Loan Approval Prediction: A CRISP-DM Implementation

1st Vicente A. Lomelín-Ibarra
School of Engineering and Science
ITESM
Monterrey, Mexico
0000-0003-1454-6701

***Abstract*—Data from home loans from the Dream Housing Finance (DHF) company were analyzed. The company aims to automate the loan eligibility process in real time with the information provided with the online application form from their customers. Using the knowledge from the course, a CRISP-DM methodology was implemented to determine whether a predictive model can be employed to aid in this process.**

***Index Terms*—data science, CRISP-DM, predictive modeling**

I. Introduction

Dream Housing Finance (DHF) is a company that deals with home loans. The process for home-ownership requires a great investment, and the approval of a home loan represents a great aid for new homeowners to be able to afford a new home for themselves and their families. Given the range of presence of DHF over urban, semi-urban and rural sectors, the determination of eligibility for a loan can be ardour, and the automatizing the process might help improve customer service by indicating if the customer is eligible through an initial online application form. Given the objective of predicting home loan eligibility, the questions are raised: Can the process be automated? In addition, if it can, what is the most appropriate model to predict loan approval? Using historical data provided by DHF, several attributes are explored, and based on this information, machine learning models are evaluated to predict loan eligibility and steps that might help the company improve their service through the application of CRISP-DM [1] methodology to develop the project.

II. Method

A. Python Programming

Python provides numerous libraries for data exploration and model implementation. The project was programmed with a *Jupyter Notebook* with *Python 3*. The libraries and packages used for this work include: *numpy*, for array manipulation, *pandas* for dataframe manipulation and operations, and *matplotlib* and *seaborn* for the visualization of data. For machine learning implementation, libraries from *sklearn* were employed since they include algorithms and tasks that the users can control in preparing the data and building the models.

B. Data Exploration and cleaning

The historical data provided by DHF contains information from 614 entries across 13 attributes, both categorical (LoanID, Gender, Married, Dependents, Education, SelfEmployed, PropertyArea and LoanStatus) and numeric (ApplicantIncome, CoapplicantIncome, LoanAmount, LoanAmountTerm and CreditHistory). The target variable is a binary 'Y' (approved loan) or 'N' (declined loan). The first step consists of loading the dataset into a pandas dataframe for data manipulation. The variable of LoanID was dropped since it indicates an identification for the customer, and does not provide any useful information for the data analysis or future predictive models. The next step consists of exploring and cleaning the data. Plots for every predictor attribute was plotted though seaborn and matplotlib packages. For categorical data, the frequency of each value was explored. During this phase, it was noted a mild unbalance on the dataset; however, the degree is not high enough that it requires sampling measures to deal with the class imbalance. In the case of the numeric attributes, the distribution of the variables was explored. Missing values were detected in the attributes of Gender, Married, Dependents, SelfEmployed, LoanAmount, LoanAmountTerm and CreditHistory, for 149 missing values. Before proceeding, the data must be cleaned. Given the number of observed missing values, eliminating these entries is not a viable option, since it would signify a great loss of data. Therefore, depending on the type of data, imputation methods were used to substitute the missing values. Since there were no observable relations that could help infer values to impute the missing values, different approaches were used depending on the behavior of the attributes observed through the initial visualizations. For the categorical attributes, the imputation values are considered to the values with the highest frequency. In contrast, the numeric values employed two different techniques. For the LoanAmountTerm, the data appear ordinal, therefore the imputation made used of the observed median of the attribute. A similar approach was considered for CreditHistory since the data are binary (1 for good credit history and 0 for bad credit history), the imputation also considered the observed median of the attribute. In contrast, the LoanAmount attribute considered the observed mean of the attribute to replace the missing values. A heatmap was drawn using seaborn's heatmap with Spearman's correlation to measure the degree to which the rankings of each attribute align with, and determine there is a strong correlation between the target variable of LoanStatus and the rest of the independent variables. The only

strong correlation observed with the target variable was that of the CreditHistory.

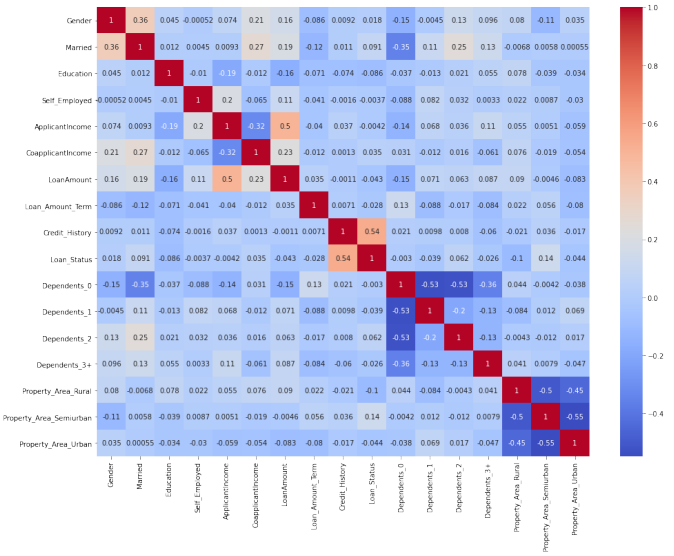

Fig. 1. Spearman Correlation heatmap of ranking for each of the variables of the historic loan data.

