Statistical Machine Learning Individual Project

ISEG School Of Buisness

Submitted by: Noor Ayesha

Date: 31/03/2022

Contents

Abstract:

The data set is from a Portuguese banking institution to predict the success of a telemarketing (phone calls) campaign [1]. Specifically, the classification goal is to predict if the client will subscribe a term deposit to help the telemarketer focus on potential clients.

Dataset Attribute Information:

Input variables:

Bank Client Data:

- client_id (numeric)
- age (numeric)
- Job: type of job(categorical : ‘admin’, ’blue-collar’, 'housemaid','management','retired','self-employed','services','student','technician', 'entrepreneur', 'unemployed', 'unknown)
- Marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
- Education: (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree', 'unknown')
- Default: has credit in default? (categorical: 'no','yes','unknown')
- Housing: has housing loan? (categorical: 'no','yes','unknown')
- Loan: has personal loan? (categorical: 'no','yes','unknown')

Related with the last contact of the current campaign:

- Contact: contact communication type (categorical: 'cellular','telephone')
- Month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
- Day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')

Other attributes:

- Campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
- pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
- Previous: number of contacts performed before this campaign and for this client (numeric)
- poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')

Social and economic context attributes:

- emp.var.rate: employment variation rate - quarterly indicator (numeric)
- cons.price.idx: consumer price index - monthly indicator (numeric)
- cons.conf.idx: consumer confidence index - monthly indicator (numeric)
- euribor3m: euribor 3 month rate - daily indicator (numeric)
- nr.employed: number of employees - quarterly indicator (numeric)

Target variable

- Subscribe : did the customer subscribe? (binary: 1='yes', 0='no')

Type of Machine Learning Problem:

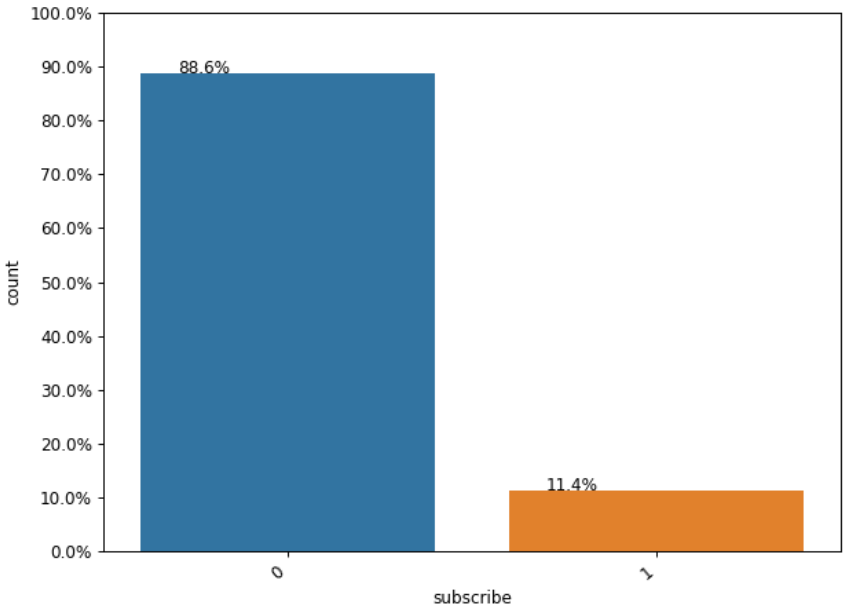
It is a binary classification problem where our objective is to predict whether a customer will subscribe a term deposit or not given the data of the customer.

Performance Metric:

The performance metric used in the experimental analysis is:

- AUC Score

Variable selection :



Points considered during variable selection are as follows:

- After doing EDA I figured out that the data is imbalanced, where "no" (i.e. 0 in above figure) is the majority class.
- After doing univariate analysis of I figured out that day_of_week and month features does not help very much when it comes to predicting the target variable. But on the other hand, some numerical features tend to predict the target variable much better.
- After basic data preprocessing, I encoded the categorical data into One hot encoding method as categorical data has big impact.

