Predict Client Subscriptions to Bank Marketing Campaigns

EE 660 Course Project

Nisha Jacob, nejacob@usc.edu   Muskaan Parmar, muskaanp@usc.edu

December 8, 2022

1 Abstract

The increasingly vast number of marketing campaigns over time has reduced its effect on the general public. We propose a semi-supervised learning approach to predict the success of telemarketing calls for selling bank long-term deposits. A Portuguese retail bank was addressed, with data collected from 2008 to 2013, thereby including the effects of the financial crisis at the time. We have implemented three main semi-supervised algorithms, namely, the propagating Nearest Neighbour algorithm, Expectation Maximization and Semi-supervised Support Vector Machine. To compare the effect of how much more or less a model can learn with more labeled data, five supervised learning models have also been built, which are, Logistic Regression, Random Forest, Support Vector Machine, Multi Layer Perceptron and Decision Tree Classifiers. We make use of features such as client age, marital status, education status, job along with information like the month in which they were contacted as well as the number of times they have been contacted in the current campaign. Since the dataset is highly imbalanced, we avoid using accuracy as a metric to determine if the model is effective. A model's success is determined by its sensitivity and specificity scores. Our best performing model is the Decision Tree Classifier (for SL) and the Label Propagation Algorithm (for SSL) with a sensitivity score of 0.95, 0.88 and a specificity score of 0.62, 0.62 respectively.

2 Introduction

2.1 Problem Type, Statement and Goals

In this project, we present five supervised systems and three semi-supervised systems for the prediction of client subscription to bank marketing campaigns. This is a classification problem since we predict whether a certain client subscribes ('yes') or does not subscribe ('no') to the campaign. This problem statement appeared interesting to us since bank marketing calls are something a bulk of people face regularly and mostly ignore. It would be beneficial to banks if there was a system in place to identify the categories of customers most likely to subscribe to their campaigns. The topic is not trivial since the vast majority of customers do not subscribe (88%), leading to a highly imbalanced dataset. We use the dataset provided by UCI (Moro et al., 2014) [1]. It compiles data related to the direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. The dataset contains a high number of categorical features, thereby adding to the complexity in feature processing. Encoding the categorical data with 1's and 0's would lead to a sparse data matrix from which the model may not be able to learn information.

We also aim to predict client responses without the true label specified (SSL), for a fraction of the data points and study how the model performance varies with the predicted labels. This, again, is a classification problem where we first need to predict labels of the unlabeled points in the train dataset, prior to predicting labels of unseen data. It is not always easy to acquire labeled datapoints since labeling can be a challenging and expensive task. If we can get a model to be able to predict the labels, the process of data collection can be made much easier. Since our dataset is already highly imbalanced, eliminating labels for a major chunk of the data would make it even more hard for the model to accurately predict customer subscriptions. This imbalance makes the problem non-trivial for SSL, since most SSL algorithms do not take class weights into account, leading to decision boundaries classifying all the datapoints as a single class.

2.2 Literature Review

In the paper by S. Moro, P. Cortez and P. Rita, they discuss a data mining approach using the R language to predict whether a client subscribed to long term deposits or not [2]. They compare four different classification algorithms like Logistic Regression, Decision Tree,Support Vector Machine (SVM) and Neural Networks(NN) using classification metrics like Area Under Curve(AUC) and Area of the LIFT cumulative curve (ALIFT). For all the algorithms, a rolling window feature is used to perform model updates and discarding the old data. Before implementing Neural Networks and SVM, they perform standardization on the numerical features. CART algorithm is used for decision trees and sequential minimal optimization (SMO) is used for training the SVM model. Hyperparameter tuning was only performed for NN and SVM models. The paper talks about heavy computation needs to run a model like SVM as compared to any other model. The results showed that Neural Network gave the best AUC and ALIFT scores as compared to other models.

The work by S. Moro, R. Laureano and P. Cortez discusses about using a data mining technique based on CRoss-Industry Standard Process for Data Mining (CRISP-DM) methodology [3]. They try out three different models: NB, DT and SVM with Gaussian kernel. From data analysis, they realized that the collection of data related to clients like time of contact, duration of contact, type of contact, etc. played an important role in deciding whether the client subscribed or not. Also, they found that call duration played an important role in the client subscribing. They found that the banks saw most success in the last months of every trimester. Classification metrics like ROC, AUC and cumulative Lift curve area were used to compare the performance of models. The results showed that SVM performed the best. Sensitivity analysis on the SVM model showed the input importance for the model.

2.3 Our Prior and Related Work (Mandatory)

Prior and Related Work - None

2.4 Overview of Our Approach

2.4.1 Main topic

The baseline systems are:

1. **The trivial baseline**: This model outputs class labels 0 and 1 with a probability N_1/N and N_2/N respectively, where N_i is the population of datapoints with class label S_i and N is the total population of datapoints. This is all based on the traning set.

2. **The non-trivial baseline**: The nearest means classifier is used as a non-trivial baseline model. It assigns datapoints the class label whose mean (derived from the training samples) is closest to the point.

The models used for the SL task are:

1. **Logistic Regression**: This statistical model estimates the likelihood of an event occurring based on a given dataset of independant variables. Since the outcome is a probability, the dependant variable is bounded between 0 and 1.

2. **Support Vector Machine**: This algorithm tries to find a hyperplane in an N-dimensional space that has the maximum margin (greatest separation for datapoints of both classes) and distinctly classifies the datapoints.

3. **Decision Tree**: Decision Trees categorize instances by organizing them in a tree from the root to a leaf node, which provides the classification of the instance. An instance is categorized by checking the attribute specified by the tree's root node, and going along the branch of the tree that corresponds to the attribute's value.

4. **MultiLayer Perceptron**: This model consists of three types of layers - the input layer, output layer and hidden layer. The true computational engine of an MLP lies in the hidden layers placed between the input and output layer. MLPs can approximate any continuous function and can solve problems that are not linearly separable.

5. **Random Forest**: This model consists of a large number of individual decision trees that operate as an ensemble. Every tree in the random forest provides a class forecast and the classification that receives the most votes becomes the model's prediction.

   Due to the dataset being imbalanced, model performances are not compared using the usual accuracy score metric. We instead use sensitivity and specificity to be able to identify how accurately our model can predict true negatives and true positives. A higher score implies better model performance. We use a probability cut-off point to classify observations, to understand the most optimal trade off between specificity and sensitivity. This also helps take into account the fact that the classes are imbalanced in weights and balances the metrics accordingly.

$$Specificity = \frac{TN}{TN + FP} \tag{1}$$

$$Sensitivity = \frac{TN}{TN + FN} \tag{2}$$

2.4.2 Extension (SSL)

The baseline system is:

1. **Propagating Nearest Neighbour**: This is a type of Self Learning model that moves data from the unlabeled set to the labeled set based on the confidence of predictions. This process is iterated until there are no unlabeled points left.

The models used for the SSL task are:

1. **Label Propagation**: This algorithm helps find communities in a graph. It propagates labels throughout the network and forms communities based on this label propagation process, thereby using network structure alone as a guide.

2. **Expectation Maximization**: This algorithm performs maximum likelihood estimation in the presence of latent variables. It first estimates the latent variable values and then optimizes the model, repeating the steps till convergence is achieved.

3. **Semi Supervised Support Vector Machine**: This algorithm constructs a support vector machine using both labeled and unlabeled training data and tries to maximize the margin width for separation between classes.

For the SSL models as well, performance is compared using the sensitivity and specificity metrics on the same test data as SL, in order to verify which algorithm helps better in our case. At the end, we also do a 1:1 comparison between SL and SSL Logistic Regression models, wherein we train the SL model with the same amount of labeled data as the SSL model to identify which model learns better.

3 Implementation

3.1 Data Set

We have used a dataset that contains information related to direct marketing campaigns of a Portuguese banking institution. The marketing campaigns are based on phone calls. Often, more than one contact to the same client is required in order to determine if they would subscribe to the campaign or not. We have to predict whether the client would subscribe to the deposit or not, making it a binary classification problem.