Additionally, the feature importance algorithm of ExtraTreesClassifier from the *sklearn* library was used to help determine which of the attributes could prove to be the most important for the model. The algorithm was iterated 10 times, and the feature importance was calculated through the mean value of the results. The complete results are shown in the included Jupyter Notebook. Consistent with what was observed in the correlation matrix, the most relevant feature was determined to be CreditHistory, followed by ApplicantIncome, LoanAmount and CoapplicantIncome. For the predictive analysis, seven machine learning models were fitted on training data to learn the parameters, and tested with hold out data from the original dataset.

C. Predictive Modeling

For the modeling stage of the project, the data is divided into training and test sets. The training set is used to fit the models to learn parameters, and tested using the test set to obtain each of the model's performance and to determine the predictive capacity from each considered models. The data split is performed using the *train_test_split* function of *sklearn*, in which the complete data and target variables are provided as input along with a split ratio. The models used for this work are briefly described in the following subsections.

1) Decision Trees: The model is implemented using *sklearn's* package of *DecisionTreeClassifier*. The classifier is a simple and effective method for classification tasks. The tree structure allows to split the data based on the attributes at every node of the tree until the leaves belong to the same class. The data split is performed through information gain based on the purity measure of entropy.

2) Support Vector Machine: The Support Vector Machine (SVM) is a mathematical-based model that proposes a linear function to separate the data, along with boundaries to maximize the area for the classification task. The SVM model seeks to determine the optimal space separation for the target variable classification task. The model is implemented through the *SVC* package from *sklearn*.

3) Logistic Regression: Provided by the package *LogisticRegression* from *sklearn*, a probability estimate is calculated and fitted to the model to create classification boundaries for each class based on these probabilities. The model allows to perform a binary classification by reducing the space of the data prediction to values between 0 and 1.

4) Naive Bayes: The Naive Bayes classifier is a probability classifier based on the Bayes theorem. The classifier assumes that the presence or absence of an entry is not related to the presence or absence of any other of the attributes. Naive Bayes classifiers are known to work well with small data sets, and make use of the mean and variance of the attributes, and do not require the co-variance of the attributes. The implemented model was constructed using the *GaussianNB* package from *sklearn*.

5) Random Forest: The Random Forest classifier is an ensemble learning method for classification problems. The algorithm builds multiple decision trees to fit the model. The trees are intentionally overfit, and a single tree from the forest is used to perform the classification. The model was implemented through the *RandomForestClassifier* package from the *sklearn* ensemble library.

6) XGBoost: The Extreme Gradient Boosting is an advanced implementation of the Gradient Boosting algorithm that has proven to have high predictive power. The algorithm incorporates regularization methods to reduce over-fitting. The python implementation was performed using the *XGBClassifier* package from the *xgb* library.

7) Gradient Boosting Algorithm: The Gradient Boosting algorithm uses boosting for the classification tasks. The algorithm uses a set of weak learners to create a strong learner, combining the mean predictions of the weak learners to generate a prediction. The implementation of the model was performed through the *GradientBoostingClassifier* ensemble package from *sklearn*.

D. Model Evaluation

To evaluate all the models, the accuracy and AUC scores were used. Since there is a mild imbalance in the classes, the accuracy score might not provide all the necessary feedback to evaluate the models, and the AUC score helps complement the evaluation. The evaluation is performed through a 10-fold cross-validation using *sklearn's* cross_validate function. The algorithms' accuracy and AUC score comparison are presented in Fig. 2 and Fig. 3 respectively. The worst results are observed with the SVM, with consistent values around 70%, along with Decision Trees classifiers, with a wide distribution of the results, but with a median lower than 70%. The Random Forest classifier presents better results, with a median accuracy

of approximately 78%, a max of 85% and a minimum of approximately 72%. Similar results are observed with the Naive Bayes and GBM classifiers, with a median value around 77%. The XGB classifier provided the highers max value, with a score higher than 85%, however, the median, the first and third quartile with values lower than 80%. The best results were obtained through the Logistic Regression classifier, with the median and the third quartile being higher than 80% and the max score reaching 85%. Given these results, the Logistic Regression model might be the best modeling technique to predict the loan approval. Given the evaluation of the models, the best model appears to be the Logistic Regression model, with the highest mean accuracy (80.30%) and mean AUC (0.76) scores.

