FinalProject

STA 141 Final Project

Group 27: Sameerah Helal, Mason Del Rio, Kaiming Fu, Yulu Jin

Introduction

Background

Customer acquisition and retention are one of most important goals of any business. This is particularly true of banks, which rely on long-term connections, like loans or accounts, in order to stay in business. We are given a data set related to a marketing campaign performed by a Portuguese bank; including client details and social/economic context variables; with the variable of interest being campaign success, defined by whether or not a client subscribed to a long-term deposit with the bank.
Given this data, our primary question of interest is to identify the best model to predict whether or not the a client will subscribe to a long term deposit. We will attempt to find the most important individual features as well as the most useful combinations and linear combinations. We will then test different prediction models, tuning to get the best parameters, then comparing the models to identify the one best suited to predict our variable of interest.

Literature Review

We have done literature review about support vector machine(SVM) and outliers computation.
SVM was proposed by Vapnik and his fellows in the 1990s and was applied to analyze data used for classification and regression analysis. Given a set of training examples, a classification model that assigns examples into different categories is generated by mapping the input data into a higher dimensional space and constructing an optimal separating. hyper-
planehttps://link.springer.com/article/10.1023/A:1018628609742
(https://link.springer.com/article/10.1023/A:1018628609742)
Moreover, given a data set, outliers are data points that deviate from the rest of the samples, often to the point of skewing any models trained using them. There are many methods for computing outliers, including, but not limited to z-score, local outlier factor, one-class SVM, and isolation forest. For our data, we use isolation forest because it is suited for data with many, sometimes irrelevant attributes, and is robust despite needing few parameters.
https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/icdm08b.pdf?q=isolation-forest
(https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/icdm08b.pdf?q=isolation-forest)
Isolation forest is a nonparametric, unsupervised outlier detection algorithm that is principled upon outliers being few in number, and far from the rest of the data. As a method that uses binary trees, the idea of isolation forest is that anomalies would be more easily "isolated" into leaf nodes than the other data points. The measure of ease of isolation in this case is height, or path-length from the root. The algorithm randomly selects a feature and a split value within the min-max range, and as it compares observations to the split value

during prediction, “path lengths” is recorded. Those points that are isolated faster, outliers, will have shorter path lengths. For each observation, an outlier score in the unit range $[0, 1]$ is computed, correpsonding the the sample's "outlierness", with 1 being more like an outlier and 0 being less like an outlier.

Proposed methods

In this section, we perform logistic regression, random forest and SVM for classification on the "bank-additional-full.csv" dataset. Cross validation is also performed for each algorithm when it applies.

Logistic Regression and LASSO Regularization

In order to select which features matter the most, we will use Lasso Regularization on our Logistic Regression Model. #### Logistic Regression

The equation for Logistic Regression is :

$P = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$

Where P is our target variable which varies from 0 to 1, and the β_0 , β_1 , and X are the independent Variables.

Or similarly, $log(\frac{p_i}{1-p_i}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 \ldots \beta_k X_k$ Where $log(\frac{p_i}{1-p_i})$ is the Log Likelihood.

We first index the data into a Training Set and a Testing Set using the sample function.

As we can see, 70% of our data was split for indexing to our training and testing data.

28831/41188 = 0.69999854

We now have our data split up, randomly, most importantly, and we can fit our logistic model to the training data using the glm() function.

In order to see how accurate our model is, we need to set up a confusion matrix. We use the test data that was separate from the training model in order to simulate some form of real world implementation and feed it into the predict.glm() function.

The Logistic Regression model works fairly well for the test data, with 91.3% accuracy. This is a model that works well with the data that was randomized and set once, for training and testing purposes. However, we must use K-Fold Cross Validation in order to test the model's ability to predict outside data not used in the process of creating this model.

Logistic Regression with K-Fold Cross Validation

We will use the caret library to implement cross validation. Below, we will again create a training and testing set for the data. However, this time the K-Fold Cross Validation will randomly select portions of the training data 10 times.