The dataset contains 41,188 datapoints for the months from May 2008 to November 2010. The dataset is significantly imbalanced, with only 12% (4,943) datapoints belonging to the minority class (client subscription to campaign). There are 20 features in total, of which 10 are numerical and the remaining are categorical.

Numerical Feature	Datatype	Description
age	Integer	Age of client
duration	Integer	Duration of the last contact (in seconds)
campaign	Integer	Number of contacts performed during the current campaign for this client
pdays	Integer	Number of days that passed by after the client was last contacted from a previous campaign
previous	Integer	Number of contacts performed before this campaign and for this client
emp.var.rate	Float	Employment variation rate (measured quarterly)
cons.price.idx	Float	Consumer Price Index (measured monthly)
cons.conf.idx	Float	Consumer Confidence Index (measured monthly)
euribor3m	Float	Euribor 3 month rate (measured daily)
nr.employed	Float	Total number of employees (measured quarterly)

Table 1: Description of Numerical Features

3.2 Dataset Methodology

The dataset is initially divided into a train and test set with an 80:20 ratio. The same train and test sets are used for both SSL and SL to maintain uniformity of results. This leads to a training set consisting of 32,951 points and a test set containing 8,237 points. The split is performed in such a way so as to keep the ratio of samples from both classes the same across train and test data.

While training each model, stratified Kfold cross validation is used to divide the dataset into a train set and validation set. We implement cross validation with k = 10 and repeat the validation process 3 times, finally taking an average of the best scores across all runs to provide a less noisy estimate.

The validation set is separated from the train set at the beginning of the train loop, before the model is used. We then proceed to train the model on the training data and post training, its performance is evaluated on the validation set. This process makes it seem like the model is trying to predict on unknown data. The sensitivity score obtained by testing on the validation set is stored and the same process is evaluated on the next fold by initializing all model parameters again. Finally, the model that gives the best sensitivity across all folds is chosen as the best model. This model is finally trained on the original training dataset (train and validation) and the trained model is ultimately used to predict labels for the unknown test set.

The test set is not used anywhere in the process flow until at the very end when the model is required to predict on unknowns. Hence, the test set does not exert undue influence on the model selection process. No decisions are made using the test set and it is used once during the workflow.

In Figure 2, we can see the step by step process of how this project was implemented.

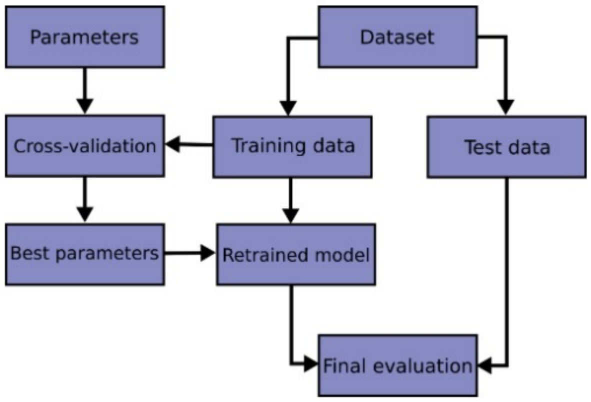


Figure 1: Cross Validation workflow

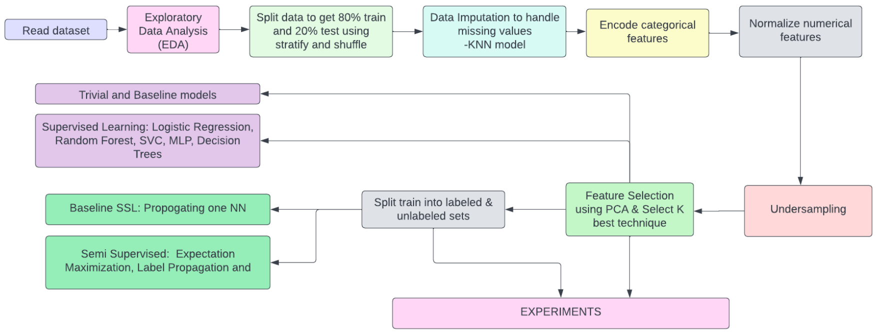

Read dataset
Exploratory Data Analysis (EDA)
Split data to get 80% train and 20% test using stratify and shuffle
Data Imputation to handle missing values -KNN model
Encode categorical features
Normalize numerical features
Trivial and Baseline models
Supervised Learning: Logistic Regression, Random Forest, SVC, MLP, Decision Trees
Baseline SSL: Propogating one NN
Split train into labeled & unlabeled sets
Feature Selection using PCA & Select K best technique
Undersampling
Semi Supervised: Expectation Maximization, Label Propagation and
EXPERIMENTS

Figure 2: Flow chart showing the different steps implemented

Categorical Feature	Description	Categories	Cardinality
job	Type of job	'admin', 'blue-collar', 'entrepreneur', 'housemaid', 'management', 'retired', 'self-employed', 'services', 'student', 'technician', 'unemployed', 'unknown'	12
marital	Marital status	'divorced', 'married', 'single', 'unknown'	4
education	Education completed by client	'basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'illiterate', 'professional.course', 'university.degree', 'unknown'	8
default	Whether client has credit in default or not	'no','yes','unknown'	3
housing	Whether the client has housing loan or not	'no','yes','unknown'	3
loan	Whether the client has a personal loan	'no','yes','unknown'	3
contact	Contact communication type	'cellular', 'telephone'	2
month	Month of last contact	'apr', 'may', 'jun', 'jul', 'aug', 'sep', 'oct', 'nov', 'dec'	9
day of week	Day of last contact	'mon','tue','wed','thu','fri'	5
poutcome	Outcome of the previous marketing campaign	'failure', 'nonexistent', 'success'	3

Table 2: Description of Categorical Features

3.3 Preprocessing, Feature Extraction, Dimensionality Adjustment

We make use of the bank-additional-full.csv file which comprises of 41188 entries and 20 features. The dimensionality of the dataset is 20. We see that the dataset has a high number of datapoints (constraints) and a comparatively low number of features (degrees - of - freedom). The rule of thumb states that if number of constraints lie within (3-10) of the d.o.f, it is a good model, Since our dataset exceeds this limit, we would need to take steps to ensure that the model does not overfit.

The target label y for this dataset is a binary categorical output taking values 'yes' or 'no'.

The dataset is highly imbalanced in nature as it contains only 12% of data points for the class label 'yes'. Due to the imbalanced nature of dataset, we will be making use of classification performance metrics like Specificity and Sensitivity which would help us understand how well our models work in predicting a class correctly.

We drop the feature 'duration' as the original repository states that this feature should only be used for benchmark purposes and discarded in case one wants to build a prediction model. To get a better understanding of the distribution of the dataset, we perform Exploratory Data Analysis (EDA). We begin with generating a histogram plot for age showing the count of clients for different age groups when clients subscribed vs when they didn't. The plot in Figure 3 shows that there are 3 broad age categories that influence the way clients subscribe to a term deposit. The categories are as follows:

(i) Below 30 years old (Young): Between the age of 20 and 30 clients begin to take subscriptions.

(ii) 30 years old to 60 years old (Middle aged): More unwilling to take deposits than other categories. A lot of clients take subscriptions around the age of 30.

(iii) Above 60 years old (Retired):Most of the retired clients end up taking subscriptions

In Figure 4, the count_yes and count_no columns show the number of customers for a particular job, who subscribed and did not subscribe respectively. The prob_no and prob_yes columns show the probability of distribution for both count_no and count_yes w.r.t. count_total for a particular job.

