Stat 441/841 Final Project - Fall 2019

Jacob Raymond 20831748
Hieu Nguyen 20532698
Dingding Hu 20651178
November 30th, 2019

Contents

Executive Summary **1**

1 Introduction **1**

2 Data **1**

3 Preprocessing & Feature Engineering **2**
3.1 Missing data 2
3.2 Imputation for Numerical Columns 3
3.3 Imputation for Categorical Columns 3
3.4 Structure Training, Validating & Testing 7

4 Methodology **8**
4.1 Models Considered 8
4.1.1 Naive Bayes 8
4.1.2 Random Forest 8
4.1.3 Logistic Regression 10
4.1.4 Generalized Additive Model 11
4.1.5 Support Vector Machine 12
4.1.6 Gradient Boosting 12
4.2 Comparison between models 15
4.3 Model Averaging 15

5 Conclusions **16**

6 Future Work **17**
6.1 Bayesian Model Averaging 17
6.2 GAM Modelling 17
6.3 Factor Importance 17
6.4 Neural Network 17

7 Contribution **18**

A The Bank Marketing Dataset

B Literature Review

1 Introduction

The financial sector is in the midst of a transformation. The Canadian industry, long dominated by the "Big Five" (RBC, CIBC, BMO, TD, and Scotia), is facing increasing competition with international banks and novel technologies such as robo-advisors. This shift is compounded by socioeconomic phenomenon including the foreboding of a North American recession and the global resurgence of populism. Hence, more than ever, financial institutions must identify ways to recruit and retain a loyal customer base. Targeted phone-based campaigns have long been useful tools to this end.

In this project, we aim to discover the attributes of a successful telephone-based marketing campaign promoting financial services. In particular, we are interested in identifying the characteristics of clients who have subscribed to a term deposit with a bank after being contacted by a customer engagement representative. We will be utilizing campaign led by a Portuguese bank between 2008 and 2010 to fit a variety of classification models.

2 Data

We analyzed the "Bank Marketing" dataset published by Moro et al. to supplement their 2014 paper "A Data-Driven Approach to Predict the Success of Bank Telemarketing". The researchers collected the data from a Portuguese bank that routinely performed telephone-based marketing campaigns between 2008 and 2010. They subsequently donated a reduced data set comprised of 11,162 observations and 17 variables for public interest. We obtained the dataset from Kaggle. See appendix A for additional details.

We are interested in developing a model to determine whether or not prospected clients will make a long-term deposit with the bank. Thus, we will use the binary "Deposit" variable as our response, meaning our classification model will have two classes. A priori, both outcomes are roughly proportioned equally: 52% of observations are of type "No" will the other 48% are of type "Yes". A presentation of the other 16 explanatory variables are composed of can be found in Appendix A.

The data was published in an entirely clean state. There are no missing observations, although some values were inputted as "Unknown". None of the variables exhibited bivariate correlation superior 0.2 aside from "pdays" (the number of days that passed between the latest two campaigns in which the client was contact) and "previous" (the number of campaigns where the client was contacted before the current campaign). While the correlation is 0.51, this is to be expected: over two third of clients had only been contacted once, meaning they were assigned pdays=-1 (code for observations that were not previously contacted) and previous=0. More details regarding the statistics summary of the data will be discussed in Section 3 of the report.

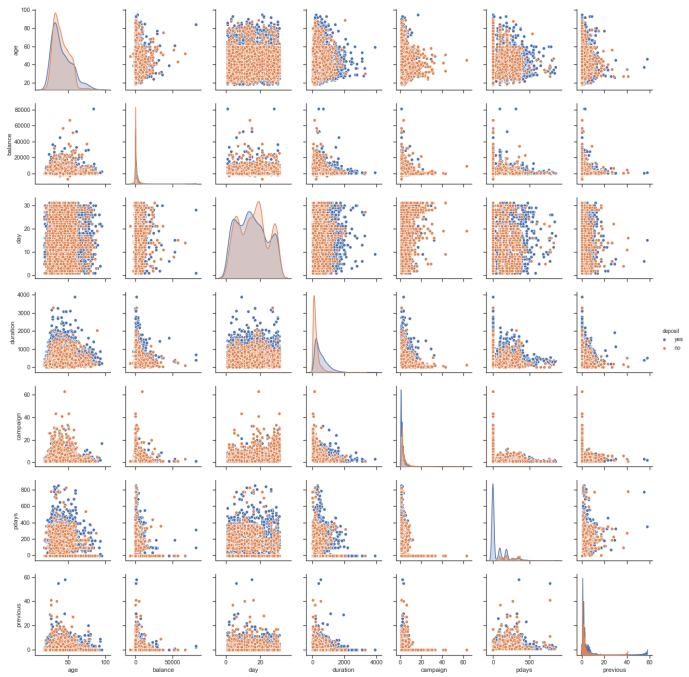

Figure 1: Plot of the data. Orange represents "No Deposit", blue represents "Yes Deposit". There is severe overlap between both classes for every variables.

3 Preprocessing & Feature Engineering

3.1 Missing data

The data was download directly from Kaggle where it has been carefully cleaned and ready to be processed. Hence, there are no missing values in this data.

	column_name	percent_missing (%)
age	age	0.0
day	day	0.0
poutcome	poutcome	0.0
previous	previous	0.0
pdays	pdays	0.0
campaign	campaign	0.0
duration	duration	0.0
month	month	0.0
contact	contact	0.0
job	job	0.0
loan	loan	0.0
housing	housing	0.0
balance	balance	0.0
default	default	0.0
education	education	0.0
marital	marital	0.0
deposit	deposit	0.0