After the cross validation is done, we need to see how well our improved model performs on test data. We use the predict() function to store the predictions generated by our model in predictions.

Now, we use the confusionMatrix() function from the caret package to generate our accuracy.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    no   yes
##        no  10696   833
##        yes   282   545
##
##                Accuracy : 0.9098
##                  95% CI : (0.9046, 0.9148)
##     No Information Rate : 0.8885
##     P-Value [Acc > NIR] : 5.554e-15
##
##                   Kappa : 0.4482
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9743
##             Specificity : 0.3955
##          Pos Pred Value : 0.9277
##          Neg Pred Value : 0.6590
##              Prevalence : 0.8885
##          Detection Rate : 0.8657
##    Detection Prevalence : 0.9331
##       Balanced Accuracy : 0.6849
##
##        'Positive' Class : no
##
```

As we can see, the Logisitic Regression Model with and without cross validation has ~90-91% accuracy, but we need to add some form of bias because this model is overfit to this certain dataset. If we wanted to generalize this data to other banks in Portugal, we would need to compensate for the overfittness of this model to this dataset.

This is where Lasso Regularization comes in.

Lasso Regularization

Lasso Regularization for Logistic Regression is done by minimizing this Cost Function:

$$J(\Theta) = \frac{1}{m}\Sigma_{i=1}^{m}[-y^i log(h_\Theta(x^i)) - (1-y^i)log(1-(h_\Theta(x^i))] + \frac{\lambda}{2m}\Sigma_{j=1}^{n}|\Theta_j|$$

First, we build a matrix for all the features of the dataset, which we'll call x, and a vector for the response variable, we'll call y, which consists of 1's and 0's for Yes and No.

In Lasso Regularization, there is a tuning paramater called Lambda in which we need to find the minimal value of, through cross validation. The minimum lambda will give us the most parsimonious model, but it will also generate a model which generally overfits. So we will pick the lambda which falls one standard error away from the minimum value in order to choose the simplest model without overfitting.

From this graph, we see that the lambda value between -8 and -7 will give us the most minimal model, but one standard error away from that value will give us the lambda value near -5, which will give us the minimal model with the least overfitting.

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                            1
## (Intercept)    48.676286978
## (Intercept)     .
## age             .
## job             .
## marital         .
## education       0.006273558
## default        -0.155829619
## housing         .
## loan            .
## contact         0.243650417
## month           0.058329218
## day_of_week     .
## duration        0.004044978
## campaign        .
## pdays          -0.001476937
## previous       -0.006067491
## poutcome        .
## emp.var.rate   -0.102596398
## cons.price.idx  .
## cons.conf.idx   0.020526121
## euribor3m       .
## nr.employed    -0.009820448
```

From the list of coefficients for the lasso model, we see that it selected 12 features that were the most significant on the model.

We now use our test data to see how accurate our model is.

```
##      true
## pred     no   yes
##   no  10738   902
##   yes   240   476
```

```
## [1] 0.9075753
```

As we see, the accuracy of this model has decreased because we introduced some bias with the Lasso Regularization method. In order for this model to be applied to other data sets, we will sacrifice accuracy for the sake of not overfitting the model.