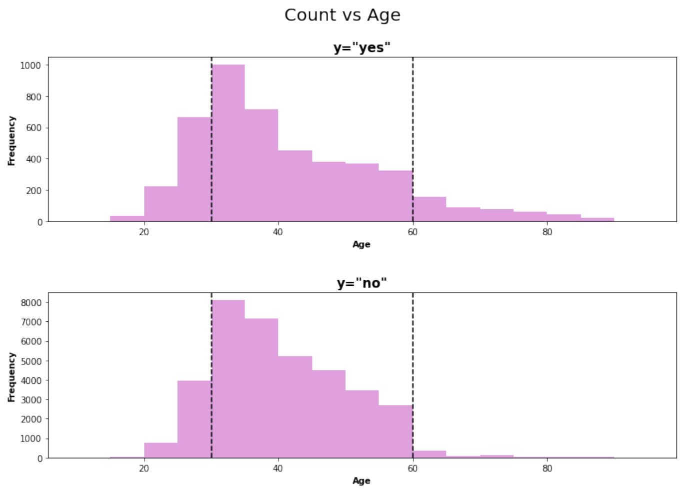
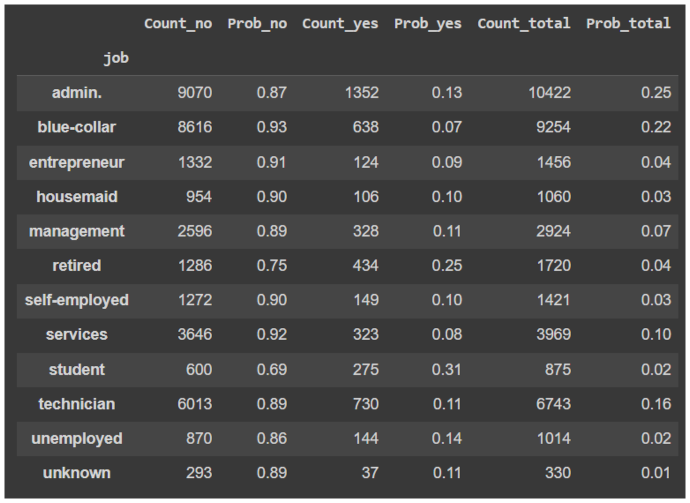


Figure 3: Histogram plot for age showing the distribution of subscribed clients vs non subscribed clients



job	Count_no	Prob_no	Count_yes	Prob_yes	Count_total	Prob_total
admin.	9070	0.87	1352	0.13	10422	0.25
blue-collar	8616	0.93	638	0.07	9254	0.22
entrepreneur	1332	0.91	124	0.09	1456	0.04
housemaid	954	0.90	106	0.10	1060	0.03
management	2596	0.89	328	0.11	2924	0.07
retired	1286	0.75	434	0.25	1720	0.04
self-employed	1272	0.90	149	0.10	1421	0.03
services	3646	0.92	323	0.08	3969	0.10
student	600	0.69	275	0.31	875	0.02
technician	6013	0.89	730	0.11	6743	0.16
unemployed	870	0.86	144	0.14	1014	0.02
unknown	293	0.89	37	0.11	330	0.01

Figure 4: Group wise distribution of clients for each job

We can see that a majority of clients with the job of admin end up subscribing to a term deposit. In the case of jobs like blue-collar, entrepreneur and services, very few people end up subscribing. A lot of students are observed to end up subscribing. Similarly, we use the group wise distribution table to come to results:

- For marital, it is observed that single people subscribe to a term deposit more often than married or divorced.
- For default column, there are only 3 for default='yes' so we can drop this feature as the remaining group is 'no' and there is not much information about the minority group 'yes'.
- From contact column, we get to know that people with contact as cellular subscribed more as compared to the people who had telephone as their contact source.
- For pdays, 999 indicates client was not previously contacted, so we convert all the pdays less than 999 as 1 and rest as 0.
- Since previous indicates number of contacts performed before the current campaign and for the current client, we set all previous contacts of more than 2, to 2.Such clients who were contacted more than twice in the previous campaign are more likely to subscribe to a term deposit.
- poutcome indicates success or failure in the previous campaign, it tells us that success in the previous campaign means these clients are likely to subscribe again.

We can see that in Figure 6, the group illiterate has very few points so we drop off the category 'illiterate' from the dataset.

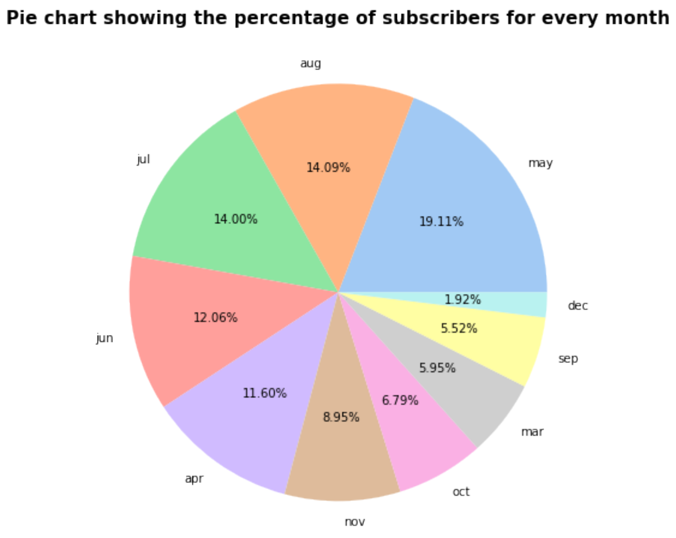


Figure 5: Percentage of subscribers for every month

The pie chart in Figure 5 depicts that in the month of May most clients had subscribed. This was closely followed by the month of July and August. It shows that contacting clients in these months would be profitable.

For categorical features, we make use of chi square and p values to understand the importance of these features. A p-value of 0.05 or lower indicates that the feature is statistically significant to validate a hypothesis.

For loan, for housing, and for day_of_week, the p value seems to be statistically insignificant,thus we can drop this feature.

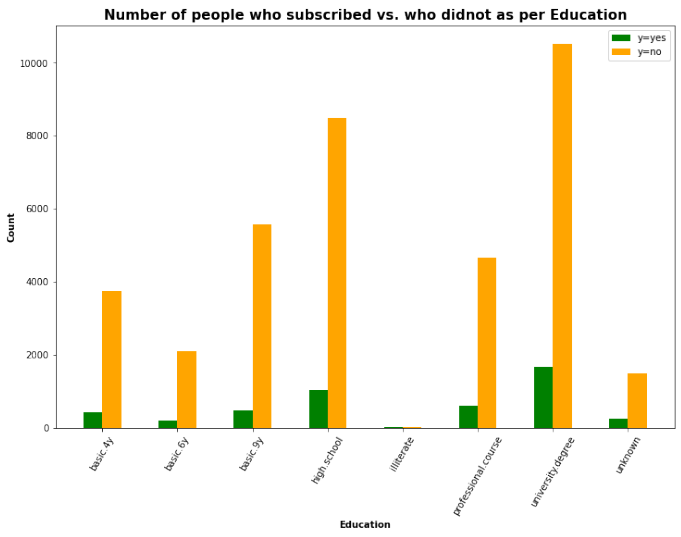
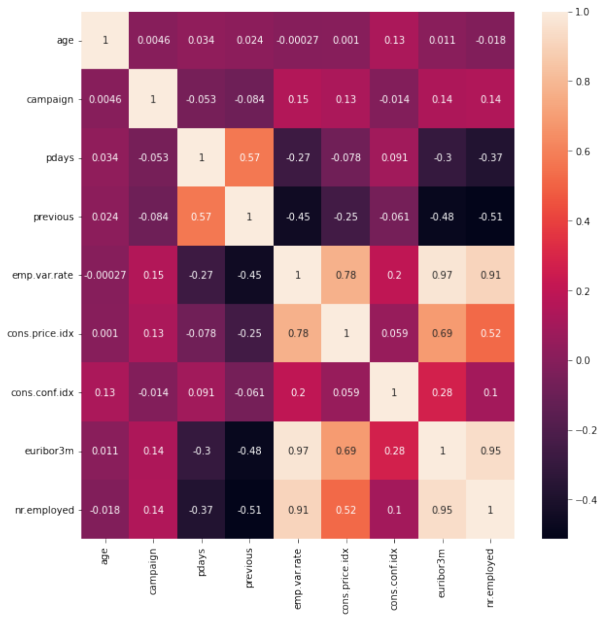