Figure 2: Summary of missing data

3.2 Imputation for Numerical Columns

	age	balance	day	duration	campaign	pdays	previous
count	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000
mean	41.231948	1528.538524	15.658036	371.993818	2.508421	51.330407	0.832557
std	11.913369	3225.413326	8.420740	347.128386	2.722077	108.758282	2.292007
min	18.000000	-6847.000000	1.000000	2.000000	1.000000	-1.000000	0.000000
25%	32.000000	122.000000	8.000000	138.000000	1.000000	-1.000000	0.000000
50%	39.000000	550.000000	15.000000	255.000000	2.000000	-1.000000	0.000000
75%	49.000000	1708.000000	22.000000	496.000000	3.000000	20.750000	1.000000
max	95.000000	81204.000000	31.000000	3881.000000	63.000000	854.000000	58.000000

Figure 3: Data Summary

Looking at the summary (mean, median, min, max, std) of the numerical features, under the *balance* column, we observe a biggest spread among any other features, indicating by the largest range. This is due to the fact that it contains outliers from both negative and positive ends. With such high spread along with the fact that these outliers only takes up a small fraction of the whole dataset (but not tiny enough to get rid of them completely), we decided to apply log transformation on this set to squash the huge spread under this column. Before log transformation, the absolute value of the min plus, namely $6847 + 1 = 6848$, is added to every single element under *balance* column. This not only ensures that the center of the data will be shifted to the right towards the positive end, it also reduces the range of the data by a reasonable factor. Below are the distribution plots before and after transformation; the latter looks much better:
Next, under the *pdays*, we notice that the value -1 takes up the largest proportion among any other values. This value, as mentioned above, indicates that the relevant clients had never been contacted previously. Because it is the mode for this column, we replace it by 999 indicating the magnitude of how long a client was last contacted.

At the end, due to the variety in unit measurement and range in numerical variables, we apply **Standard Scaler** on all numerical columns to further center the data to standard normal with mean of 0 and standard deviation of 1. This ensures that our data is in consistent scale and make optimization a lot more efficient this way. **Important Note:** the test are standardized using the mean and standard deviation calculated from the trainset to avoid explicit cheating.

3.3 Imputation for Categorical Columns

First, since we're interested in the getting potential clients to say yes and open an account, lets take a look at the frequency of clients response over various categorical

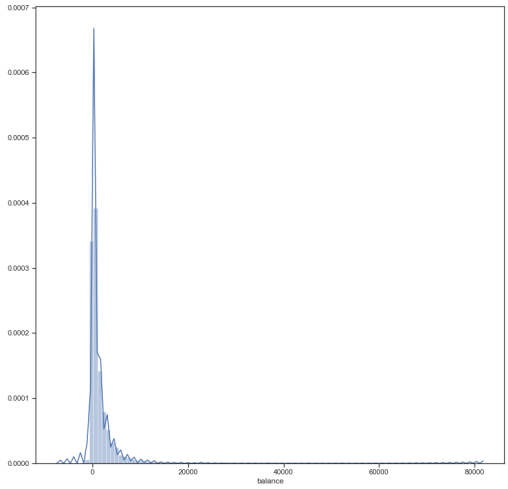

Figure 4: Original Balance

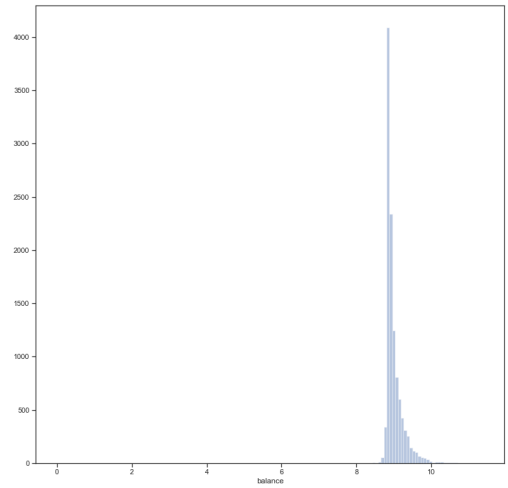

Figure 5: Transformed Balance

variables.

Out of all of these categorical variables, a few stands out in terms of relative strong proportion of clients saying "yes". First of all, under the *job* category, a few professions actually have a higher chance to make a deposit than others. For instance, we observe that out of all the students contacted, roughly 75% of them said "yes";