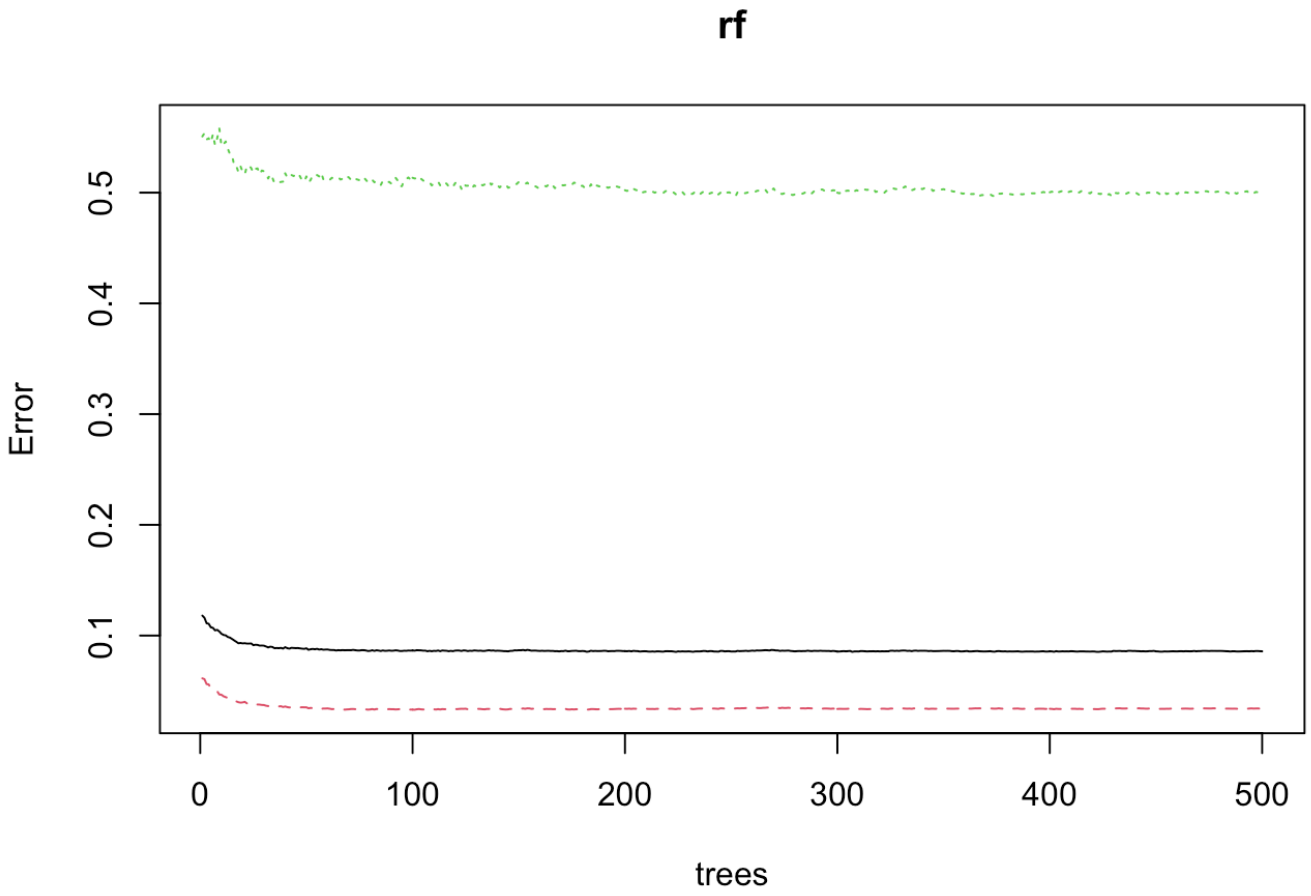
Random Forest

Another method we can use on this data is the Random Forest Method.

Using Random Forest gives us a vast amount of decision trees in which a model is made from the culmination of them. Using the ranodmForest library package, we can create the model in R. We first subset our data just like we previously did for the Logistic Regression model, with 70% of the data lumped into a Training Set, and 30% of our data lumped into the Test set.

We then use the function "randomForest()" and train it with the previously created training set, which will create our random forest model to use for predictions.

After this is made, we can plot to see how many trees are necessary for our model



This graph shows us that around 100-200 trees is where our model has minimum error

Now, let's confirm the variables of importance for the dataset using the varImpPlot() function

As we can see, duration ranks number 1 most important variable in this dataset, which from eyeballing the data, we see the strong relationship between a phone calls duration and the result of a term deposit being made. This graph confirms this notion.