Figure 6: Count of people who subscribed vs. who didn't as per Education



	age	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
age	1	0.0046	0.034	0.024	-0.00027	0.001	0.13	0.011	-0.018
campaign	0.0046	1	-0.053	-0.084	0.15	0.13	-0.014	0.14	0.14
pdays	0.034	-0.053	1	0.57	-0.27	-0.078	0.091	-0.3	-0.37
previous	0.024	-0.084	0.57	1	-0.45	-0.25	-0.061	-0.48	-0.51
emp.var.rate	-0.00027	0.15	-0.27	-0.45	1	0.78	0.2	0.97	0.91
cons.price.idx	0.001	0.13	-0.078	-0.25	0.78	1	0.059	0.69	0.52
cons.conf.idx	0.13	-0.014	0.091	-0.061	0.2	0.059	1	0.28	0.1
euribor3m	0.011	0.14	-0.3	-0.48	0.97	0.69	0.28	1	0.95
nr.employed	-0.018	0.14	-0.37	-0.51	0.91	0.52	0.1	0.95	1

Figure 7: Correlation heatmap showing high correlation for euribor3m with emp.var.rate & nr.employed

The correlation heatmap in Figure 7 shows euribor3m having strong correlation with emp.var.rate and nr.employed. Thus, we will remove this feature from future analysis.

Split of data into train and test : Next, we split the dataset into 80% training and 20% test data. We will use this distribution for all supervised algorithms and for semi supervised algorithms too. The split is performed after shuffling data and the split is stratified i.e. the distribution of the data for the two classes is maintained in both the train and the test set. Else, it may happen that all the minority classes end up in test data or train data which would lead to incorrect results as the machine learning model doesn't get to learn the distribution of data for both the classes.

Handling Missing Data : Some of the categorical features of the dataset like marital, job, education and loan have a category 'unknown' present in them. We discard two data entries which have more than four column entries as category 'unknown' because we cannot predict a value for such cases.For the rest of the data, we decided to use a machine learning model to predict the category for the unknown labels for a particular feature. We perform the data imputation technique on the following features- marital, job, education and loan. We decided to make use of the data imputation technique as it is not always possible to discard data with missing values. We use a supervised learning technique- KNN where the numerical features like age, campaign, emp.var.rate, cons.price.idx, cons.conf.idx and nr.employed are used to predict class for the unknown category present in the target label. Ordinal encoding is performed on the target label prior to training data. The train data for this model is split into train and validation using shuffle and stratify technique to perform hyper parameter tuning for the optimal value of k in the range of 1 to 20. For the test set of the data, we use the trained model obtained for each feature to predict the value for the corresponding feature in test set. Table 1 shows the optimal value of k obtained for each of the four categorical features. The high error on validation set for the case of job and education is due to the fact that there are a lot categories present for each of these features. Once we have predicted categories for the unknown class present in each feature, the cardinality of these features also reduces by 1. Thus, the updated cardinality for marital, job, education and loan is 3, 11, 7 and 2 respectively.

Feature	**Minimum Error**	**optimal K**
marital	0.36	16
job	0.79	17
education	0.77	17
loan	0.15	15

Table 3: Results for Trivial Model

Categorical Feature Encoding : Different feature encoding techniques are used to encode the categorical variables. The months are encoded as an integer in the range from 1 to 12. For marital, job, education and outcome we use frequency encoding technique. Frequency encoding is a technique to assign the weight of each class in the feature in both direct & inverse proportion,depending on the nature of the data. We tried to use one hot encoding for these features but it generated a lot of features due to the high cardinality which led to curse of dimensionality and showed poor performance of the model on data. Thus, feature encoding ensures that the information is not lost and it doesn't lead to an increase in the dimensionality. The feature loan is converted to binary value by mapping yes to 1 and no to 0. Similarly, contact is converted by mapping telephone to 1 and cellular to 0. The target label y for both train and test is converted to binary value by mapping yes to 1 and no to 0 for easy calculation. We normalize all the numeric features for both train and test by using min max scaler to scale each feature to a range of 0 to 1.

We had worked on creating new features by merging data from numerical and categorical features to create new columns. We merged the features age and job to generate a new feature to get average age per job. We also created a new feature to find average campaigns per month. We had to discard these values as they did not have any correlation with the target label y.

Undersampling : Now, we perform under sampling on our highly imbalanced data by using the Repeated Edited Nearest Neighbour algorithm. This algorithm works by using k=3 nearest neighbour approach and removes points that do not 'agree' enough with their neighbourhood. After under sampling, the class 0 has 21042 data points and class 1 has 3709 data points. Due to the high number of points in the dataset, we cannot use techniques like SMOTE or oversampling as it would lead to an even further increase in the number of data points which would eventually lead to model overfitting.

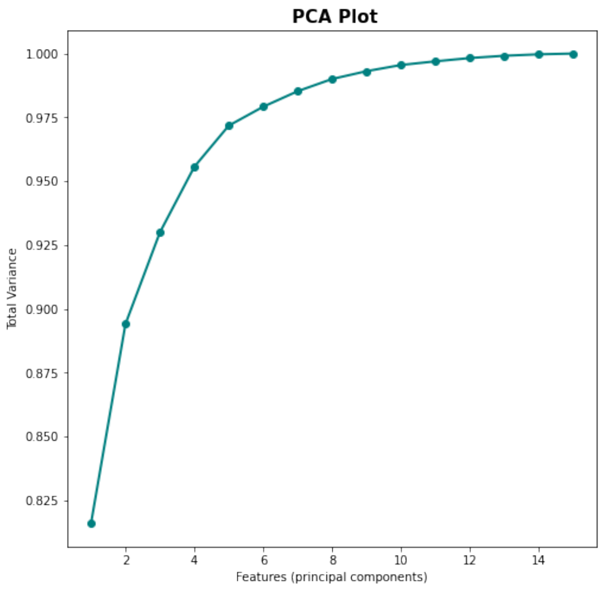


Figure 8: PCA Plot

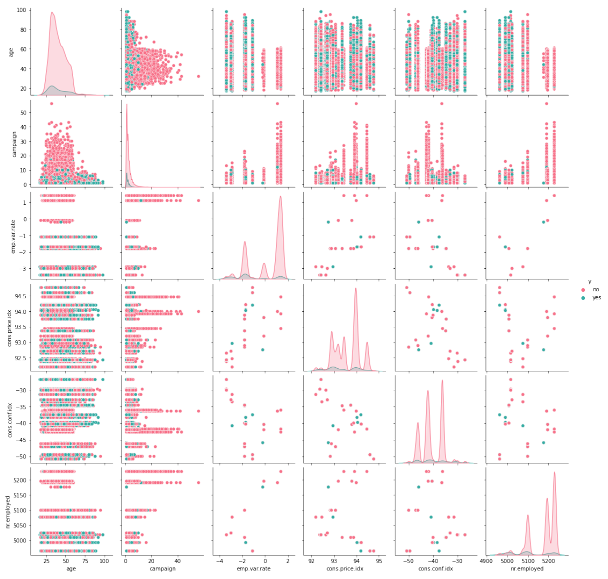


Figure 9: Pair plot of numerical features

Dimensionality Reduction : The PCA plot in Figure 8 shows the fifteen features that are present after the removal of a few of them during data pre-processing. PCA makes use of a new coordinate system and transforms the data in such a way that the largest variance of the data lies on the first coordinate axis and it follows the same method for the second largest variance and so on. A high variance indicates more information. From the plot, we can see that we only need 13 of the

15 principal components (features) to explain approximately 99% of the data. It means that when we reduce the number of features from 20 to 13 we are still able to retain 99% of information from the original data. Now, we use Select K best method to perform feature selection of 13 features out of the 15 features. The features age and loan get discarded by this method.

Predictive power of inputs: Table 4 shows the different predictive power scores for all the features in the dataset w.r.t. target label y.A high PPS score of, for instance, 0.80, would show that the label can be predicted well using the given feature.