Machine Learning Algorithms used in this Project:

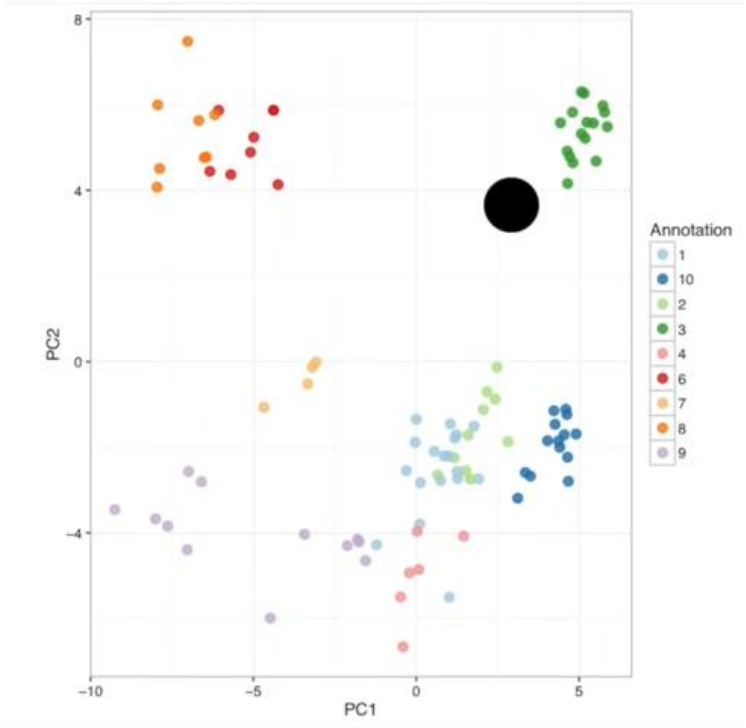
- KNN
- Logistic
- RBF Kernel SVM

- Random Forest
- AdaBoostClassifier

KNN (k-nearest neighbour):

kNN means k nearest neighbours. It is a super simple algorithm to classify the data.

Let us take an example of cell as shown in below picture to decide that it may belong to any group of cells of red, blue and green.



Briefly, if we already had many data that define different cell types we could use it to decide which type of cell this new cell is.

Step 1: will start with a data set with known categories in this case we will take different cell types from an intestinal tumour we then cluster that data as shown in picture into Red blue and green category.

Step 2: We can use PCA in step 2 to add a new cell with unknown category to the clot we do not know this cells category because it was taken from another tumour where the cells were not properly sorted and so what we want to do is we want to classify this new cell. In order to achieve that we want to figure out what cell this is most similar to and then we are going to call it that type of cell.

Step 3: we classify the new cell by looking at the nearest annotated cells i.e. the nearest neighbours if the K in K nearest neighbours is equal to 1 then we will only use the nearest neighbour to define the category in this case the category is green because the nearest neighbour is already known to be the green cell type if K equals 11 we would use the 11 nearest neighbours, in this case the category is still green because the 11 cells that are closest to the unknown cell are already green.

If suppose the new cell is somewhere else about halfway between the green and the red cells then if k equals 11 and the new cell is between two or more categories, we simply pick the category that gets

the most votes. In this case we can say that seven nearest neighbours are red three nearest neighbours are orange one nearest neighbour is green since red got the most votes the final assignment is red.

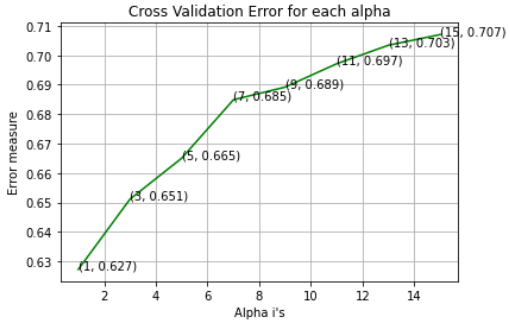
Points to consider on picking a value for K:

- There is no physical or biological way to determine the best value for K so we may have to try out a few values before settling on one do this by pretending part of the training data is unknown and then what we do is we categorize that unknown data using the K nearest neighbour algorithm and we assess how good the new categories match what we know already.
- The low values for K like k = 1 or K = 2 can be noisy and subject to the effects of outliers.
- The large values for K smooth over things but we do not want K to be so large that a category with only a few samples in it will always be out-voted by other categories.