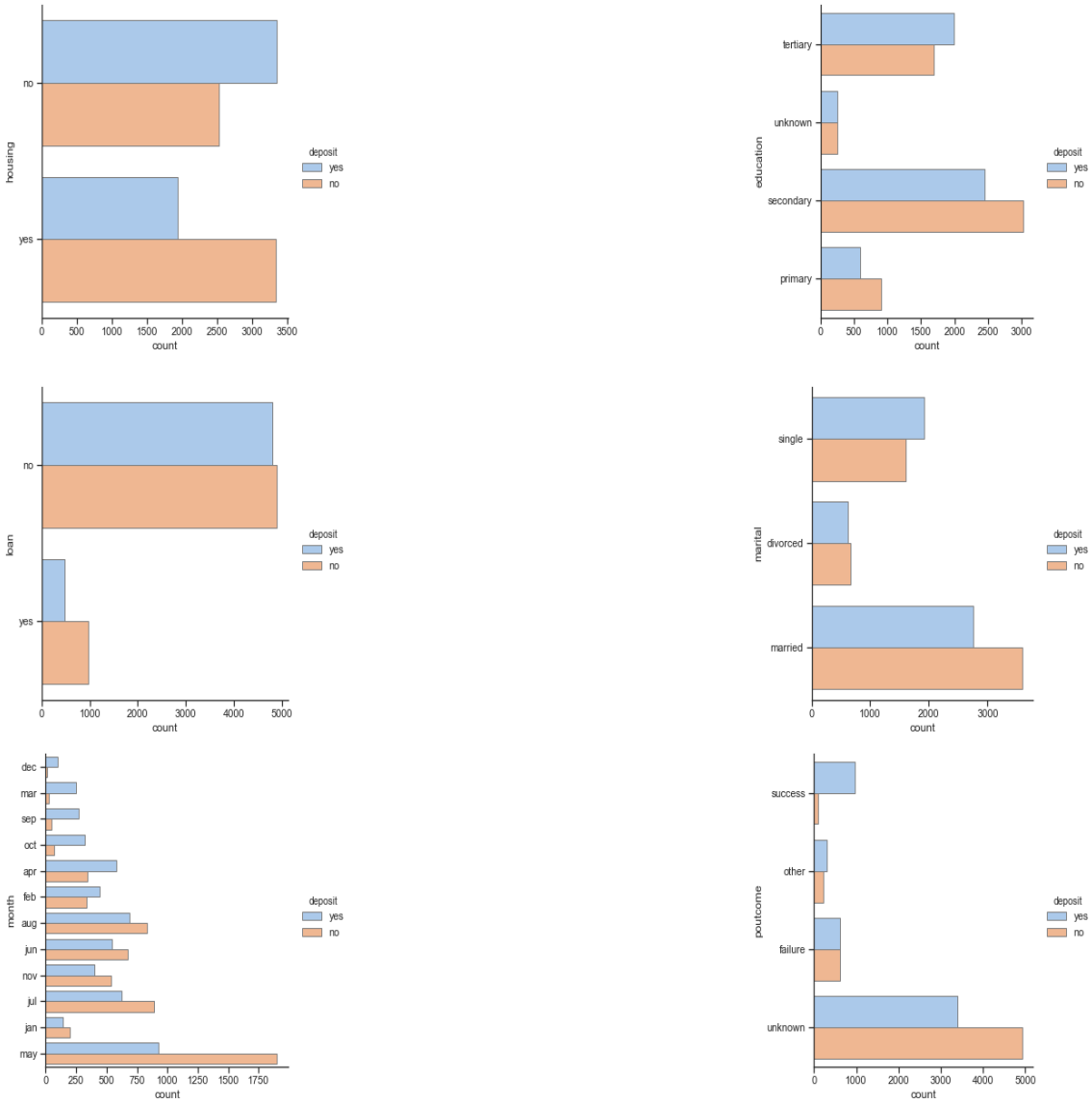

Figure 6: Categorical variables based on Y/N response

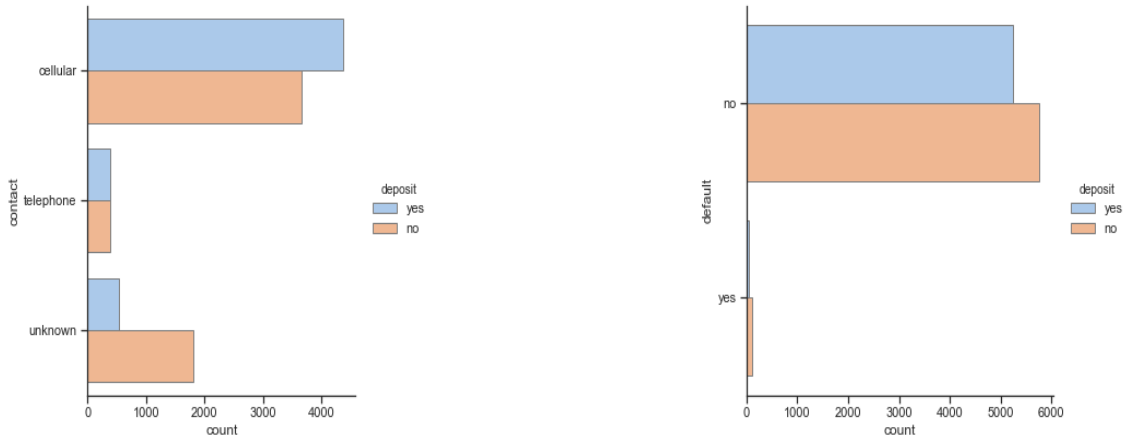

Figure 7: Categorical variables based on Y/N response

around 66% of retired clients make the deposits. However, these 2 professions only takes up to about 1/8 of the sample that we are working with. Secondly, under the *month* category, some months turn out to have more clients saying "yes" than other months. Some of the noticeable months sorted by "yes" to "no" ratios are December, March, September, October, April and February. We find this very interesting, as most of these month correspond to the end of financial quarters, when publicly traded typically scurry to offer favourable results to their investors. Thus, instead of having all 12 months as 12 different categories, we grouped them in to 3 different groups: *month_EarlyQuarter*, *month_MidQuarter*,*month_EndQuarter*. For example, January, April, July and October will be group into *month_EarlyQuarter* representing the beginning of each quarter. Next, we believe that leave the *day* as numerical might not contribute meaningful effect towards a good prediction. This is simply because these are just the day in a given month and do not imply ordinal meaning. Hence, similar to what we perform on the *month*, we group these days into 3 main groups based the density of "yes" to "no" odds.

We can see that when we look at the odds given days, most of the odds are between 2/3 an 3/2. There are two days [1,10] that odds of yes vs no is a lot higher than the others and 5 days that has the odds of no vs yes is a lot higher than the others [19,20,28,29,31]. So it is reasonable to group "days" into these three groups. Furthermore, we also notice that some subcategories are labelled as *unknown*, which only take up a small proportion among other major subcategories in each category. Therefore, the *unknown* values are replaced by the **mode** in each category.