Feature	**PPS**
age	0.001
duration	0.14
p_days	0.15
age	0.19
previous	0.16
emp.var.rate	0.12
cons.price.idx	0.15
cons.conf.idx	0.18
euribor3m	0.15
nr.employed	0.18
job	0.2
marital	0.18
education	0.16
default	0.16
housing	0.15
loan	0.003
contact	0.15
month	0.16
day_of_week	0.09
duration	0.25
poutcome	0.14

Table 4: Predictive Power of Inputs

3.4 Training Process

3.4.1 Trivial Model

This model outputs class labels 0 and 1 at random with probability N_1/N and N_2/N respectively, where N_i is the population of datapoints with class label S_i and N is the total population of datapoints. This is all based on the training set. This was chosen to be the trivial model since it doesn't really learn anything from the data and outputs labels at random.

This model does not have any learnable parameters since it does not learn anything hence, there is no hypothesis set.

Dataset	**Specificity**	**Sensitivity**
Train	0.15	0.85
Test	0.14	0.85

Table 5: Results for Trivial Model

3.4.2 Baseline Model

We use the nearest means classifier as the baseline model. It assigns datapoints the class label whose mean (derived from the training samples) is closest to the point. This model was chosen as the baseline since it is a simple classification algorithm that helps classify datapoints linearly in feature space. Its algorithm involves three simple steps:

- The centroid for each target class is computed while training.
- After training, given any point, say 'X', the distances (Euclidean) between the point X and each class' centroid is calculated.
- Out of all the calculated distances, the minimum distance is picked. The centroid to which the given point's distance is minimum, its class is assigned to the given point.

This model does not have any learnable parameters as well, resulting in a simple hypothesis set. The parameters of the model include the mean of the datapoints of each individual class, as calculated from the training data, as well as the distance metric used to calculate the separation between points.

Dataset	**Specificity**	**Sensitivity**
Train	0.7	0.58
Test	0.66	0.57

Table 6: Results for Baseline Model

3.4.3 Logistic Regression

Logistic regression is a process of modeling the probability of a discrete outcome given an input variable. Logistic model (or logit model) is a statistical model that models the probability of one event (out of two alternatives) occurring by having the log-odds (the logarithm of the odds) for the occasion be a linear combination of one or more independent variables ("predictors"). Logistic regression estimates the parameters of a logistic model.

$$Logistic function = \frac{1}{1+e^{-x}} \tag{3}$$

The loss function used here is related to "maximum likelihood estimation (MLE)" which is a conditional probability. Log Loss is the negative average of the log of corrected predicted probabilities for each instance.

$$LogLoss = -\frac{1}{N}\Sigma_{i=1}^{N} log(P_i) \tag{4}$$

We chose this model since it is simple to implement, interpret and is efficient to train. It does not make any assumption about the distribution of classes in feature space. The parameters chosen for the final trained model include C (regularization coefficient) = 10.0 and solver (algorithm used in the optimization problem) = $'lbfgs'$. These parameters have been chosen by model selection through determination of performance on the validation set.

The learnable parameters for the model include determining the probability that the datapoint belongs to class 1 and learning the decision function accordingly. The hypothesis set is a function of these learnable parameters. We have ∼25000 training datapoints and about 13 features. This exceeds the general rule of thumb where number of datapoints $N \sim (3-10) d.o.f.$. Hence to avoid overfitting, we prevent the model regularization coefficient from getting too high during hyperparameter tuning. We see that the performance on train and test set is almost the same. Since the minority class is significantly lesser in number, it appears that the model classifies a bunch of 1's as 0's leading to a low sensitivity score as compared to specificity.

Dataset	Specificity	Sensitivity
Train	0.88	0.62
Test	0.80	0.65

Table 7: Results for Logistic Regression

3.4.4 Support Vector Machine

The SVM algorithm aims to find a hyperplane in an N-dimensional space (N–number of data points) that distinctly classifies the datapoints. This hyperplane should have the maximum margin (maximum distance between datapoints of classes). This is so that future data points can be classified with confidence. It can work on sparse data, including under-determined systems. Support vectors are data points that lie closer to the hyperplane and have an influence on the hyperplane's position and orientation. The classifier margin is maximized using these support vectors. Lagrange optimization is used to minimize the loss function represented by the below equation

$$f(w) = \frac{1}{2}||w||^2 \tag{5}$$

The SVM kernel is a function that takes a low dimensional input space and transforms it to an expanded feature space. We make use of the RBF kernel that is defined as $K(x_i, x_j) = e^{-\gamma||x_i x_j||^2}$. This results in a more complex decision boundary.

This was chosen as one of the models to be implemented since it is effective in high dimensional spaces and we believed it would give us a good performance. The parameters chosen for the final trained model include C (regularization coefficient) = 1.0 These parameters have been chosen by model selection through determination of performance on the validation set.

The learnable parameters for the model include determining the weights of the individual features and learning the decision function accordingly. The hypothesis set is a function of these learnable parameters. We have ∼25000 training datapoints and about 13 features. This exceeds the general rule of thumb where number of datapoints $N \sim (3-10)d.o.f.$. Hence to avoid overfitting, we prevent the model regularization coefficient from getting too high during hyperparameter tuning. We see

Dataset	Specificity	Sensitivity
Train	0.98	0.49
Test	0.92	0.5

Table 8: Results for Support Vector Machine Classifier

that the performance on train and test set is almost the same. Since the minority class is significantly lesser in number, it appears that the model classifies a bunch of 1's as 0's leading to a low sensitivity score as compared to specificity.

3.4.5 Decision Tree

Decision Tree Classifier is a method of supervised learning by splitting the data into partitions iteratively and generating a tree-like structure for making decisions by using a set of rules. They are useful for solving cases where we have decision related problems.

A decision tree consists of nodes, edges and leaf nodes. The nodes represent the test feature. The edges are used to connect the node to another node/leaf. The root node contains the whole population of the dataset. The leaf nodes are the nodes that do not split further and are used to show the predicted outcome.

A measure called Gini impurity is used to decide the optimal split from the root node. It can perform binary splits and would help us in generating a decision tree.

$$Gini = 1 - \Sigma_{i=1}^{C}(p_i)^2 \tag{6}$$

where p_i stands for the probability.

A gini impurity value of zero means that the node is a pure node i.e. a leaf node. Usually a node's gini impurity is less than its parent node's gini impurity. A low gini impurity indicates a high homogeneity of the node.

We decided to use this model since it is very intuitive and easy to explain to both technical and non-technical stakeholders. The parameters chosen for the final trained model include max depth = 50.0 and min samples split = 20.0 These parameters have been chosen by model selection through determination of performance on the validation set.

The learnable parameters for the model include determining the weights of the individual features and learning the feature and its threshold that the node should be split on . The hypothesis set is a function of these learnable parameters. We have ∼25000 training datapoints and about 13 features. This exceeds the general rule of thumb where number of datapoints $N \sim (3 - 10)d.o.f.$. Hence to avoid overfitting, we prune the tree, preventing it from getting too low. We also increase the min sample split value to avoid the tree continuing to split into further nodes where there are only few samples present.

Dataset	**Specificity**	**Sensitivity**
Train	0.92	0.76
Test	0.80	0.66

Table 9: Results for Decision Tree Classifier

We see that the performance on the test set has reduced as compared to the train set. Since the minority class is significantly lesser in number, it appears that the model classifies a bunch of 1's as 0's leading to a low sensitivity score as compared to specificity.

3.4.6 Multilayer Perceptron

A Multilayer Perceptron has input and output layers, and one or more hidden layers with many neurons stacked together. These neurons can use any activation functions to propagate from one layer to the next. Typical choices for the activation include the tanh function specified by $tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$ or the logistic sigmoid function specified by $sigmoid(a) = \frac{1}{1+e^{-a}}$.

Multilayer Perceptron falls under the category of feedforward algorithms, because inputs are combined with the initial weights in a weighted sum and subjected to the activation function before being propagated to the next layer. It uses backpropagation to iteratively adjust the weights in the network, with the goal of minimizing the cost function.