KNN model Implementation Results:

The AUC for KNN Algorithm applied on our bank data appears to be as follows for Train set, Cross validation and AUC for Test set is 0.696.

```
AUC for k =  1 is 0.6272217567828833
AUC for k =  3 is 0.6511045233543139
AUC for k =  5 is 0.6650175792858012
AUC for k =  7 is 0.6848717893636369
AUC for k =  9 is 0.6891831897631988
AUC for k =  11 is 0.6971393382168661
AUC for k =  13 is 0.7034127444451604
AUC for k =  15 is 0.7070417391677944
```




```
For values of best alpha =  15 The train AUC is: 0.8450541134502733
For values of best alpha =  15 The cross validation AUC is: 0.7070417391677944
For values of best alpha =  15 The test AUC is: 0.7194050613783234
```

:

	auc_train_knn	auc_cv_knn	auc_test_knn
0	0.854025	0.719848	0.696495

Advantages of knn:

- High accuracy
- Simplicity
- versatility

Disadvantages of Knn:

- Accuracy depends on the quality of the data.

- With large data, the prediction stage might be slow.
- Sensitive to the scale of the data and irrelevant features.
- Require high memory – need to store all of the training data.
- Given that it stores all of the training, it can be computationally expensive.

Logistics Model:

Logistic Regression is used when the dependent variable (target) is categorical.

For example,

- To predict whether an email is spam (1) or (0)
- Whether the tumor is malignant (1) or not (0)

Consider a scenario where we need to classify whether an email is spam or not. If we use linear regression for this problem, there is a need for setting up a threshold based on which classification can be done. Say if the actual class is malignant, predicted continuous value 0.4 and the threshold value is 0.5, the data point will be classified as not malignant which can lead to serious consequence in real time.

From this example, it can be inferred that linear regression is not suitable for classification problem. Linear regression is unbounded, and this brings logistic regression into picture. Their value strictly ranges from 0 to 1.

Simple Logistic Regression

Model

Output = 0 or 1

Hypothesis => Z = WX + B

hΘ(x) = sigmoid (Z)

Sigmoid Function

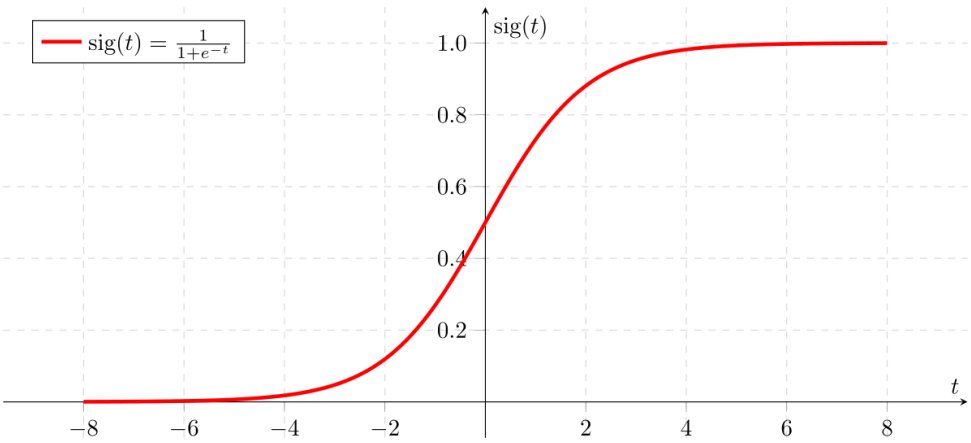


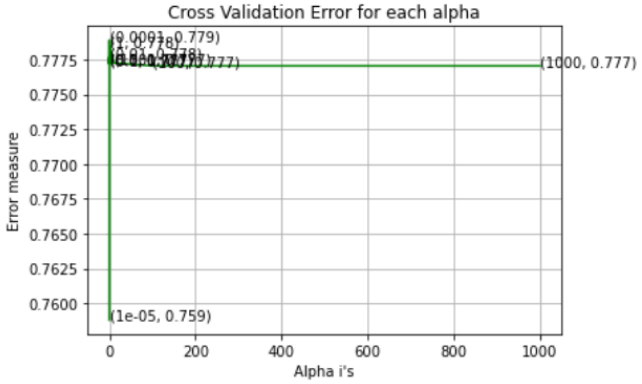
Figure: Sigmoid Activation Function

If ‘Z’ goes to infinity, Y (predicted) will become 1 and if ‘Z’ goes to negative infinity, Y (predicted) will become 0.