After all the necessary transformations were completed, we applied *One Hot Encoding* on all categorical features. One important point worth mentioning is the consequence of applying *One Hot Encoding* - it automatically introduces *multicollinearity*. To avoid this, we arbitrarily dropped one of the encoded columns from each category

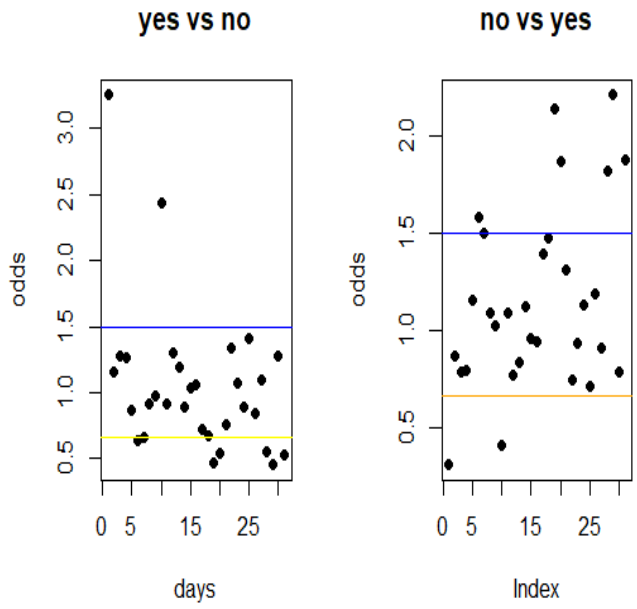

Figure 8: Y/N Odds by Day

through the **drop_first** argument in Python.

3.4 Structure Training, Validating & Testing

The way we structure the data before fitting any models is as follow:

- Data is randomly shuffled and divided in to train, valid and test sets with 70%/15%/15% ratio. The reason why we include the valid set is to even further assess overfitting and evaluation of the ensemble approach after all models are trained and validated.
- Standardization is applied. The validation and test sets are standardized using the mean and standard deviation computed from the training set
- All hyperparameters are tuned using nested cross validation on the training set or out-of-bag errors

4 Methodology

4.1 Models Considered

We fitted the processed data to a variety of models. We will then classify the observations in the test set using each of these models and present a final classification decision using a model averaging procedure.

4.1.1 Naive Bayes

Naive Bayes is an approach in which we assume that the features are all independent for a given class. Thus, the marginal densities for every class can be estimated independently using one-dimensional kernel densities. This has the advantage of essentially eliminating the biases that could affect each individual marginal, resulting in a model that adheres well to the training data while maintaining a reasonable level of variance.

A preliminary exploration of the data suggests that it would be well suited to a naive Bayes model. There is a low level of correlation(Figure 10) between the explanatory variables, granting credibility to the assumption of independence. Moreover, the marginal distributions of the response variables(Figure 9) do not appear to follow any common distributions, suggesting that a non-parametric kernel approach is appropriate.

We also considered fitting a naive Bayes model to the principal component of the data. Given the fact that the principal components are uncorrelated while conserving all of the data's information, running a PCA would bolster our confidence that the assumptions of the naive Bayes model are met.

4.1.2 Random Forest

Even after applying PCA, we cannot be certain that the naive Bayes approach will prevent over-fitting because of the curse of dimensionality. Thus, we also considered using a random forest model. In this approach, we generate a large number N bootstrap samples from the training data, and use them to fit N classification trees. For each tree, at every node, we only consider a random subset of size m of the explanatory variables. The class prediction is determined through majority vote. The random forest approach is unlikely to overfit the data. Indeed, by averaging multiple individual trees, which are prone to low bias but high variance. Random forests represents a significant improvement in terms of variance over individual trees despite introducing a small amount of bias.

We utilized the **randomForest** package in R. We therefore had to tune two hyperparameters: **mtry**, the number of variables randomly sampled at each split, and **ntrees**,

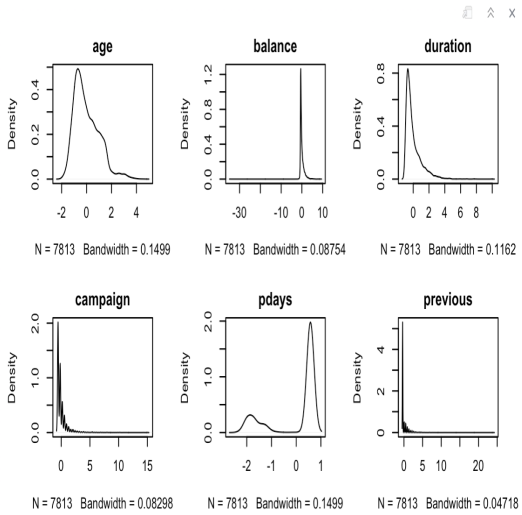

Figure 9: Kernel density estimate of each numerical feature

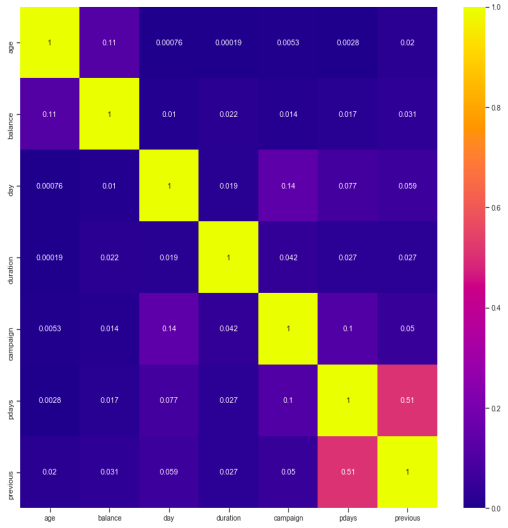

Figure 10: Correlation between numerical variables

the number of trees to grow in the forest. Since the out-of-bag error tends to stabilize as the number of trees increases, this second parameter will ensure that we do not waste computational power. We considered four candidates for **ntree**: 250, 500, 750, and 1000. We also considered the integers 1 to 6 as possible candidate values for **mtry**. These choices come from the convention of taking a number of variables equal to the root of the number of factors (in our case, $\sqrt{34} \approx 6$). We calculated the out-of-bag (OOB) error for each of the 24 combinations of candidates. Cross-validation was not necessarily since the OOB procedure tests the model on data it was not trained on; namely, each observation is tested with the set of trees that were trained on bootstrap samples that the observation wasn't a part of. Over all, we found that

fitting a model with 5 variables per split and 750 trees resulted in the smallest OOB error.