Now, let's see how accurate this model is after feeding the model our test data. We will use the caret package to create the confusion matrix.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 10577   710
##          1   346   724
##
##                Accuracy : 0.9145
##                  95% CI : (0.9095, 0.9194)
##     No Information Rate : 0.884
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5318
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9683
##             Specificity : 0.5049
##          Pos Pred Value : 0.9371
##          Neg Pred Value : 0.6766
##              Prevalence : 0.8840
##          Detection Rate : 0.8560
##    Detection Prevalence : 0.9134
##       Balanced Accuracy : 0.7366
##
##        'Positive' Class : 0
##
```

From this confusion matrix, we can see that the accuracy was at 91.45%, and the Kappa value was at .53, which makes for a moderate model. This model gives us around the same accuracy as the logistic regression model.

Support Vector Machine(SVM)

As we know, linear regression explores the linear relationship between predictors and the corresponding variable(y here). However, for the data collected in the real world, the relationship between predictors and the response is always quite complex and non-linear. The bank data for this project contains 20 variables as predictors and it's trivial that the relationship between those variables and y is non-linear and hard to explore.

Thus, supporting vector machines is proposed to characterize such a complex non-linear relationship. To solve this problem, the input data is mapped into a higher dimensional space. By applying Kernel functions, a nonlinear problem in the lower dimensional space has been transferred into a linear one and thus an optimal separating hyper-plane can be learned. Actually, the hyper-plane is a boundary plane to perform the classification task.

SVM full model

We use all the 20 variables to predict the response y by the SVM algorithm. Since the SVM model can only deal with numerical data, first we convert all characteristic variables into numerical ones. Then we seperate the data into training and testing set and the training set accounts for 70%.

Since there is randomness in the training set generation part and also in the SVM training part, we apply set.seed() to fix the result. After training, the prediction accuracy on the test set is 90.928%.

```
##
## Call:
## svm(formula = y ~ age + job + marital + education + housing + loan +
##     contact + month + day_of_week + duration + campaign + pdays +
##     previous + poutcome + emp.var.rate + cons.price.idx + cons.conf.idx +
##     euribor3m + nr.employed + default, data = training, importance = T,
##     type = "C-classification")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  5987
```

```
##       predict.SVM..test...1.20...type....response..
## test.y     0     1
##      0 10725   254
##      1   867   511
```

```
## [1] 0.9092822
```

SVM by Variable Category

In the context of the business problem of predicting the success of telemarketing in getting a client to sign on to a long-term deposit based on a limited number of variables, we may encounter the case where the bank is not able to collect or access the full range of variables. In addition to feature selection using other methods, we may also attempt to predict y based on different categories of information the bank may have. We consider life information (age, job, marital status, and education level), bank information (whether or not the client has a housing loan, personal loan, or credit in default), last contact information (number of times contacted during this campaign, number of days between the most recent two campaigns, number of times previously contacted, and outcome of the previous campaign for a particular client), and social and economic context at the time (employment variation rate, consumer price index, euribor 3 month rate, and number of employees).
We first load and encode the data and then train four SVM models by separating the variables by category.

We test our models using the test set containing 30% of the data, which we feed into the model and use the results to compute accuracy.