This model was chosen because it can solve complex nonlinear problems and can handle large amounts of input data well. The parameters chosen for the final trained model include activation = *ReLU* and alpha (L2 regularization term) = 0.0001 These parameters have been chosen by model selection through determination of performance on the validation set.

The learnable parameters for the model include determining the weight matrix of each layer present in the MLP and their biases. The hypothesis set is a function of these learnable parameters. We have ∼25000 training datapoints and about 13 features. This exceeds the general rule of thumb where number of datapoints $N \sim (3 - 10)d.o.f.$. Hence to avoid overfitting, we prevent the model regularization coefficient from getting too high during hyperparameter tuning. It looks like the

Dataset	**Specificity**	**Sensitivity**
Train	0.95	0.62
Test	0.86	0.64

Table 10: Results for Multilayer Perceptron

model is able to predict more percentage of 1's in the test set as compared to the train set, implying that it has learnt about the minority class. However, specificity is still quite low as compared to sensitivity due to the low numbers of the minority class.

3.4.7 Random Forest

Random forest consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest gives a class prediction and the class with the most votes becomes our model's prediction. The basic principle on which random forests operate is: A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models. Random Forests ensure that the constituent trees are not too correlated through

- Bootstrap Aggregation: Each individual tree is allowed to randomly sample from the dataset, with replacement, resulting in different trees. This process is known as bagging.
- Feature Randomness: To split a node, each tree in a random forest can pick only from a random subset of features. This forces even more variation amongst the trees in the model.

We chose this model since it is robust to outliers and can handle unbalanced data well. The parameters chosen for the final trained model include max depth = 60.0 and min samples split = 20.0 These parameters have been chosen by model selection through determination of performance on the validation set.

The learnable parameters for the model include determining the weights of the individual features and learning the feature and its threshold that the node should be split on. This would need to be learnt for each tree and then an average would be taken. The hypothesis set is a function of these learnable parameters and is highly complex. We have ∼25000 training datapoints and about 13 features. This exceeds the general rule of thumb where number of datapoints $N \sim (3-10) d.o.f..$ Hence to avoid overfitting, we prune the constituent trees depth, preventing them from getting too low. We also increase the min sample split value to avoid the trees continuing to split into further nodes where there are only few samples present.

Dataset	**Specificity**	**Sensitivity**
Train	0.94	0.72
Test	0.83	0.67

Table 11: Results for Random Forest

We see that the performance on the test set has reduced as compared to the train set. Since the minority class is significantly lesser in number, it appears that the model classifies a bunch of 1's as 0's leading to a low sensitivity score as compared to specificity.

3.5 Model Selection and Comparison of Results

For the supervised algorithms other than trivial and baseline models, we used grid search to find optimal hyper parameters by repeatedly performing stratified k fold technique and used sensitivity as a performance metric.

From Table 12, we see that almost all models are able to achieve a high specificity score, indicating that most of them are able to accurately classify 0's in the train and test set. The trivial model however has a low specificity, which is, as expected. Most models have a low sensitivity arising due to the fact that there are very few datapoints of the class in the dataset. This makes it difficult for the model to learn information about the data, leading to incorrect predictions.

4 Final Results and Interpretation

Based on Table 12, we see that surprisingly, Decision Tree Classifier emerges as the best model of all the models tested. It is able to give a specificity score of 86% and a sensitivity score of 64%. Out-of-sample performance for all our systems can be calculated according to :

$$\frac{\epsilon}{B_L} = \sqrt{\frac{1}{2N} log(\frac{M}{\delta})} \tag{7}$$

Algorithm	**Train Specificity**	**Train Sensitivity**	**Test Specificity**	**Test Sensitivity**
Trivial	0.15	0.85	0.14	0.85
Baseline	0.7	0.58	0.66	0.57
Logistic Regression	0.88	0.62	0.80	0.65
Support Vector Classifier	0.98	0.49	0.92	0.5
Decision Tree Classifier	0.86	0.64	0.95	0.62
Multi Layer Perceptron	0.95	0.62	0.86	0.64
Random Forest Classifier	0.94	0.72	0.83	0.6

Table 12: Results for all Supervised Algorithms

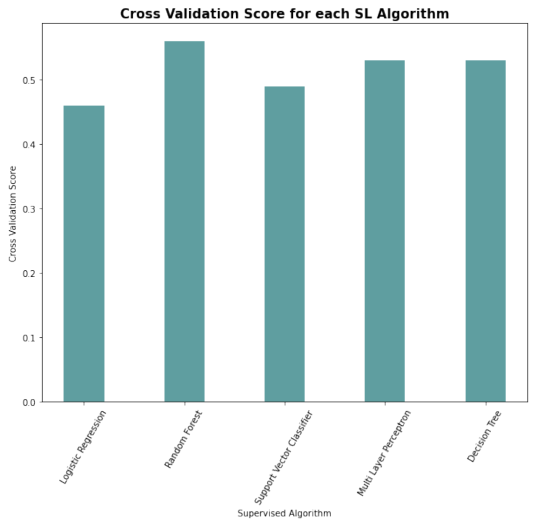


Figure 10: Best Cross Validation Score for SL Algorithms

where N is the number of datapoints (24,993), M is the cardinality of the hypothesis set, and δ is the tolerance. To calculate the out-of-sample performance using the validation set, we substitute the value of M with the hyperparameters that we tune over. Table 13 highlights the results, with $\delta = 0.1$.

To calculate the out-of-sample performance on the test set, we use M = 1 for all our models since the best hypothesis has already been identified. In this case, $\frac{\epsilon}{B_L} = 4.47 \times 10^{-3}$.

Decision Tree has been able to learn from the data and predict accurately on the test data. It is marginally able to outperform Random Forest on this dataset. This could be because the random forest is biased toward certain features resulting in it not being able to accurately predict labels for class 0, since there is a reduction in its specificity, on test data. It could also be due to the fact that multiple trees end up confusing the model further rather than providing a clear majority of votes. The best decision tree classifier has a max depth of 50.0 and uses a min sample split (minimum number of samples required to split an internal node) of 20.

There is a difference gap between the results of our baseline model and the decision tree classifier. The baseline model is able to achieve mediocre results on the test data, with almost equal values of specificity and sensitivity, leading us to the conclusion that quite a lot of 1's and 0's (more 0's than 1's) have been misclassified. This could be because a lot of class 0 datapoints may be lying closer to the class 1 mean than the class 0 mean (data is more spread out).

Our decision tree classifier is able to give us the best results since it is able to handle the large

Model	M	$\frac{\epsilon}{B_L}$
Logistic Regression	7	6×10^{-3}
SVM	4	5.66×10^{-3}
Decision Tree	6	5.96×10^{-3}
MultiLayer Perceptron	5	5.83×10^{-3}
Random Forest	6	5.96×10^{-3}

Table 13: Out-of-sample bounds on validation set

amount of data and classify features constructively to generate an accurate prediction. Pruning of the tree by restricting the maximum depth of the tree and restricting the samples required to split a node has helped the algorithm avoid overfitting on the given data. The model could be stimulated to work better through further dimensionality reduction, to avoid sparsity of the data matrix. Bias towards the dominant class can also be avoided by further downsampling the majority class.

5 Implementation for the extension

5.1 Data Set

Same as Section 3.1

5.2 Dataset Methodology

The dataset split into train and test remains the same as SL. However, now, we divide the train data into labeled and unlabeled datapoints to facilitate the model to learn and predict labels for the unlabeled data before being tested on unknowns. We implement 4 partition splits in the data: 40% labeled data (9,997 labeled training points - 14,996 unlabeled), 30% labeled data (7,498 labeled training points - 17,495 unlabeled), 20% labeled data (4,999 labeled training points - 19,994 unlabeled) and finally 10% (2,499 labeled training points - 22,494 unlabeled).

