |
|  | Column % | 100 | 100 | 100 |  |
|  | Total % | 87.042 | 12.958 | 100 |  |

***Table 7* Training table for Naïve Bayes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **term\_deposit\_subscribed** |  | **no** | **yes** | **Total** | **Lift** |
| **no** | Count | 11055 | 850 | 11905 | 1.062586 |
|  | Row % | 92.86014 | 7.139857 | 100 |  |
|  | Column % | 93.71821 | 49.94125 | 88.19825 |  |
|  | Total % | 81.90102 | 6.297229 | 88.19825 |  |
| **yes** | Count | 741 | 852 | 1593 | 4.241639 |
|  | Row % | 46.51601 | 53.48399 | 100 |  |
|  | Column % | 6.28179 | 50.05875 | 11.80175 |  |
|  | Total % | 1.428 | 10.257 | 11.80175 |  |
| **Total** | Count | 11796 | 1702 | 13498 |  |
|  | Row % | 87.39072 | 12.60928 | 100 |  |
|  | Column % | 100 | 100 | 100 |  |
|  | Total % | 87.39072 | 12.60928 | 100 |  |

***Table 8* Evaluation table for Naïve Bayes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **term\_deposit\_subscribed** |  | **no** | **yes** | **Total** | **Lift** |
| **no** | Count | 3739 | 274 | 4013 | 1.063125 |
|  | Row % | 93.17219 | 6.82781 | 100 |  |
|  | Column % | 93.66232 | 48.66785 | 88.10099 |  |
|  | Total % | 82.08562 | 6.015368 | 88.10099 |  |
| **yes** | Count | 253 | 289 | 542 | 4.313984 |
|  | Row % | 46.67897 | 53.32103 | 100 |  |
|  | Column % | 6.337675 | 51.33215 | 11.89901 |  |
|  | Total % | 1.428 | 10.257 | 11.89901 |  |
| **Total** | Count | 3992 | 563 | 4555 |  |
|  | Row % | 87.63996 | 12.36004 | 100 |  |
|  | Column % | 100 | 100 | 100 |  |
|  | Total % | 87.63996 | 12.36004 | 100 |  |

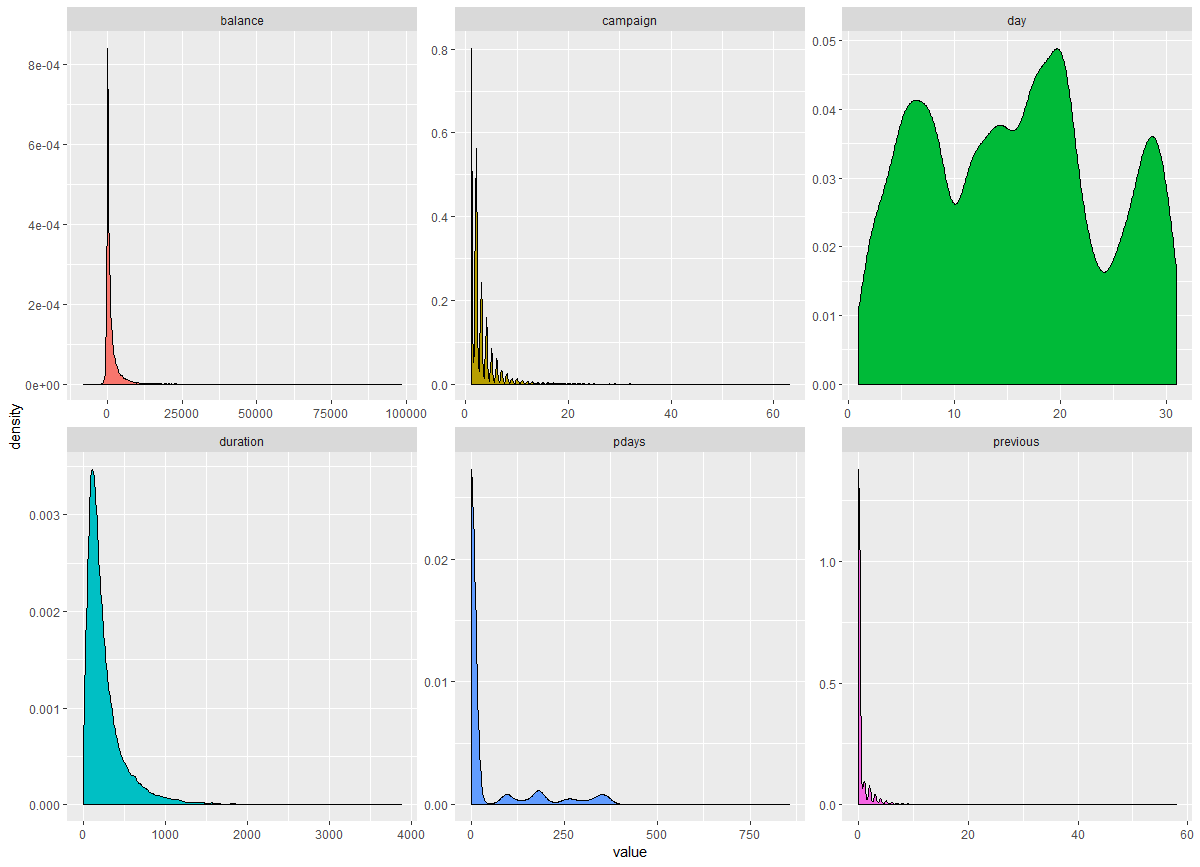
***Table 9* Testing table for Naïve Bayes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **term\_deposit\_subscribed** |  | **no** | **yes** | **Total** | **Lift** |
| **no** | Count | 3751 | 270 | 4021 | 1.065816 |
|  | Row % | 93.28525 | 6.714748 | 100 |  |
|  | Column % | 94.12798 | 47.53521 | 88.3154 |  |
|  | Total % | 82.38524 | 5.930156 | 88.3154 |  |
| **yes** | Count | 234 | 298 | 532 | 4.490079 |
|  | Row % | 43.98496 | 56.01504 | 100 |  |
|  | Column % | 5.87202 | 52.46479 | 11.6846 |  |
|  | Total % | 1.428 | 10.257 | 11.6846 |  |
| **Total** | Count | 3985 | 568 | 4553 |  |
|  | Row % | 87.52471 | 12.47529 | 100 |  |
|  | Column % | 100 | 100 | 100 |  |
|  | Total % | 87.52471 | 12.47529 | 100 |  |