Logistic model Implementation Results:

The AUC for Logistic Algorithm applied on our bank dataset appears to be as follows for Train set, Cross validation and AUC for Test set is 0.770. Which is very good as compare to knn.

```
AUC for k =  1e-05 is 0.7588027807828919
AUC for k =  0.0001 is 0.7789023211030127
AUC for k =  0.001 is 0.7771831375032312
AUC for k =  0.01 is 0.7776700343699952
AUC for k =  0.1 is 0.7771735636884352
AUC for k =  1 is 0.7784263657388728
AUC for k =  10 is 0.7772665664607383
AUC for k =  100 is 0.7770518394717443
AUC for k =  1000 is 0.7770504717839162
```




```
For values of best alpha =  0.0001 The train AUC is: 0.7540178425524422
For values of best alpha =  0.0001 The cross validation AUC is: 0.7789023211030127
For values of best alpha =  0.0001 The test AUC is: 0.7706045153830113
```

Figure: Logistic Algorithm Implementation

	auc_train_logistic	auc_cv_logistic	auc_test_logistic
0	0.754018	0.778902	0.770605

Advantages and Disadvantages of Logistic:

- Logistic Regression struggles with its restrictive expressiveness (e.g. interactions must be added manually) and other models may have better predictive performance.
- Another disadvantage of the logistic regression model is that the interpretation is more difficult because the interpretation of the weights is multiplicative and not additive.

- Logistic regression can suffer from **complete separation**. If there is a feature that would perfectly separate the two classes, the logistic regression model can no longer be trained. This is because the weight for that feature would not converge, because the optimal weight would be infinite.

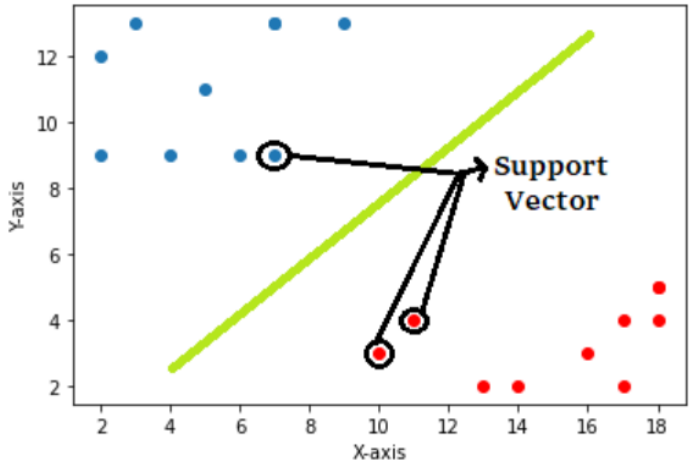
- On the good side, the logistic regression model is not only a classification model, but also gives us the probabilities. This is a big advantage over models that can only provide the final classification. Knowing that an instance has a 99% probability for a class compared to 51% makes a big difference.

- Logistic regression can also be extended from binary classification to multi-class classification.

SVM (Support vector Machine):

Support Vector Machine is a supervised machine learning algorithm. Support Vector Machine can be used both in linear and non-linear classification, regression, and even outlier detection. Support Vector Machine is a non-probabilistic linear classifier that uses the geometrical approach to distinguish the different classes in the dataset.

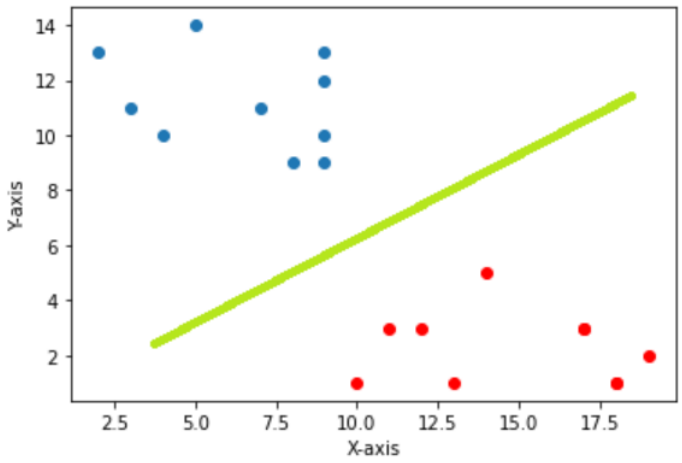
Vectors are the data points represented on the n-dimensional graph. For example, we represent a point on a 2D graph like (x,y) in below figure and in a 3D graph like as (x,y,z) where x,y,z are the axis of the graph.



Support Vectors

So, Support Vectors are the vectors on the n-dimensional graph which are the closest points to the hyperplanes and influence the orientation of the hyperplane. Through this support-vector, we pass the positive and negative margins for the hyperplane.