For SSL algorithms, cross-validation has not been implemented since the model is constantly and iteratively learning from the data. The entire training set is used to train the model. The test set is not used anywhere in the process flow until at the very end when the model is required to predict on unknowns. Hence, the test set does not exert undue influence in the model selection process. No decisions are made using the test set and it is used once during the workflow.

5.3 Preprocessing, Feature Extraction, Dimensionality Adjustment

Same as Section 3.3

5.4 Training Process

5.4.1 Baseline Model

We use the propagating nearest neighbour algorithm as our baseline model. This is a type of self training classifier where the model iteratively learns by moving datapoints from the unlabeled to the labeled dataset. We implement the algorithm using k = 17 nearest neighbours, taking into account the large size of the dataset and to prevent the model from being too sensitive to outliers.

The algorithm used is as follows:

- Initialize L = D_L (labeled dataset) and U = D_U (unlabeled dataset)

- Keep iterating until U = ϕ
- Find the k nearest points from L to datapoint x in U using Euclidean distance. Do this for all points in the unlabeled dataset.
- Points in U with the highest confidence of classification (above threshold = 0.95) are assigned a label according to the label that is predominantly present among its k neighbours. These points are then moved to L.
- Increment L count, decrement U count by the same number.

We chose this model as a baseline since it is a simple and straightforward method to implement as a baseline. The parameters chosen for the final trained model include k = 17 and confidence measure threshold = 0.95. These parameters have been chosen by trial and error, viewing how the model performs differently with each. We view the performance of the model by varying the number of labeled datapoints from 40% down to 10%.

We have ∼25000 training datapoints (after all datapoints are labeled) and about 13 features. This exceeds the general rule of thumb where number of datapoints $N \sim (3-10)d.o.f.$. Hence to avoid overfitting, we set a high confidence measure before moving datapoints from U to L, with the assumption that high confidence predictions are mostly correct.

Dataset	**Specificity**	**Sensitivity**
Train	0.97	0.74
Test	0.85	0.64

Table 14: Results for Propagating Nearest Neighbour with 40% labeled data

We see that the performance on the test set has reduced a bit compared to the train set. Since the minority class is significantly lesser in number, it appears that the model classifies a bunch of 1's as 0's leading to a low sensitivity score as compared to specificity.

5.4.2 Label Propagation

Label Propagation algorithm (LPA) is a fast algorithm for finding communities in a graph. LPA works by propagating labels throughout the network and forming communities based on this process of label propagation. The intuition behind the algorithm is that a single label can quickly become dominant in a densely connected group of nodes, but will have trouble crossing a sparsely connected region.

The algorithm used is as follows:

- Every node is initialized with a unique community label
- These labels propagate through the network
- At every iteration of propagation, each node updates its label to the one that the maximum numbers of its neighbours belongs to (we use k = 7). Ties are broken arbitrarily but deterministically.
- LPA stops if either convergence, or the user-defined maximum number of iterations is achieved

This model was chosen since it has quick running time and can propagate labels quickly between separated clusters. The parameters chosen for the final trained model include k = 7 and kernel = rbf. These parameters are chosen by heuristics. We view the performance of the model by varying the number of labeled datapoints from 40% down to 10%.

We have ∼25000 training datapoints (after all datapoints are labeled) and about 13 features. This exceeds the general rule of thumb where number of datapoints $Nsim(3-10)d.o.f.$. However, LPA does not effectively utilize node features and only views node labels, thereby helping combat overfitting.

Dataset	**Specificity**	**Sensitivity**
Train	0.97	0.67
Test	0.88	0.62

Table 15: Results for Label Propagation with 40% labeled data

We see that the performance on the test set has reduced a bit compared to the train set. Its performance is comparable to that of the baseline model. Since the minority class is significantly lesser in number, it appears that the model classifies a bunch of 1's as 0's leading to a low sensitivity score as compared to specificity.

5.4.3 Expectation Maximization

This algorithm performs maximum likelihood estimation in the presence of latent variables. It does this by first estimating the values for the latent variables, then optimizing the model, then repeating these two steps until convergence.

The Gaussian Mixture Model, or GMM, is a mixture model that uses a combination of Gaussian probability distributions and requires the estimation of the mean and standard deviation parameters for each. EM can be used here, for density estimation. The 2 steps in the algorithm are:

- E-Step. Estimate the missing variables in the dataset

  $$p(H|D, \theta) = \Pi_{h=l+1}^{l+u} p(y_h|X_U, Y_L, X_L, \theta) \tag{8}$$

  where H indicates the latent variables

- M-Step. Maximize the parameters of the model in the presence of data

  $$\theta^{t+1} = max(\theta) E_{H|D,\theta}(lnp(D|H, \theta) \tag{9}$$

This model was chosen since it tries to use the existing data to determine the optimum values for variables and finds model parameters. The parameters chosen for the final trained model include the mean and covariance statistics of the training data. Using these heuristics, the model tries to propagate labels to the unknown test data.We view the performance of the model by varying the number of labeled datapoints from 40% down to 10%.

Dataset	**Specificity**	**Sensitivity**
Train	0.42	0.74
Test	0.4	0.72

Table 16: Results for Expectation Maximization with 40% labeled data

The performance of the model is not great, implying that it is unable to accurately determine the right labels for the datapoints. This could be because the model is not completely correct in accordance with the data. Performance on the test and train data is about the same. Sensitivity is lower here, indicating that more 0's are incorrectly getting classified as 1's.

5.4.4 Semi Supervised Support Vector Machine

This algorithm constructs a support vector machine using both labeled and unlabeled training data and tries to maximize the margin width for separation between classes. It treats labeled data the same as SVM (hinge loss) and uses the hat loss to predict labels for unlabeled data.

$$L_{hat}(g(x_i)) = L_{hinge}(y_i, g(x_i)) = max[1 - |g(x_i)|, 0] \tag{10}$$

The complete optimization function can be stated as

$$J_{S3VM}(w, w_o, \lambda) = min(\Sigma_{i=1}^{l} max[1-y_i(w^T x_i+w_o), 0]+\lambda_1||w||^2+\lambda_2\Sigma_{i=l+1}^{l+u} max[1-|w^T x_i+w_o|, 0]), \lambda_1 > 0, \lambda_2 > 0 \tag{11}$$

subject to

$$\frac{1}{u}\Sigma_{i=l+1}^{l+u}(w^T x_i + w_o) = \frac{1}{l}\Sigma_{i=1}^{l} y_i \tag{12}$$

This helps avoid the case where most or all of the unlabeled data lies on the same side of the decision boundary. We use the QN-S3VM BFGS optimizer to implement the above algorithm.

This model was chosen since it uses a different approach to semi supervised learning. Rather than try to predict labels for the unlabeled data, it determines a decision boundary that minimizes the loss for both kinds of points in the dataset. The parameters chosen for the final trained model include the optimal weight vectors and constants λ_1 and λ_2. We view the performance of the model by varying the number of labeled datapoints from 40% down to 10%.

Dataset	**Specificity**	**Sensitivity**
Train	0.9	0.6
Test	0.55	0.58

Table 17: Results for S3VM with 40% labeled data

For the original training data, which remained imbalanced after undersampling, S3VM unfortunately determined a decision boundary classifying all the points on one side. To get a better idea of how it works, we undersample the data some more, to make the ratio of classes 1:1. Here we use the Instance Hardness Threshold method that trains a classifier on the data and the samples with lower probabilities are removed. The model is trained on this set of data and the above results are obtained. We see that specificity of the test data is very low as compared to the train data. This could be due to the fact that the test data has a much higher imbalance ratio as compared to the train data and the model ends up predicting some of the 0's as 1's.

5.4.5 Sensitivity and Perturbation Analysis

To extend our project, we decided to check the sensitivity of our SSL models to a small amount of noise on the training data. From Table 14 and 18, we see that, there is not a lot of change in the train and test data metrics for the Propagating NN, Label Propagation and S3VM models. However the EM model specificity drops by a large margin on the train and test data. This highlights the fact that the EM model does not align with the model specified by the data, resulting in a poor performance by the algorithm. It is unable to determine the right labels for datapoints and introducing a small amount of disturbance in the data causes a large difference in the metrics.