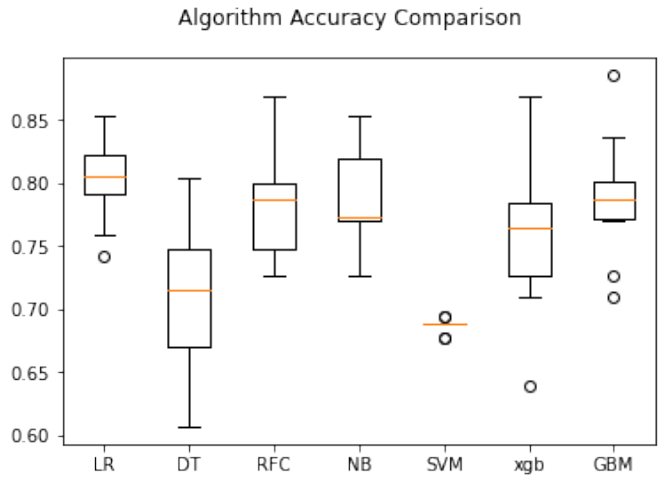

Fig. 2. Algorithms' accuracy score comparison of 10-fold cross-validation evaluation.

E. Hyper-parameter tuning

Given the models, the hyper-parameters of each model can be tuned to obtain the optimal parameters for each model. The *GridSearchCV* package from *sklearn* allows the perform a quick search for the different parameters of each model, barring the Naive Bayes model, which takes priors as parameter, which are essentially non-existent in this project. Given that the function uses a brute force approach to determine the optimal parameters for each model, the number of parameters and parameter values was limited due lack of computational resources and time to be able to perform an exhaustive search. The parameters considered for each model are briefly explained in the following subsections.

1) Decision Trees: The model hyper-parameter tuning for the Decision Trees classifier is performed through an iterative search of the models split criterion and depth of the trees. The process consists of training the model with different values for each parameter and obtaining their accuracy and AUC scores. Plotting the results from each constructed model allows us to visually identify which combination of parameters returns the best scores. The optimal parameter for the Decision Trees

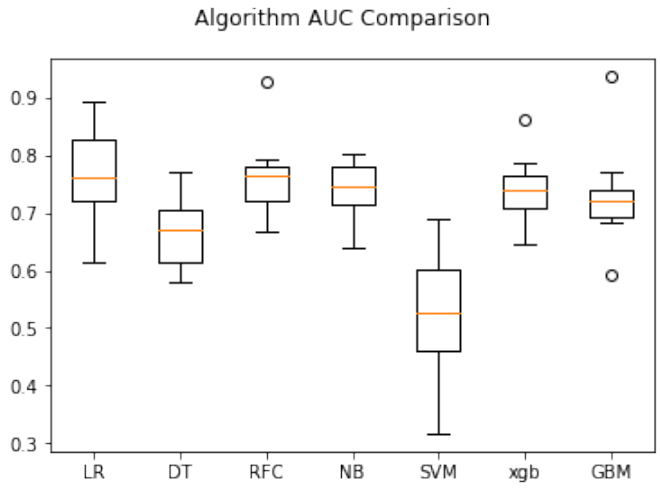

Fig. 3. Algorithms' AUC score comparison of 10-fold cross-validation evaluation.

model for this work was found to be the gini criterion along with a max depth of the trees of 5. Nevertheless, the optimal parameters could not significantly improve the accuracy score of the implemented model.

2) Support Vector Machine: The SVM model hyper-parameter tuning process may include the search of the *C*, gamma, and kernel parameters of the function. C is a regularization parameter. This regularization was inversely proportional to *C*. The value of *C* should always be positive. The gamma indicates the kernel coefficient for the type of kernel that is to be used in the algorithm. The kernel specifies the type of kernel to be used in the algorithm. For this work, the optimal parameters found for the SVM model implementation were a *C* value of 100, gamma of 0.001 and an rbf kernel. The implementation of these parameters upon the SVM classifier could improve the classification accuracy score by approximately 10%. However, this improvement remained lower than the results obtained from the Logistic Regression.