```
## [1] 0.888727
```

```
## [1] 0.8884843
```

```
## [1] 0.896415
```

```
## [1] 0.8901837
```

Our results are that life information and bank information have approximately 88.8% accuracy, while last contact information has 89.6% and socio-economic context has 89.0%. So if the bank were able to use only one of these categories, it would have the best, if only marginally better, accuracy by using last contact information as their predictor. This is consistent with our previous findings that duration of phone call had the highest relative importance as a predictor, since the highest accuracy among the categories belongs to that of last contact information.

Cross Validation for SVM

To get the best possible parameters for SVM, we tune the model. To accommodate the function requirements, we use a different encoding in one aspect: we encode the output y as factor levels instead of numeric binary or continuous. Our data is appropriately formatted for our needs, and we may perform cross validation. In this case, we use k-fold cross validation with $k = 10$.

The result of performing 10-fold cross validation for SVM on this data set is a tuned model that has an accuracy of 90.92% when tested on the test set. This is not an increase from our model that did not include cross validation. However, this may be accounted for by a possible decrease in over fitting, meaning that, while the accuracy for this particular test set is not better, the cross-validated model will generalize better.

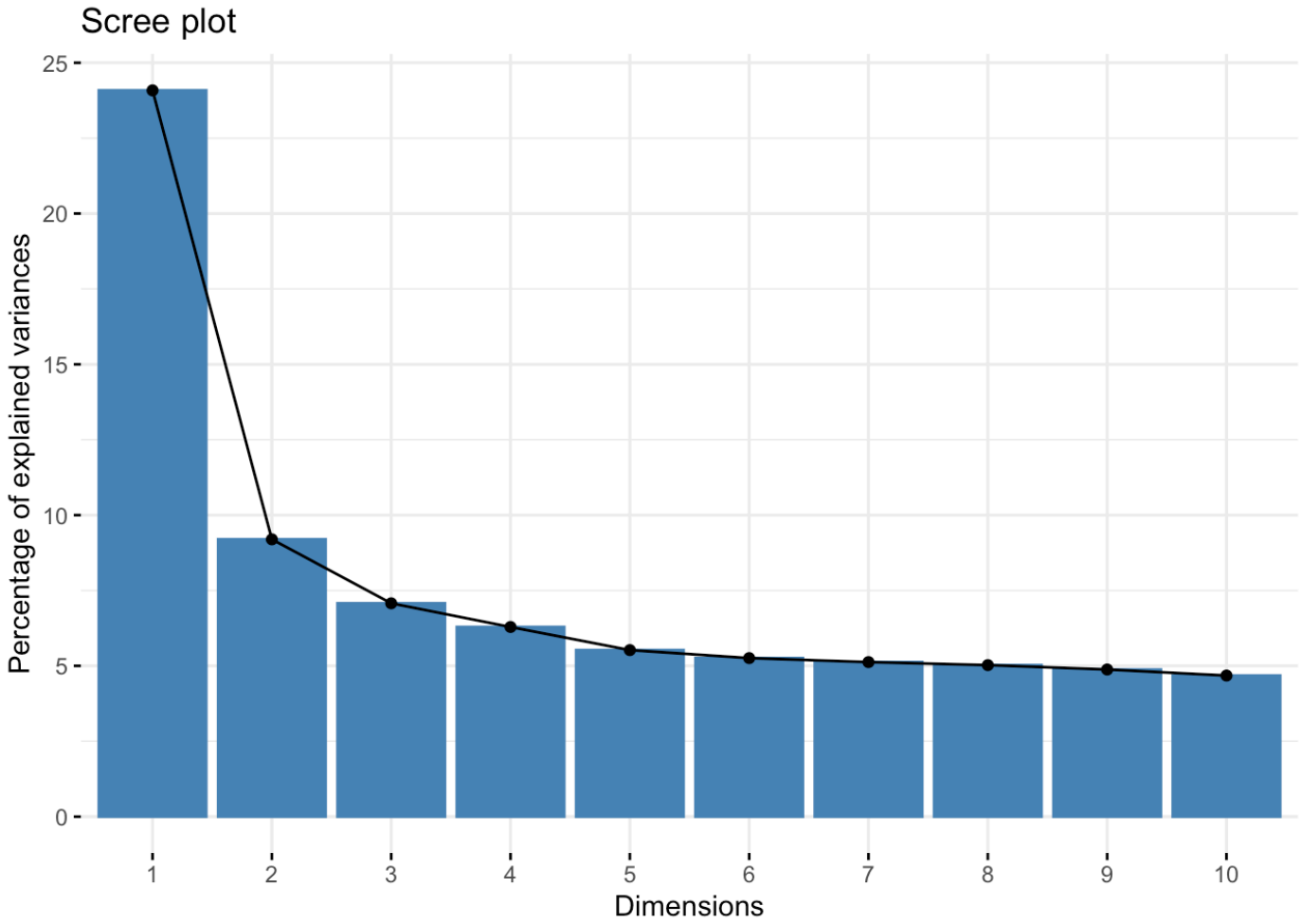
Principal component analysis(PCA)

Principal component analysis(PCA) is widely used for dimensionality reduction. PCA computes the principal components and then the data points can be projected onto the first few principal components to obtain lower-dimensional data while preserving as much of the data's variation as possible. Moreover, it also eexplores the data and tells us the most important variables in the dataset.

We perform PCA on the dataset. By the summary result listed below, we have that the first 13 principal components give a cumulative proportion of 0.89615, which well represent the data.