Example: In 2D graph representation we separate points using a line as shown below



Hyperplane for 2D

The green line represented in the picture is acting as the hyperplane and the equation of this hyperplane will be equal to the equation of the line that is

$$y = w_1 x + w_0$$

We can create this as:

$$w_2 x_2 + w_1 x_1 + w_0 = 0$$

HyperPlane: A hyperplane is nothing but just a decision boundary with (n-1) dimensions where n is the number of columns in the dataset. Hyperplane separates points/vectors of different classes.

Kernel: The kernel is a mathematical function used in SVM to transform non-linear data to a higher dimensional data set so that SVM can separate the classes of the data by using a hyperplane. In Scikit-learn, SVM supports various types of the kernel like *'linear', 'poly', 'rbf', 'sigmoid'*. If none is given, 'rbf' will be used.

For the experimental setup of the bank dataset, I have used default rbf kernel as other kernels took longer run time and performance was close to rbf.

RBF (Radial Basis Function) SVM Kernel:

RBF kernel in SVM classification, maps input space in indefinite dimensional space. Following formula explains it mathematically −

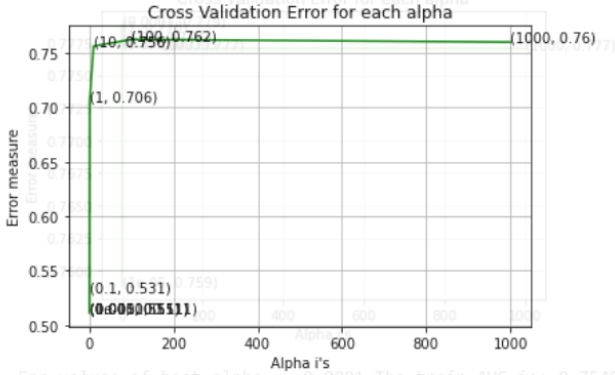
K(x,xi) = exp(-gamma * sum((x – xi^2))

Here, gamma ranges from 0 to 1. We need to manually specify it in the learning algorithm. A good default value of gamma is 0.1.

RBF Kernel SVM model Implementation Results:

The AUC for RBF kernel SVM Algorithm applied on our bank dataset appears to be as follows for Train set, Cross validation and AUC for Test set is 0.744. Which is good as compared to knn but not better than benchmark (so far) logistic.

```
AUC for C =  1e-05 is 0.511312829869208
AUC for C =  0.0001 is 0.511312829869208
AUC for C =  0.001 is 0.511312829869208
AUC for C =  0.01 is 0.511312829869208
AUC for C =  0.1 is 0.5309555624547808
AUC for C =  1 is 0.7062329637384926
AUC for C =  10 is 0.7561535694600778
AUC for C =  100 is 0.7623738137017702
AUC for C =  1000 is 0.7599707861879941
```




```
For values of best alpha for SVM=  100 The train AUC is: 0.7564267600703377
For values of best alpha for SVM =  100 The cross validation AUC is: 0.7623738137017702
For values of best alpha for SVM =  100 The test AUC is: 0.7443422894509526
```

	auc_train_RBF	auc_cv_RBF	auc_test_RBF
0	0.756427	0.762374	0.744342

Advantages of SVM:

1. SVM is effective in high-dimensional space.
2. It is still effective in cases where several dimensions are greater than the number of samples.
3. SVM is memory efficient.

DisAdvantages of SVM:

1. If the number of features is much greater than the number of samples, then it avoids over-fitting in choosing the kernel function.
2. SVM does not directly provide probability estimation like Logistic Regression.

Random Forest:

Random Forest is a powerful and versatile **supervised machine learning algorithm** that grows and combines multiple decision trees to create a “forest.” It can be used for both classification and

regression problems. Random Forest grows multiple decision trees which are merged together for a more accurate prediction.

The logic behind the Random Forest model is that multiple uncorrelated models (the individual decision trees) perform much better as a group than they do alone. When using Random Forest for classification, each tree gives a classification or a "vote." The forest chooses the classification with the majority of the "votes." When using Random Forest for regression, the forest picks the average of the outputs of all trees.

The key here lies in the fact that there is no correlation between the individual models—that is, between the decision trees that make up the larger Random Forest model. While individual decision trees may produce errors, much of the group will be correct, thus moving the overall outcome in the right direction.