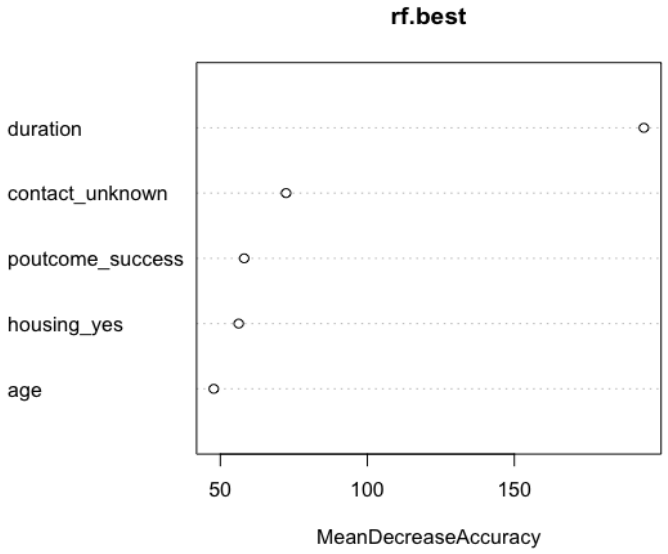

Figure 11: Feature Importance

As we can see in figure 11, the most important features are, in descending order, *duration*, *contact_unknown*, *poutcome_success*, *housing_yes*, and *age*.

4.1.3 Logistic Regression

The naive Bayes method we tried has resulted in a posterior distribution for *deposit*. Thus, we can calculate the log odds and compare it to numerical explanatory variables.

The log odds compared to the numerical features may be linear in *campaign*, *age*, and *duration*. Hence, logistic regression might be appropriate for this data. In this kind of model, we seek to evaluate the probability that a given observation belongs to a baseline class over another. We suppose that the probability that an observation belongs to the baseline class can be modelled using the a logistic regression: $p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$. It can be shown that, in this binary model, the log odds are linear:

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

The coefficients β are estimated numerically using maximum likelihood estimation.

In order to utilize the **multinom** function from the **nnet** R package to fit this logistic regression model, we first had to rescale the numerical variables so they fall within the hypercube with sides of length $[0, 1]$. Moreover, given the fact that our data contained

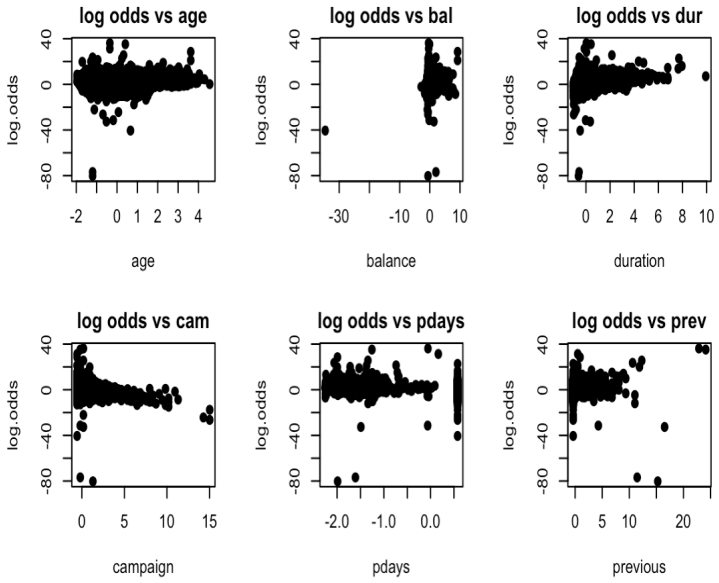

Figure 12: Log odds vs numerical feature

a large number of explanatory variables, we decided to use LASSO regularization in order to prevent overfitting by introducing a small amount of bias in the model. We thus had one parameter to tune: the penalty term. We used five fold cross validation and considered 100 likely candidates. Since we value a parsimonious model, we identified both the tuning parameter that resulted in the minimum error and the one associated sparsest model within one standard-error of the minimum. Our estimates are
$\lambda_{min} = 0.000339$

$\lambda_{1SE} = 0.010588$

4.1.4 Generalized Additive Model

Similarly to the intuition that lead us to try logistic regression, we can notice that the log odds compared to a handful of the features (namely *balance*, *pday*, and *previous*) appear to be non-linear. We thus explored the possibility of using a generalized additive model (GAM), in which the log odds are expressed as the sum of p univariate smooth functions fitted to each explanatory variable. Hence, considering that categorical variables cannot be fitted using a generalized additive model, we had the following model

$$DEPOSIT \sim s(AGE)+s(BALANCE)+s(CAMPAIGN)+s(PDAYS)+s(PREVIOUS)$$
$$+JOB + MARITAL + EDUCATION + DEFAULT$$

$$+CONTACT + DAY + MONTH + POUTCOME$$
$$+HOUSING + LOAN$$

4.1.5 Support Vector Machine

We implemented the support vector machine algorithm using the **e1071** package in R. This method functions by identifying an optimal separation function to segregate the two binary classes. Here, the sign that the function takes for an observation gives us its predicted class. This separation function is calculated by taking the pairwise inner product of basis functions for the points in our training set. We will classify the observations in our dataset using a radial Gaussian kernel. The separation function takes the form

$$\hat{f}(\mathbf{x}) = \hat{\beta}_0 + \sum_{i=1}^{n} \hat{\alpha}_i y_i e^{-\gamma||\mathbf{x}-\mathbf{x_i}||}$$

We have two hyperparameters to tune:

- γ, which quantifies the importance that observations far from $\mathbf{x}$ will have in the decision, as opposed to closer observations. This parameter should be quite small, and thus we consider the first six negative powers of 10 (i.e. 10^{-a} for $a = 0, 1, 2, 3, 4,$ and5
- C, the cost parameter, which determines the size of the hyperplane's margin. Here, we want to consider both the impact of small values of C (leading to larger margins at the risk of misclassification) as opposed to larger values (which results in a tighter fit and and thus a model that hardly generalizes). Therefore, we consider the powers of ten from 10^{-5} up to 10^5, resulting in 11 candidates.