```
## Importance of components:
##                            PC1     PC2     PC3    PC4     PC5     PC6     PC7
## Standard deviation     2.1948 1.35624 1.18958 1.1216 1.05106 1.02550 1.01252
## Proportion of Variance 0.2409 0.09197 0.07076 0.0629 0.05524 0.05258 0.05126
## Cumulative Proportion  0.2409 0.33282 0.40358 0.4665 0.52172 0.57430 0.62556
##                            PC8    PC9   PC10   PC11    PC12    PC13    PC14
## Standard deviation     1.00255 0.9879 0.9674 0.9497 0.91067 0.87377 0.82992
## Proportion of Variance 0.05026 0.0488 0.0468 0.0451 0.04147 0.03817 0.03444
## Cumulative Proportion  0.67581 0.7246 0.7714 0.8165 0.85797 0.89615 0.93059
##                           PC15    PC16    PC17    PC18    PC19    PC20
## Standard deviation     0.77229 0.63194 0.52046 0.30219 0.14289 0.09942
## Proportion of Variance 0.02982 0.01997 0.01354 0.00457 0.00102 0.00049
## Cumulative Proportion  0.96041 0.98038 0.99392 0.99848 0.99951 1.00000
```

The screen plot of principal components is shown below. The y-axis explains the percentage of explained variance for each principal component and we see that the first 8 principal components have a percentage over 5%.



By calculating the eigenvalue corresponding to each principal component, we have the following result. We would eliminate any variable that didn't have an eigenvalue greater than 1. The idea behind this is that if the eigenvalue is less than 1, then the component accounts for less variance than a single variable contributed. Thus, we view the first 8 principal components as important ones.

```
##  [1] 4.817087520 1.839383411 1.415111190 1.258025265 1.104729596 1.051643444
##  [7] 1.025195603 1.005114510 0.976005532 0.935940119 0.901938282 0.829320176
## [13] 0.763474234 0.688765080 0.596425595 0.399346889 0.270875507 0.091316810
## [19] 0.020416860 0.009884378
```

By summarizing the first 8 principal components, we see that several variables are less significant than others, such as "contact", "day_of_week", "month", since they contribute little in the first 8 principal components. We then drop those variables and train the SVM model based on the same training and testing set. The test accuracy is then 90.807%. Thus, there is no big difference between this model and the previous one trained by all the variables, which indicates that the dropped variables do not have much effect in the prediction process.

Isolation Forest

Using isolation forest, we compute the number of outliers in the data set. Since the output is not a logical value, we use a threshold of 0.5, which corresponds to average outlierness, to identify outliers. If a sample has greater than 0.5, or greater than average outlierness, then we mark it as an outlier.

We find that, in our data set with over 40,000 entries, we have 748 outliers, or that 1.82% of the data has greater than average likelihood of being an outlier.

Results

In this project, we perform logistic regression, random forest, SVM and their generalized models on the bank data. The results of different models are shown in the table. Therefore, we see that both logistic regression and random forest have better prediction accuracy on the test set. However, there is no signifcant difference in the performance of different models and all considered methods have acceptable prediction accuracy.

```
##          [,1]                 [,2]            [,3]    [,4]
## Model    "Logistic regression" "Random forest" "SVM"   "SVM with PCA"
## Accuracy "0.913"              "0.913"         "0.909" "0.908"
```

Extra Credit

Other than logistic regression and random forest, we also run SVM to do the classification. For logistic regression and SVM, we perform cross validation. Moreover, for logistic regression, lasson regularization is applied to construct a sparse model. To analyze the outlier, we perform the isolation forest algorithm.