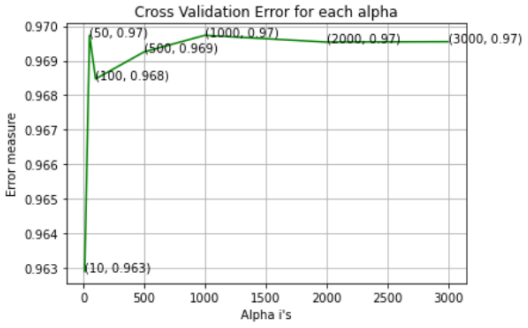
Training of trees in Random Forest:

Decision trees in an ensemble, like the trees within a Random Forest, are usually trained using the "bagging" method. The "bagging" method is a type of ensemble machine learning algorithm called Bootstrap Aggregation. An ensemble method combines predictions from multiple machine learning algorithms together to make more accurate predictions than an individual model. Random Forest is also an ensemble method.

Random Forest model Implementation Results:

The AUC for random forest Algorithm applied on our bank dataset appears to be as follows for Train set, Cross validation and AUC for Test set is 0.768. Which is good as compared to knn and RBF and its close to our logistic benchmark (so far).

```
AUC for number of estimators =  10 is 0.9628987322901521
AUC for number of estimators =  50 is 0.9697364875861815
AUC for number of estimators =  100 is 0.9684727440331199
AUC for number of estimators =  500 is 0.9692557453146434
AUC for number of estimators =  1000 is 0.9697358037422674
AUC for number of estimators =  2000 is 0.969538856695037
AUC for number of estimators =  3000 is 0.9695497981976609
```




```
For values of best alpha for random forest=  50 The train AUC is: 0.9999887496788853
For values of best alpha for random forest =  50 The cross validation AUC is: 0.9697364875861815
For values of best alpha for random forest =  50 The test AUC is: 0.76874826832497
```

	auc_train_RF	auc_cv_RF	auc_test_RF
0	0.999989	0.969736	0.768748

Advantages Of Random Forest:

- Ease of use
- Efficiency
- Accuracy
- Versatility – can be used for classification or regression

Disadvantages Of Random Forest:

- Because random forest uses many decision trees, it can require a lot of memory on larger projects. This can make it slower than some other, more efficient, algorithms.
- Sometimes, because this is a decision tree-based method and decision trees often suffer from overfitting, this problem can affect the overall forest.

XGBoost:

Gradient Boost has three main components.

- **Loss Function:** The role of the loss function is to estimate how best is the model in making predictions with the given data. This could vary depending on the type of the problem.

- **Weak Learner:** Weak learner is one that classifies the data so poorly when compared to random guessing. The weak learners are mostly decision trees, but other models can be used in GBM.

- **Additive Model:** It is an iterative and sequential process in adding the decision trees one step at a time. Each iteration should reduce the value of loss function. A fixed number of trees are added, or training stops once loss reaches an acceptable level or no longer improves on an external validation dataset.

XGBoost, which stands for Extreme Gradient Boosting is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning library. It provides parallel tree boosting.

XGBoost Features:

- **Regularized Learning:** Regularization term helps to smooth the final learnt weights to avoid over-fitting. The regularized objective will tend to select a model employing simple and predictive functions.

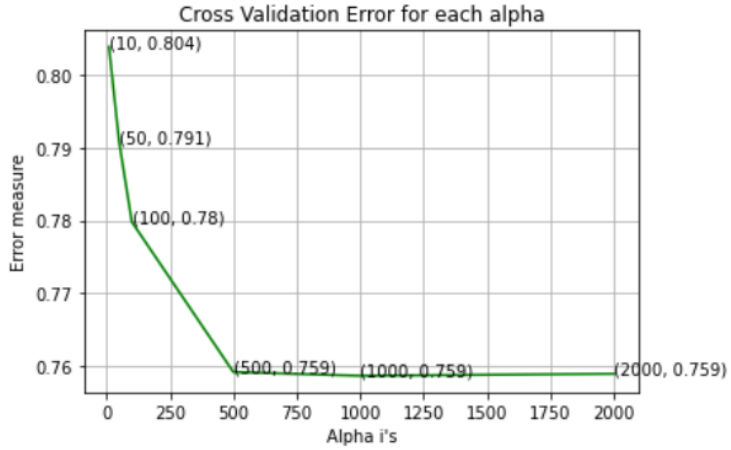
- **Gradient Tree Boosting:** The tree ensemble model cannot be optimized using traditional optimization methods in Euclidean space. Instead, the model is trained in an additive manner.