We determined optimal values for these hyperparameters using 10-fold cross validation through the **tune.svm** function. The values that achieved the smallest CV error are are $\gamma = 0.001$ and $C = 100$.

4.1.6 Gradient Boosting

We implemented gradient boosting using **gbm** package in R. Unlike any other tree based algorithms where trees are grown level-wise, gradient boosting grows the trees leaf-wised. On a high level, it starts by constructing a single leaf as an initial guess of the response *deposit*, then sequentially builds the tree based on the classification error being made by the previous tree. This process continues until no further improvement can be made. Our gradient boosting produces a misclassification rate of 0.1743284 on the test set.

Regarding hyperparameter tuning, our procedure is structured as follow:

- Grid search scope:
  - shrinkage: 0.001 to 0.01 with step size of 0.001
  - bag fraction: from 0.3 to 0.9 with step size of 0.01
- For each combination of hyperparameters in the defined grid, we performed 10-folds cross validation on the training data. The hyperparemeters, mean CV error as well as the mean optimal number of trees were recorded at the end. The process continued to the next combination in the grid
- The combination of hyperparameters that produced the lowest mean cv error was selected to fit the final model

Below is the set of hyperparameters we used for our final gradient boosting model:

- shrinkage: 0.005
- bag fraction: 0.4
- mean cv error: 0.1709969
- training error: 0.1510303
- validation error: 0.160693

The importance of a predictor is measured by the mean increase/improvement in prediction error as we sequentially construct the boosted trees in our model. The more a feature is used to make key splits in the decision trees, the higher the relative influence will be. The top 5 ranking of importance for this question is as follow: *duration, poutcome_success, contact_unknown, pdays, age, balance*. By taking a look at the correlation plot between these features versus the deposit outcome, *duration* has the highest correlation (=0.45) with the *deposit* outcome relative to other features. Its relative influence compared to other features is at least by a factor of 6 as we can see from the importance plot.

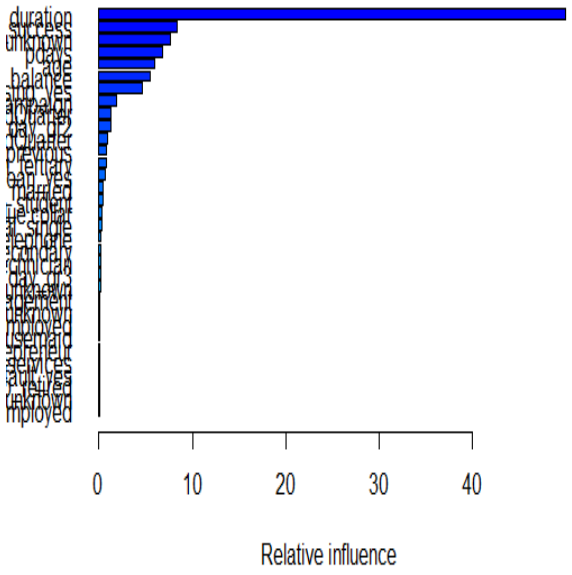

Figure 13: Feature Importance

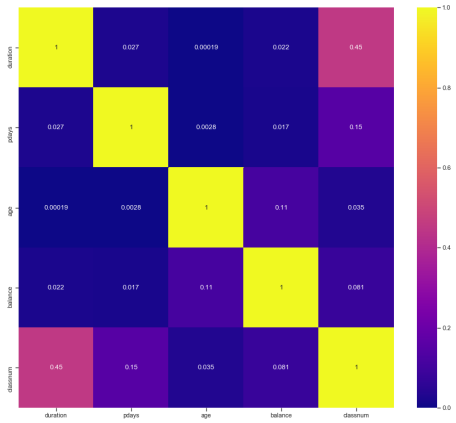

Figure 14: Absolute Correlation between most important features and class label

4.2 Comparison between models

Model	Training Error	Validation Error
Generalized Additive Model	0.291	0.306
Gradient Boosting	0.157	0.167
Logistic Regression	0.185	0.187
Logistic Regression (1SE)	0.191	0.193
Logistic Regression (CV-MIN)	0.186	0.189
Naive Bayes	0.252	0.262
Naive Bayes (PCA Rotation)	0.214	0.244
Random Forest	0.020	0.167
Support Vector Machine	0.164	0.177

As we can see, all of the models we considered have reasonable training and validation errors. The generalized additive model had, over all, the highest training and validation misclassification rates. We could explain this lackluster performance by attributing it with the lack of interaction terms. The two naive Bayes model did not fare much better, having the second and third worst validation errors despite PCA rotation. As we previously mentioned, in this class of algorithms, we assume that the factors are independent and therefore uncorrelated.

On the flipside, the random forest model performed the best out of all the approaches we considered. Its training error of less than 2% outperformed the other models by a wide margin. While such a low training error could suggest overfitting, random forests had a validation error of only 16.7%, which is once more the lowest. Gradient boosting and SVM also performed well over this data.

This analysis highlights that the interaction between parameters is crucial to a model's performance, especially as the dimension increases. Assuming that factors are independent is restrictive and unrealistic in a 34-dimensional space. Although the correlation plot indicated a low correlation between the numerical variables, the majority of factors in our dataset were categorical and, thus, we are unable to discount the idea that they might possess some co-interaction. We believe this explains the relative poor performance of the GAM and naive Bayes models: in the former approach, we do not consider interactions between factors, while in the later, we assume the explanatory variables are independent. The random forest, meanwhile, was the best performing approach. Tree models are heavily reliant on interactions between parameters, since each split is based on the value taken by one of the factors. As such, the value of every terminal node is determined by a slew of variables.