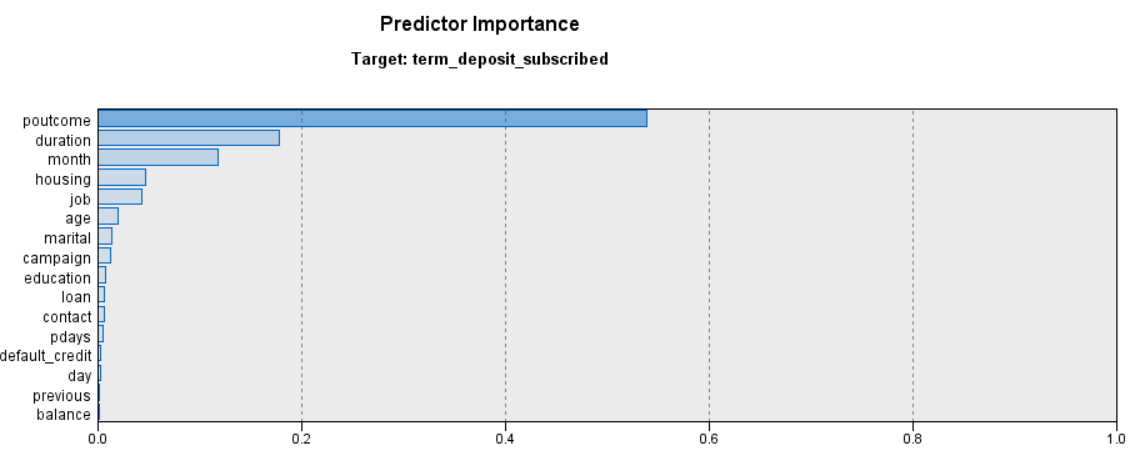
**Table *10* IBM nodes’ general settings**

|  |  |
| --- | --- |
| IBM NODE | GENERAL SETTING |
| SVM | Use predefined roles  Uncheck the *use partition data*  Uncheck the *build model for each split*  Calculate the *predictor importance* |
|  |  |

**Fig 1 Density plot of continuous variables[Not normal distribution]**



**Fig 2 Predictor importance for SVM**



**Fig 3 Gain chart class ‘no’ Fig 4 Gain chart class ‘yes’**