- **Shrinkage and Column Subsampling:** Besides the regularized objective, two additional techniques are used to further prevent overfitting. The first technique is shrinkage introduced by Friedman. Shrinkage scales newly added weights by a factor η after each step of tree boosting. Similar to a learning rate in stochastic optimization, shrinkage reduces the influence of each tree and leaves space for future trees to improve the model.

XGBoost Tree Methods: XGBoost has 4 built-in tree methods namely

1) **Exact:** exact means XGBoost considers all candidates from data for tree splitting. During each split finding procedure, it iterates over all the entries of input data. It's more accurate (among other methods) but slow in computation performance.
2) **Approx:** This algorithm builds a gradient histogram for each node and iterate through the histogram instead of real dataset.
3) **hist :** It runs sketching before training using only user provided weights instead of hessian. The subsequent per-node histogram is built upon this global sketch. This is the fastest algorithm as it runs sketching only once.
4) **gpu_hist:** The gpu_hist tree method is a GPU implementation of hist, with additional support for gradient based sampling.

XGBoost model Implementation Results: The XGBoost Algorithm with the tree methos as "exact" applied on our bank dataset appears to be as follows for Train set, Cross validation and AUC for Test set is 0.816. Which is best of all the models tested so far.




```
For values of best alpha for XGBoost =  1000 The train AUC of XGBoost is: 0.8162947785394076
For values of best alpha for XGBoost =  1000 The cross validation AUC of XGBoost is: 0.8162947785394076
For values of best alpha for XGBoost =  1000 The test AUC of XGBoost is: 0.8162947785394076
```

	auc_train_GBoost	auc_cv_GBoost	auc_test_GBoost
0	0.816295	0.820314	0.785385

Advantages Of XGBoost:

- XGBoost utilizes the power of parallel processing and that is why it is much faster than GBM. It uses multiple CPU cores to execute the model.
- XGBoost has an in-built capability to handle missing values. When XGBoost encounters a missing value at a node, it tries both the left and right hand split and learns the way leading to higher loss for each node. It then does the same when working on the testing data.
- XGBoost on the other hand make splits upto the max_depth specified and then start pruning the tree backwards and remove splits beyond which there is no positive gain.

- XGBoost accepts sparse input for both tree booster and linear booster and is optimized for sparse input.
- It supports customised objective function as well as an evaluation function.

Disadvantages Of XGBoost:

- The biggest limitation is probably the black box nature.
- Trees in general are unstable. That is, small changes to the feature set or training set can result in significantly different model

Evaluation metrics:

```
metric_evals = pd.concat([XGBoost_val,knn_val,Logistic_val,RBF_val,RandomForest_val], axis=1)
metric_evals
```

	auc_train_GBoost	auc_cv_GBoost	auc_test_GBoost	auc_train_knn	auc_cv_knn	auc_test_knn	auc_train_logistic	auc_cv_logistic	auc_test_logistic	auc_trai
0	0.816295	0.820314	0.785385	0.854025	0.719848	0.696495	0.754018	0.778902	0.770605	0.7

Clearly XGBoost is having highest AUC hence we can conclude that XGBoost with “exact” tree method is better performing model.