5.4.6 Comparing the performance of SL and SSL models

We also performed an experiment to compare the behaviour of SL vs SSL models on the same amount of labeled data. Both models use a Logistic Regression classifier at the core.

For the SSL model, we use 30% labeled training data, and implement a similar principal as the Propagating NN algorithm. The only difference is that we now use a Logistic Regression classifier to train on the labeled data and give predictions on the unlabeled data. Predictions with the highest confidence are moved to D_L from D_U and this process is continued till $D_U = \phi$. This model gives a specificity of 0.79 and sensitivity of 0.68 on the test data.

For the SL model, a general Logisic Regression classifier is used on the same training dataset (30% labeled training data) that is divided into K-folds, resulting in a validation set, for model selection. After selecting the best hyperparameters, the model gives a specificity of 0.82 and sensitivity of 0.65 on the test data.

We see that both models give very similar metrics on both the train and test data indicating that SSL performs just as well (or sometimes better) than SL on unknowns by predicting labels for datapoints.

5.5 Model Selection and Comparison of Results

Algorithm	**Train Specificity**	**Train Sensitivity**	**Test Specificity**	**Test Sensitivity**
Baseline(Prop NN)	0.96	0.76	0.84	0.64
Expectation Maximization	0.42	0.74	0.4	0.72
S3VM	0.90	0.60	0.55	0.58
Label Propagation	0.97	0.6	0.88	0.62

Table 18: SSL Algorithm Results for 40% labeled data

Algorithm	**Train Specificity**	**Train Sensitivity**	**Test Specificity**	**Test Sensitivity**
Baseline(Prop NN)	0.97	0.8	0.85	0.63
Expectation Maximization	0.55	0.72	0.51	0.72
S3VM	0.85	0.60	0.52	0.57
Label Propagation	0.97	0.6	0.88	0.62

Table 19: SSL Algorithm Results for 30% labeled data

Algorithm	**Train Specificity**	**Train Sensitivity**	**Test Specificity**	**Test Sensitivity**
Baseline(Prop NN)	0.98	0.82	0.86	0.64
Expectation Maximization	0.52	0.75	0.49	0.74
S3VM	0.85	0.60	0.51	0.58
Label Propagation	0.97	0.6	0.88	0.62

Table 20: SSL Algorithm Results for 20% labeled data

From the tables, we can infer that EM does not fit well to our data, indicating that the model to represent the data has not been approximated in a correct manner. S3VM does not do a great job at classifying the data as well. The specificity and sensitivity are very similar in values indicating that quite a lot of 0's and 1's have been misclassified. This could be due to the fact that the distribution of points in the train and test set differ a lot. The train set has a 50:50 ratio between points of both classes, whereas the test data has a severe imbalance in the class 1 datapoints. Label Propagation and Propagating Nearest Neighbour are able to perform quite well and accurately predict labels on unknowns.

6 Final Results and Interpretation for the extension

Based on the above tables, we see that Label Propagation performs best with all percentages of labeled data and gives a specificity score of 0.88 and a sensitivity score of 0.62 with all 4 partitions (10%, 20%, 30% and 40% of labeled data). Our baseline model - propagating nearest neighbour is also able to perform surprisingly well and gives high scores for the dataset. Both these techniques are community based - that is, they predict the label of datapoints based on the class majority of points around them. This method appears to be useful in predicting labels for our dataset.

As the number of labeled datapoints reduces, EM performance gradually reduces. As the statistics of the initial learned parameters includes lesser datapoints, the accuracy of predicted labels keeps decreasing. The same applies to S3VM as well, where-in its predicted decision boundaries are not able to accurately predict labels for the test set data at the same rate.

Algorithm	Train Specificity	Train Sensitivity	Test Specificity	Test Sensitivity
Baseline(Prop NN)	0.99	0.88	0.88	0.6
Expectation Maximization	0.52	0.75	0.45	0.70
S3VM	0.83	0.64	0.50	0.55
Label Propagation	0.97	0.6	0.88	0.62

Table 21: SSL Algorithm Results for 10% labeled data

Algorithm	Train Specificity	Train Sensitivity	Test Specificity	Test Sensitivity
Baseline(Prop NN)	0.97	0.74	0.85	0.64
Expectation Maximization	0.09	0.68	0.11	0.67
S3VM	0.90	0.60	0.55	0.57
Label Propagation	0.97	0.67	0.88	0.62

Table 22: SSL Algorithm Results for 40% labeled data with noise

Keeping the number of nearest neighbours too low, may lead to a decrease in performance of the Label Propagation algorithm. Finding the optimal value helps the algorithm give correct predictions and results in high metrics. The model could be stimulated to work better by reducing bias towards the dominant class (further downsampling the majority class).

After adding a small amount of Gaussian noise to the data, most of our algorithms remain stable except for the EM algorithm, implying that the model may be incorrect in relation to the data.

On comparing the performance between SSL and SL Logistic Regression Algorithms being trained on the same amount of data, we can infer that both models are able to give very similar results, with SSL giving a slightly higher sensitivity. We can conclude that SSL algorithms can perform better on unknowns, when trained with the same amount of labeled data.

7 Contributions of each team member

- **Nisha:** Worked on dimensionality reduction techniques to reduce the number of input features. Implemented the following Supervised learning algorithms: Trivial, Baseline, Logistic Regression and Support Vector Machine. Implemented the following Semi-Supervised learning algorithms: Prop NN, Expectation Maximization and S3VM. Performed the experiments for noisy data.
- **Muskaan:** Worked on preprocessing the numerical and categorical features of the dataset, handling "unknown" data points and Exploratory Data Analysis. Implemented the following Supervised learning algorithms: Decision Tree Classifier, Multi Layer Perceptron and Random Forest Classifier. Implemented the following Semi-Supervised learning algorithms: Label Propagation. Performed the experiments for converting SL dataset to SSL dataset.

We worked together on completing the report writing work.

8 Summary and conclusions

Decision Tree Classifier gives us the best results on our Supervised Learning Task - it provides a specificity score of 0.95 and a sensitivity score of 0.62 on the test dataset. Label Propagation Algorithm gives the best results on our Semi Supervised Learning Task - it provides a specificity score of 0.88 and a sensitivity score of 0.62 on the train dataset. We see that Semi-Supervised Learning is able to achieve comparable scores with Supervised Learning although the number of

Algorithm	Train Specificity	Train Sensitivity	Test Specificity	Test Sensitivity
SSL Logistic Regression	0.96	0.69	0.79	0.68
SL Logistic Regression	0.9	0.61	0.80	0.67

Table 23: SSL Algorithm Results for 30% labeled data by converting SL dataset to SSL dataset

labeled datapoints provided to the algorithm is significantly lower. SSL algorithms are able to predict labels for the unlabeled data and continue training.

One aspect of SSL that we would like to explore further is introducing class weights in SSL algorithms to handle imbalanced datasets. This was a disadvantage that we faced during the project, since our dataset is highly imbalanced and we needed to take this into account while training SSL algorithms. If algorithms take class weights into account, this would help increase their generalization ability as well.

References

[1] *Dataset from UCI Repository*, available at `https://archive.ics.uci.edu/ml/datasets/Bank+Marketing`

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

[3] S. Moro, R. Laureano and P. Cortez , *Using Data Mining for Bank Direct Marketing: An Application of the CRISP-DM Methodology*, in P. Novais et al. (Eds.), Proceedings of the European Simulation and Modelling Conference - ESM'2011, pp. 117-121, Guimaraes, Portugal, October, 2011. EUROSIS.

[4] "The following code for $S3VM$ was modified from `https://github.com/NekoYIQI/QNS3VM`"

[5] "The following code for $ExpectationMaximization$ was modified from https://towardsdatascience.com/implement-expectation-maximization-em-algorithm-in-python-from-scratch-f1278d1b9137"

A Code Submission

The code for our project can be found at:
`https://github.com/muskaan99/Predict-Client-Subscriptions-Semi-Supervised`