

# Case Study: Predicting Acceptance of Personal Loans Using Machine Learning Algorithms

Supraja Rapuru, Bellevue University  
DSC630 - Predictive Analytics

## 1 Introduction, Motivation

Banks spend millions of dollars on sales teams that try to find customers who will accept a personal loan if offered. The rate at which individuals accept these personal loans is very low (-10%). The low acceptance rate means that randomly choosing customers to market to is incredibly wasteful in terms of both time and money. Thus, it is in the interest of the banks to determine which customers are the most likely to accept this type of loan based on the customer's information. For my case study, I aim to apply the K-Means, Support Vector Machine, Logistic Regression, Perceptron, and Multilayer Perceptron algorithms to predict whether bank customers will be willing to take on a personal loan. The input into my algorithms will be a customer's age, years of professional experience, annual income measured in thousands of dollars, home address, ZIP code, family size, average spending on credit cards per month measured in thousands, education level, and the value of a house mortgage (if the customer has any). My algorithm will use this information to output a prediction on whether a customer will take a personal loan (1) or not (0). In practice, sales associates can then target these customers knowing that they are not wasting their time and effort on individuals who are unlikely to take a loan.

## 2 Related Work, Literature Review

This personal loan scenario can be categorized under the umbrella problem of classification for unbalanced datasets. In my example, this unbalance is due to the fact that there were more people that declined the personal loan than accepted one. While my situation deals with bank customers, there are many other unrelated situations where the dataset contains a disproportionate number of labels in a specific group. This characteristic can cause some difficulties in classification because, ML algorithm tends to learn more from majority group and may easily mis-classify the small data group.

One general solution to this problem is simply under-sampling the larger class in order to balance out the dataset. This was explored by Bao Lei and colleagues in their article about trying to **optimize support vector machines utilizing under-sampling techniques**. Conversely, another common solution to this problem is going the other direction and over-sampling points from the minority class. One such work that successfully used this was by **J.Piyasak, K. Wong, and C. Fung where they combined an oversampling algorithm (SMOTE) with Complementary Neural Network**. Interestingly, Chris Drummond and Robert Holte compared these two general methods and came to the conclusion in their paper that under-sampling was better than over-sampling. In short, they saw that "under sampling produces a reasonable sensitivity to changes in misclassification costs and class distribution. On the other hand, over-sampling is

surprisingly ineffective." While I did not implement either of these methods in this case study, I am interested in exploring these solutions in the term project work to see if my results agree with their reasoning.

Among other strategies, many other works focus on fine-tuning the algorithm's parameters, such as efforts to optimize the parameters of support vector machines, as researched by Eitrich Tatjana and Bruno Lang in their paper. This is the strategy that i focused on in this case study and the one that i believed made the most significant impact on the overall accuracy.

Furthermore, it was relatively quick and easy to adjust parameters of our algorithms because i implemented the algorithms using scikit libraries, which provide simple ways to fine-tune and adjust the parameters.

### 3 Dataset and Features

My dataset is titled "**Personal Loan Modeling**" from Kaggle. It contains a list of 5000 bank customer's data resulting from a marketing campaign. For each customer, there are 12 features and a [0,1] label of whether they accepted a personal loan offered in the bank's marketing campaign. The specific features for each customer are described in detail below:

# Age	# Experience	# Income	# ZIP Code	=	# Family	# CCAvg
Customers age in completed years	Number of years of professional experience	Annual income of the customer (\$000)	Home Address ZIP code.		Family size of the customer	Avg. spending on credit cards per month (\$000)
# Education	# Mortgage	# Securities Account	# CD Account		# Online	# CreditCard
Education Level. 1: Undergrad; 2: Graduate; 3: Advanced/Professional	Value of house mortgage if any. (\$000)	Does the customer have a securities account with the bank?	Does the customer have a certificate of deposit (CD) account with the bank?		Does the customer use internet banking facilities?	Does the customer use a credit card issued by this Bank?

I have opted to use 80% of our dataset for training (4000 examples), and the remaining 20% (1000 examples) for testing. Thankfully, our dataset was very easy to use and not much preprocessing was needed. The data did however initially contain a column of indices labeled 1- 5000 which we removed before inputting into algorithms as it was simply a numbering of data points. In terms of normalization, I tried augmenting the dataset for our Multi-Layer Perceptron Classifier by scaling the values between [0,1] using the scikit function *MinMaxScalar0* . As mentioned above, i hope to explore under-sampling and over-sampling with this dataset in the future.

## 4 Methods

### 4.1 Perceptron

I have initially implemented the Perceptron algorithm because it is a classic algorithm used for classification. The Perceptron algorithm takes in data labeled with either 0's or 1's and learns the weights of features that can then be used for prediction. The algorithm first randomly initializes the weights. Then, a data point is randomly selected. If the data point is labeled 1 and the dot product between the current weights and the data point is greater than 0, then the data point's values are added to the weights. If the data point is labeled 0 and the dot product of the weights and input is equal to or less than 0, then the input data is subtracted from the weights. The algorithm iterates this process until convergence. Prediction is then performed by taking the dot product of the weights and a new data point. If the dot product is equal to or greater than 0, then the algorithm predicts **1**. If the dot product is less than 0, then the algorithm predicts 0.