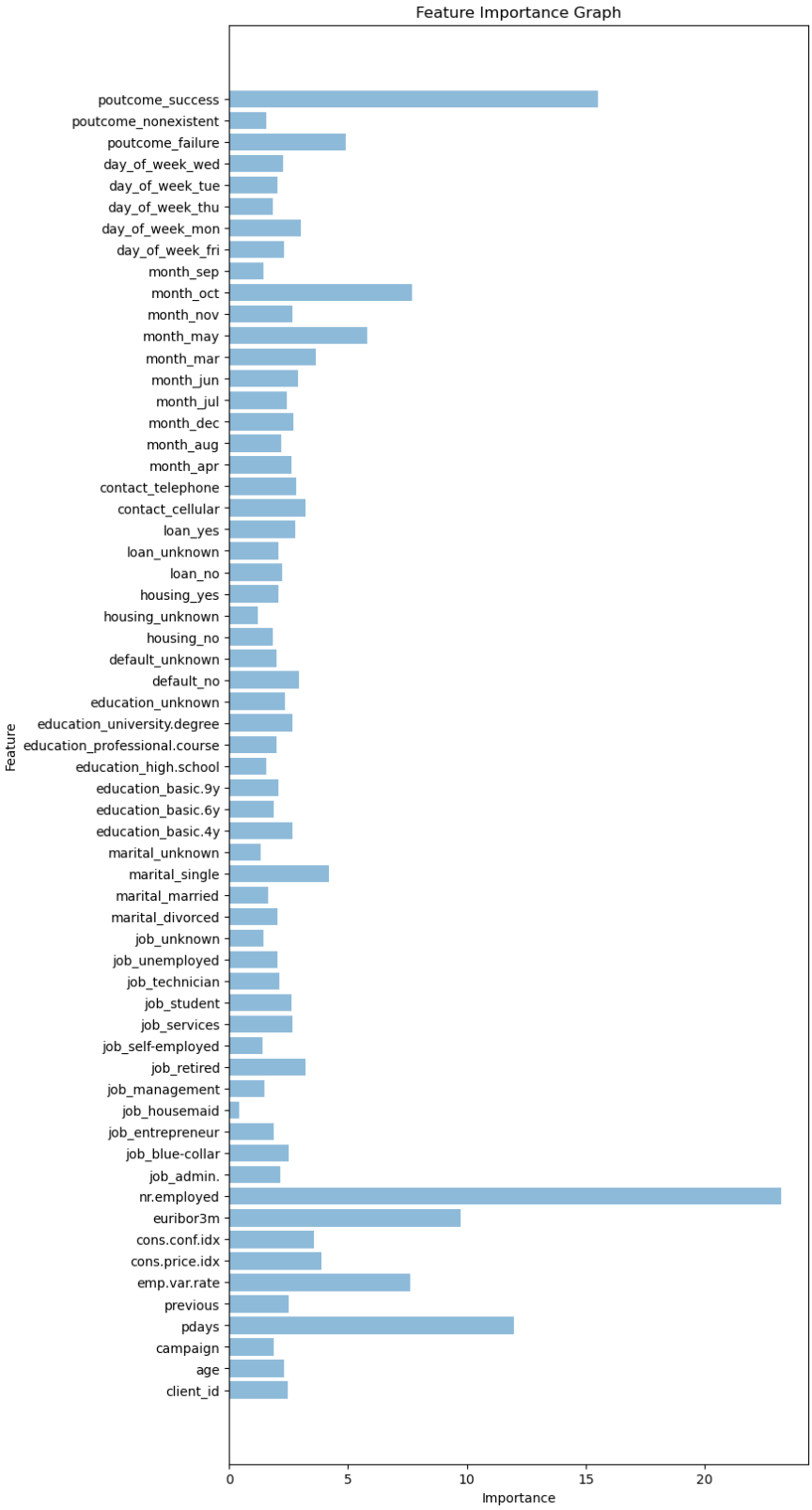
XGBoost with RandomizedSearchCV hyper parameter tuning:

Out of All the models XGBoost has highest AUC so I have applied hyper parameter tuning to XGBoost.

```
t eval_metric if you'd like to restore the old behavior.
For values of best alpha = 200 The train AUC is: 0.9126903000296692
For values of best alpha = 200 The test AUC is: 0.787057845642336
```

Unfortunately, The hyper tuning did not really make any impact on AUC. its lower than original XGBoost.

Feature Importance:



From the above feature importance graph we can see that, the most relevent features are the following:

- nr.employed

- emp.var.rate
- poutcome_success
- euribor3m

Conclusion:

- There were a lot of categorical variables and some numerical variables which capture various information about the customer and the bank-customer relationship.
- According to the plot for both logistic regression and XG Boost, we can tell that the most influential variables are nr.employed, euribor3m, poutcome_success and emp.var.rate.
- “nr.employed”, which is the number of employees in the bank, has positive effect for turning people to subscribe the term deposit. This can be due to the fact that the more employees the bank have, the more influential and prestigious this bank is.
- Employment variation rate (emp.var.rate) has negative influence, which means the change of the employment rate will make customers less likely to subscribe a term deposit.
- **Best model is XGBoost as** It repetitively leverages the patterns in residuals, strengthens the model with weak predictions, and make it better. By combining the advantages from both random forest and gradient boosting, XGBoost gave the a prediction error lower than the boosting or random forest in my experimental study.

References:

Support Vector Machine - an overview | ScienceDirect Topics

Support Vector Machine (SVM) (tutorialspoint.com)

Random forest - Wikipedia

Random Forest | Introduction to Random Forest Algorithm (analyticsvidhya.com)

XGBoost Documentation — xgboost 1.5.2 documentation