3) Logistic Regression: In the case of the Logistic Regression model, the hyper-parameters considered are the *C* value and penalty. In the SVM model, the *C* value must be a positive number, which is a regularization parameter. The penalties considered are L1 and L2. The L1 performs a regularization with the squared magnitude as the penalty. In the case of the L2, the absolute value of magnitude is the one considered as the penalty of the model. For the project, the *GridSearchCV* function returned a *C* value of 0.1 with an L2 penalty as the best parameter for the implemented Logistic Regression. The implementation of the model with these parameters did not provide a significant improvement on the obtained results of the first evaluation of the model.

4) Random Forest: In the case of the ensemble model of Random Forest, the parameters that can be tuned include the split criterion, number of estimators, maximum features that include auto, square root and log2, and max depth of the trees. The optimal parameter search executed this work indicates

that the best parameters for this model are the split criterion of entropy, along with a max depth of 6 levels, max feature of log2 and 200 estimators (trees). Like the previous cases, the obtained results of the implemented model with these parameters could not improve upon the accuracy score that was previously obtained.

5) XGBoost: For the XGB classifier, there are several parameters that can be explored to determine the optimal parameter of the model. The parameters that can be determined using the *GridSearchCV* function include the number of threads, objective function, learning rate, max depth of trees, minimum child weight and number of estimators among others. For this work, only the parameters of max depth, learning rate and number of estimators were considered. The optimal parameters returned by the function for this model were a learning rate of 0.5, max depth of 4 and 100 estimators. The results obtained through this parameter could not significantly improve on the accuracy score that was previously obtained.

6) Gradient Boosting Algorithm: Given that the XGB algorithm is an improvement on GBM, the GBM algorithm parameters are similar to those of the XGB model. For this work, the parameters that were taken into consideration for the grid search were the number of estimators, learning rate and max depth. The implementation of the grid search function determines that the optimal parameters for the data are a learning rate of 0.01, max depth of 4 and 200 estimators. Similar to the previous models, the accuracy score from the implemented model with these parameters could not improve on the accuracy score that was previously obtained.

III. Results

From the previous results, the best model performance was obtained using the Linear Regression classifier with a C value of 0.1, which could achieve a mean accuracy score of 80.30% and a mean AUC score of 0.76. The feature importance of was also examined through an Extra Trees Classifier. The implementation of the algorithm returned valuable insight into the importance of the presented attributes for the process of loan eligibility. For instance, the results of the algorithm showed that the most important feature is the credit history of the applicant. Additionally, it was found that the amount of loan requested and the income from the applicant has a significant weight upon the eligibility for a home loan.

IV. Discussion

Given the observed results from the data exploration, some recommendations can be made to DHF to improve their service:

1) Given that the main form of interaction with the customers is expected to be an online portal, before moving to the actual online form for a loan application, suggestions to improve the chances to be eligible for a loan could be made. The company could provide a small guide for the customers upon what aspects are considered the most important before actually applying for a loan.
2) According to what was observed through this work, a good credit history greatly increases the chances of being eligible for a loan. Therefore on the provided guide, emphasis on the importance of this attribute should be made. Additionally, general methods to improve their credit history could be included to serve as an aid for potential applicants and future steps for them to take.
3) Another aspect to emphasize for the future applicants are their income and co-applicant income. Although higher incomes does not necessarily mean the loan is to be approved, if the co-applicant income can complement the applicant's income, it might help upon the decision of loan approval, as well as serve as a basis for the applicants to understand their finances.
4) Most applicants appear to be married, do not have dependents, are male, graduates and are not self-employed. Upon the consultation for a loan, indicating that these factors do not heavily weight upon their eligibility might help increase the number of applicants. This last strategy could be implemented through a parallel marketing study to better understand the characteristics of the applicants.

V. Conclusion and Future Work

Through this project, the loan eligibility process was studied. An automation for the loan approval process was proposed through a function that cleans the data from the most common missing values observed in the provided historical data and performs a prediction according to the information that is contained in the data. The proposed model could not achieve an accuracy performance higher than 90%, but the results obtained from this model might help improve upon the company's process of approving or denying a loan. The work allowed us to test methods and techniques of data science and machine learning upon the development of a project with the CRISP-DM methodology. Although the models could not achieve an accuracy performance as expected, the development of the project allowed us to extract valuable insights that might be incorporated into the company's operations to provide better service. Due to the lack of expertise and time, there were some factors that could not be implemented, such as feature engineering. Perhaps engineering features such as the loan amount and loan term ratio, or total income by adding the applicant's income with the co-applicant income might have proved to be valuable features for the predictive model. Additionally, a more exhaustive hyper-parameter tuning might have helped improve upon the performance of the proposed models.

References

[1] Foster Provost and Tom Fawcett. *Data Science for Business: What you need to know about data mining and data-analytic thinking*. " O'Reilly Media, Inc.", 2013.