## 4.2 Logistic Regression

Next, I have implemented the Logistic Regression algorithm for classification. The Logistic Regression algorithm works by using the dot product of the weight vector with a data point as the input into the sigmoid function. Once these weights are found on the training data, we can dot the weights with new data to predict which class a data point is in.

## 4.3 K-Means

Another algorithm I experimented with was the KMeans Algorithm. With this algorithm, classification is decided based on a point's distance to any number of centroids. In my case, i only used two centroids because an individual can either accept the personal loan, or reject it. To start, the centroids will be randomly initialized to two of the data points in the dataset. Then, data points will be assigned to a centroid depending on which one is closer. Next, the centroids will re-center to be the mean of all points in their respective clusters. This process repeats until convergence is reached. For KMeans, convergence is reached when the two clusters created are the same as the clusters in the last iteration of the algorithm.

## 4.4 Support Vector Machines

Support Vector Machines work by finding a hyperplane that best separates the two groups of interest. The best hyperplane in this case is the one that maximizes the distance between the two groups. The distance is measured from the main dividing hyperplane to support vectors (hence the name of the algorithm) The maximum margin between the two groups is found through iterative steps of gradient descent as with many other algorithms Finally, the prediction is made by simply checking which side of the hyperplane a data point lies.

# 5 Experiments/Results/Discussion

I initially experimented with a single perceptron model. The perceptron algorithm performed poorly on our data. Every time i ran the algorithm, it predicted all zeros for our testing data. In context, this means that the algorithm predicted that no customers would take personal loans. If our entire motivation is to better target individuals who are likely to accept a personal loan, an algorithm that predicts that no one will accept a loan is not helpful. I believe that the reason for this outcome of the perceptron algorithm is due to the imbalanced nature of our data. Since the portion of customers that did take a loan is around 10%, we think that the algorithm found that all of the customers refusing a loan was the most likely scenario.

I ran into the same problem with logistic regression. The algorithm predicted all zeros, or that no one would accept a loan. Since our dataset is unbalanced, this still meant that the algorithms had about 90% accuracy. This made it clear that the algorithm's accuracy was not an indicator of success. So, I looked for different measurements. I have explored Receiver Operating Characteristic (ROC) Curve. This curve is a graphical representation showing how a classification model performs at all classification thresholds. Additionally, the Area Under the Curve (AUC) measurement is often used to quantify the evaluation of a curve. The closer to one, the better.

Afterwards, I have decided to move away from probabilistic models and try the K-means algorithm. The K-means algorithm did manage to predict some ones and had an accuracy of around 90%, with 7 false positives and 72 false negatives. While the ROC curve for the K-means algorithm did not have as high of an AUC score as logistic regression (.75 vs .81), we felt that K-means was more useful than logistic regression and the perceptron as it was able to correctly predict some customers who would take a loan. Next, I have implemented the Support Vector Machines algorithm with a linear kernel. This clearly outperformed all of the aforementioned approaches with an AUC of .94 and 27 false positives and 56 false negatives.

I believe that SVM and K-Means worked well because they do not rely on probabilistic likelihoods which can overshadow minority data when predicting. Instead, they create a more geometric representation of the two classification groups allowing for predictions based on a data points location in multidimensional space.

## 6 Conclusion/future works

In this case study, I set out to predict a customer's acceptance of a bank's marketing campaign for a personal loan. Through experimenting with several algorithms, I was proved that using support vector classifier was the clear winner in terms of both accuracy and ROC AUC score. I believe these algorithms outperformed the rest of the group because the Perceptron and Logistic Regression algorithms probabilistic nature could not handle the intricacy of the unbalanced dataset. Furthermore, the K-means algorithm is perhaps too simplistic in its approach and therefore were unable to accurately classify the minority of people who would accept the personal loan with these methods.

For future work, I would like to explore MLP network classifier and also research more solutions for classification problems with unbalanced datasets. As mentioned above, a specific area of improvement i would like to pursue is under-sampling and/or over-sampling in my dataset. Came across a literature recently using combination of under and over sampling techniques to increase effectiveness of data so would explore that and also will research if hyper parameter tuning could improve the prediction power of algorithms I am using.

## 7 References

- Bao, Lei, et al. "Boosted near-miss under-sampling on SVM ensembles for concept detection in large-scale imbalanced datasets." *Neurocomputing* 172 (2016): 198-206.
- Data, Anirban. "Personal Loan Modeling." *Kaggle*, 2020.  
<https://www.kaggle.com/teertha/personal-loan-modeling/activity>
- Drummond, Chris, and Robert C. Holte. "C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling." *Workshop on learning from imbalanced datasets*. Vol. 11. Washington DC: Citeseer, 2003.
- Eitrich, Tatjana, and Bruno Lang. "Efficient optimization of support vector machine learning parameters for unbalanced datasets." *Journal of computational and applied mathematics* 196.2 (2006): 425-436.
- Jeatrakul, Piyasak, Kok Wai Wong, and Chun Che Fung. "Classification of imbalanced data by combining the complementary neural network and SMOTE algorithm." *International Conference on Neural Information Processing*. Springer, Berlin, Heidelberg, 2010.
- Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.
- Ye, Leihua. "Classify A Rare Event Using 5 Machine Learning Algorithms " *Towards Data Science*, 19 Oct. 2019, [towardsdatascience.com/classifying-rare-events-using-five-machine-learning-techniques-fab464573233](https://towardsdatascience.com/classifying-rare-events-using-five-machine-learning-techniques-fab464573233).