4.3 Model Averaging

A way to produce a final algorithm is to use the models we have produced before and use the idea called "majority vote". Different methods may have different per-

formances on different data points and we believe that the class label of a data point that has been predicted most frequently is a better prediction than other class labels. Assume that we hvae 2n+1 algorithms, denote as $\hat{G}_1, \hat{G}_2, ..., \hat{G}_{2n+1}$ .Here we assume odd number of algorithms to avoid the case that we may have the same number of votes for two different class labels since we have only two classes here. For each data point x_i to be predicted, each algorithm $\hat{G}_i$ will produce one predicted class label denote as $\hat{G}_1(x_i), \hat{G}_2(x_i), ..., \hat{G}_{2n+1}(x_i)$. So our final prediction on x_i will be "yes" if more than n of $\hat{G}_i(x_i)$ ="yes", and "no" otherwise. In our case, we have 7 algorithms produced and for a data point, we predict the class label to be the class produced by 4 or more algorithms. We can see that the validation error is 0.1696535 which means it does not work better than random forest or gradient boosting. So we may consider still using the best single model we get from previous result.

Model	Validation Error
Majority Vote	0.1696535

5 Conclusions

Within the banking industry, capital liquidity plays a significant role in the survival and competitive edge of a business. Therefore, one of the main of objectives for banks is to figure out an effective methodology in order to raise more capital via long-term deposits. However, this turns out to be a very challenging task since its success ties closely to how well one can optimize the potential targets for telemarketing. In this study, we analyzed the Portuguese Bank Marketing dataset to try to come up with the most effective and efficient classification technique to identify potential clients to make long-term deposits. In this project, we fitted and evaluated a total of 9 different models, each of which was evaluated and ranked based on the validation error and test error. Overall, the top three best performing candidates based on validation and test misclassification error are *Random Forest (RF)*, *Gradient Boosting (GB)* and *Support Vector Machine (SVM)*, which produces validation errors of 0.167, 0.167 and 0.177 respectively. These techniques outperform the other 6 candidates by a wide margin. Also we notice that the majority vote of 7 models works approximately the same with the best 3 single models but not as good as random forest and gradient boosting algorithms.So we may consider just by using a single model as our final model. Now the fact that *Random Forest (RF)* does slightly better than *Gradient Boosting (GB)* is something that surprises us. Generally speaking, one down side of boosted trees is that it is a lot harder and computationally expensive to tune than *Random Forest (RF)*. This is due to its larger number of hyper parameters as compared to random forest. Since boosting performs optimization in the objective function base rather in parameter space, the way it sequentially learns makes boosting a better candidate than *Random Forest (RF)*. With *Random Forest (RF)*, trees are trained independently from each other using the fundamental of boostrap and bagging. This makes random forest less sensitive to overfitting than boosting. So without the test

result, we would have guessed that *Gradient Boosting (GB)* will do a better job than *Random Forest (RF)*. As mentioned previously, the performance of GAM and naive Bayes models are relatively poor because of the heavy assumption of independence among features and the negligence of interaction effects in high dimensional feature space. Therefore, in conclusion, *Random Forest (RF* is our final model.

Model	Test Error
Random Forest	0.1731343

6 Future Work

6.1 Bayesian Model Averaging

We could have averaged our model using bayesian model averaging. BMA provides a framework to consider the output of multiple models while accounting for their uncertainty. Coding our binary response as either 1 or -1, we classify each observation according to the sign of the average response from each model, weighted by the posterior probability of the models. For parametric models, we can evaluate the posteriors by using their likelihood functions and a non-informative prior, while other approaches can be utilized for non-parametric models, for instance by considering the probability that a classification tree takes a specific terminal node value given the likeliness that it goes down this path. Given more time to familiarize ourselves with the theoretical and practical aspects of BMA, we could have used employed it to solve this problem instead of using the simpler albeit less sophisticated majority vote approach we favoured.

6.2 GAM Modelling

There are other paramaters to be tuned within the GAM model under the mgcv package. For instance the 'bs' value which indicates which spline smoothers are used and the gamma value which adjusts the smoothness of the model. Should there extra time, we would like to explore more of these options.

6.3 Factor Importance

We would like to examine the importance of every factor. It is likely that some of the variables are not important to each model, and could thus be removed to improve interpretation and reduce the dataset's dimension.

6.4 Neural Network

Neural Network, known as "universal function approximator" is a powerful technique that we definitely need to consider for our future work. Neural Network allows us to ingest high dimensional data and perform implicit complex computation within

each hidden units through iterative forward and backward propagation procedure. Each hidden unit in the neural network serves as a mini computer which specializes in capturing certain patterns and correlation between explanatory and response variables. Along with the feedback loop from the back propagation step, Neural Network is certainly one of the ideal candidates for our work. The only downside with this approach is the required computational resource coming with it, which is not in our favour given our limited computing resource. With neural networks, the number of parameters which we have to deal is much larger any other methods we have tried in this projects. Moreover, back propagation is all about calculating derivatives using chain rule and this is quite computational expensive and time consuming. With that said, our guess is that Neural Network is very likely to outperform most of the algorithms we used in this project.

7 Contribution

	Preprocessing	Modelling
Hieu Nguyen	1/3	1/3
Jacob Raymond	1/3	1/3
Dingding Hu	1/3	1/3

A The Bank Marketing Dataset

We downloaded the dataset we used for analysis from Kaggle[1]. The data was originally published by Sérgio Moro, Paulo Cortez, and Paulo Rita for their 2014 paper "A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems"[2]. It was collected by an undisclosed Portuguese bank conducting multiple marketing campaigns between May 2008 and June 2013. The sample size was 52,944 observations, which consisted in telephone contacts initiated by the bank. The dataset originally contained 150 features covering to gamut of variables, including metrics about the telemarketer's interaction, client demographics, the offered product, and macroeconomic variables such as the Portuguese unemployment rate and Eurobean interest rates. Moro et al. consolidated the factors into 22 variables through feature engineering. After the study's completion, the authors donated a subset covering 17 campaigns offering term deposits between May 2008 to November 2010 to the University of California Irvine's Machine Learning Repository. Finally, the dataset was uploaded on Kaggle by user Janio Martinez. He removed the six macroeconomic and societal factors and transformed some of the categorical variables: for example, he consolidated the *education* categories into a smaller number of more inclusive groups.

The variables in our dataset are presented in the following table.

[1]See https://www.kaggle.com/janiobachmann/bank-marketing-dataset

[2]S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014

A THE BANK MARKETING DATASET

	Category	Type of Data	Description
Age	Client	Numeric	The client's age
Job	Client	Categorical, 12 classes	The client's job, categorized in one of 12 categories including "retired" and "unemployed"
Marital	Client	Categorical, 3 classes	The client's marital status: "single", "married", or "divorced" (which also includes widowed clients)
Education	Client	Categorical, 4 classes	The client's education level: "primary", "secondary", "tertiary", or "unknown"
Default	Client	Boolean	"Yes" if the client has credit in default
Balance	Client	Numeric	Average yearly balance, expressed in euros
Housing	Client	Boolean	"Yes" if the client has a housing loan
Loan	Client	Boolean	"Yes" if the client has a personal loan
Contact	Campaign	Categorical, 3 classes	The type of telephone the client used during the last contact: "cellular", "telephone" (landline), or "unknown"
Day	Campaign	Numeric	The day of the month upon which the last contact occured
Month	Campaign	Categorical, 12 classes	The month in which the last contact occured
Duration	Campaign	Numeric	The duration of the last contact, in seconds
Campaign	Others	Numeric	Number of times this client was contacted during this campaign
PDays	Others	Numeric	Number of days between the last contact from a previous campaign and the current contact for this client (-1 indicates the client was not contacted in a previous campaign)
Previous	Others	Numeric	Number of times this client was contacted prior to this campaign.
POutcome	Others	Categorical, 4 classes	Outcome of the previous campaign targeting this client: "Failure", "Success", "Other", or "Unknown" (includes clients that were not previously contacted)
Deposit	Response	Boolean	"Yes" if the client subscribed to a term deposit

B Literature Review

As mentioned in appendix A, we obtained the Bank Marketing dataset from Kaggle. As a hub for data science enthusiasts, users are encouraged to share projects that they have done with the published data. We identified three such project that we deemed to be high quality: they had clear goals, they included documentation and a commented codes, they were classification problems, and they were recent (which we defined as a project created in the last nine months).

Kaggle user Kostas Voul[1] wanted, like us, to develop a model to determine whether or not a prospective client would open a term deposit account. He tried eight approaches: linear SVM, neural networks, gradient boosting, random forest, logistic regression, decision trees, and naive Bayes. He compared the models between one another using their training error, which is unadvisable in order to prevent overfitting. However, Kostas Voul obtained similar training errors to us for most methods. He decided to use gradient boosting even though it was only the third best model in terms of error because of its favourable ROC curve. It had an 16% training error (validation and test errors were not reported). Interestingly, Kostas Voul also used a decision tree classifier to determine feature importance. He found that the three most important variables that determine if a prospective client will subscribe to a term deposit are *duration* (which is the most important variable by far), *contact*, and *housing*.

User Eryk Walczak[2], meanwhile, explored applications of XGBoost, a gradient boosting algorithm, on the bank marketing dataset. He first used one hot encoding on the data in order to separate the categorical variables and the binary factors. He then considered two models: one where the algorithm incorporated all of the factors, and another where XGBoost utilized only the five most important explanatory variable as defined by the full model. These are, in order, *poutcomesuccess* (a boolean varaible to determine if the previous campaign was a success), *contactunknown* (a boolean variable to determine whether the client's phone type was known), *balance*, *age*, and *housingyes* (a boolean variable indicating whether or not a client had contracted a housing loan). Eryk Walczak ultimately discovered that the full model had the better performance, with a test error of 30% compared to 34% for the reduced model.

Finally, Kaggle user Golden[3] documented her process to discover a classification scheme for this data. She decided to fit a random forest model to the data without any veritable justification. After performing cross validation, she reported that her model achieved a Gini test error of 15.3% and an entropy error of 17.2%. Over all, this is comparable to our test error of 17.3%. Golden also completed a an extensive data exploration process compared to other Kaggle projects. Using plots, she came

[1] https://www.kaggle.com/kostasvoul/bank-marketing-campaign-opening-a-term-deposit/notebook

[2] https://www.kaggle.com/erykwalczak/eda-and-model-race-in-r-xgboost-etcsystem-settings

[3] https://www.kaggle.com/goldens/bank-marketing-eda-model-and-plotly2–Predictive-Analysis

to the following insights:

- *Age* is positively correlated with *deposit*, suggesting that older clients are more likely to accept the term deposit offer.
- Over 80% of clients in the "no" class of *deposit* had *pdays*=-1, meaning that they were not contacted during a previous campaign.
- *Duration* is positively correlated with *deposit*, meaning that clients who eventually accepted the term deposit had a longer conversation with the telemarketer than prospects who refused.
- *Poutcome* is positively correlated with *deposit*, implying that clients who had accepted a term deposit during a previous campaign were more likely to accept another.

While it is not entirely comparable, we may also want to consider the approach utilized by Moro et al. in the paper associated with this data.[4]. The authors wanted to identify a data mining scheme to predict the success of a bank's telemarketing campaign. As mentioned in Appendix A, the dataset they possessed was significantly more extensive than ours (150 factors, which they reduced to 22 explanatory variables, and five years' worth of observations). Moro et al. considered four models: logistic regression, decision trees, neural networks, and support vector machine. They fitted each model using the **rminer** package in R. The authors compared the models using area under the ROC curve (AUC); a score close to 1 will indicate that a model's high sensitivity is not offset by a comparatively large false positive rate. In this sense, neural networks is the best performing model (0.929), following by logistic regression, radial kernel SVM, and finally decision trees.

[4]S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014