An Investigation on Vietnamese Credit Scoring based on Big Data Platform and Ensemble Learning

Quang-Linh Tran[1,*], Van-Binh Duong[1,*], Gia-Huy Lam[1,*], Trong-Hop Do[1,†]

University of Information Technology
Vietnam National University Ho Chi Minh City
[*] *{18520997, 18520505, 18520832}@gm.uit.edu.vn,*
[†] *hopdt@uit.edu.vn*

Abstract. Credit score is an important indicator which can affect many aspects of people's lives. However, evaluating credit score is done manually so it costs a large amount of money and time. This paper introduces a novel and automatic way to evaluate customer credit score by using big data platform and some ensemble learning methods. Since data related to financial activities of customers grows enormously, a big data platform to handle this amount of data is necessary. In this paper, Spark which is a distributed, data processing framework, is used to save and process data. Some experiments are carried out to compare the effectiveness of feature engineering in this problem. Moreover, a comparative study about performance of ensemble learning models is also given in this paper. A real-world Vietnamese credit scoring data set is used to develop and evaluate models. Four metrics are used to evaluate the performance of credit scoring models, namely F1-score, recall, precision and accuracy. The results are promising with the highest accuracy of 72.9% in the combination Gradient-boosted Tree and cleaned data set with removing categorical features. This paper is a foundation for using big data platform to handle financial data and many future research can carried out to optimize the performance of this paper.

Keywords: Credit scoring · Big data · Ensemble Learning · Machine Learning · Feature Engineering.

1 Introduction

Credit score plays an important role in people's lives nowadays. When anyone want to open a credit card or loan an amount of money, credit institutions will evaluate credit score of this customer to ensure that he or she can return the loan. In addition, the amount of money that credit card holders can spend also depends on their credit score. This score is essential for everyone because if they are assessed that having a low credit score, they cannot loan money to buy a car or a house. This is the reason why credit score is significant for people today. Although Vietnamese banking are a young system which have developed in some ten years, the need for an automatic credit scoring system is essential. The population of Vietnam is over one hundred million and the working-age population is over 60 million people. These people create a lot of data related to financial activities such as purchasing a house or loaning a large amount of money. This amount of data is valuable for artificial intelligence to learn and predict the credit score of these people. The amount of data from them are enormous and it increases continuously. When we use this amount of data to build a machine learning model to evaluate credit score, a huge revolution will happen in the banks sector. To be specific, many experts about credit scoring will lose their job, but the effectiveness and fairness of this credit scoring system is undeniable. This means that people have a trustable, fair and efficient credit scoring system, giving an accurate credit score to everyone.

In many banks nowadays, assessing credit scoring procedure depends heavily on manual ways. A large number of people is hired to evaluate credit score of customers of banks. However, this way shows many disadvantages. Firstly, banks need a lot of people and time to manually evaluate someone's credit score. When an expert receives data about a customer, they need time to read over data and give the score, which is intensively time-consuming. Secondly, these experts can give bias for their acquaintances, and this can result in many serious problems. Therefore, a machine learning model which evaluate credit score of anyone can replace humans in this task. Its effectiveness is undeniable from some previous research [10].

With the development of information technology in general and artificial intelligence in particular, many real-world problems have been resolved by computers. Artificial intelligence is applied to all of aspects of life, from education to health. Banking is not an exception with a lot of works now can be done by computers such as credit card fraud detection. Credit scoring is a potential area that computers can work well thanks to some state-of-the-art machine learning algorithms. These algorithms can learn valuable insights that hide on customers' data and give an accurate result, which can surpass human performance in someday. In this paper, some ensemble learning models will be used to evaluate whether a credit score of a customer is qualified.

In this paper, we carry out several experiments related to many techniques to get the highest result in building credit scoring models. We will mainly focus on feature engineering, which is important in data mining because data related to customers' financial activities is uncleaned. The improvement in applying feature engineering is significant, giving a huge improvement in models performance. In addition, many ensemble learning models can used for this problem. We will compare the performances of these models in evaluating credit score.

In the following of this paper, some related works are shown on Section 2. Section 3 illustrates some techniques that we use in to experiment and information about ensemble learning. The experiments and results are shown in Section 4. In Section 5, we give the conclusion about this paper and some future works.

2 Related Work

This subject has drawn a lot of attention from scientists and here are some previous research that have been carried out in credit scoring evaluation.

In a paper named Machine Learning-Based Empirical Investigation For Credit Scoring In Vietnam's Banking [12], Quoc-Khanh Tran and his colleagues did some experiments about using machine learning to investigate credit score in Vietnam's banking. He used a real-world data set called the Kalapa Credit Score data set and the results are remarkably. The highest result is 83% F1-score with Random Forest algorithm. Although the results are relatively high, there is still some aspects that we can improve such as applying feature engineering or using big data platform to handle data. We will use the same data set as this research to experiment.

An approach proving the effectiveness in resolving credit scoring problem is ensemble learning. Some ensemble learning methods such as bagging, boosting, and stacking are used to evaluate credit score [13] with the best model is 80.76% accuracy in the research of GangWang et al. In addition, Tounsi and his colleagues [10] prove that Boosting and Credal Decision Tree (CDT) are better than other ensemble algorithms.

With the explosion of data, applying big data platform for credit scoring problem also carried out by Tousil et al. [11]. They use social data instead of traditional financial data to evaluate credit score and some survey on proposed methods are also given to address this problem. Apache Hadoop and Apache Spark are considered to use but there are no experimental results in their research.

From the advantages and disadvantages of previous research, we will continue to investigate some untouched part of this problem to give a better performance in credit scoring.

3 Methodologies

In this section, some pre-processing and feature engineering techniques are illustrated. In addition, information about big data platform and some ensemble learning methods are also given in this section.

3.1 Pre-processing techniques

The raw data set is real world data, so it tends to be incomplete, noisy and inconsistent [2]. Predictive models should be built on qualified, clean data. Therefore, the step of data pre-processing is significantly important. In order to process the data, in this paper, some pre-processing techniques are applied to clean and normalize the data. In addition, feature engineering is also used for extracting the valuable information in data as much as possible. There are four methods explained in [4] that are chosen to prepare the data set. They are Data cleaning, Data normalization, Data transformation and Missing values imputation.

Data cleaning is the process of removing and modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted. For instance, the format of date and datetime data in a column is not correct, so it should be corrected to an only one format. Moreover, attributes that have value missing rate more than 90% should be remove from the data set.

Data normalization is a technique that structures the data for storage efficiently. The data is collected from many resources; hence, the range of numeric values is different between the attributes. This may lead to the poor-quality performance of the models built on the data. Subsequently, the data have to be normalized to the same distribution. In the categorical feature, if a large information is stored in just one feature, it is better to split the data into specific features. So that, the data is managed logically, and the storage space is used economically.

Data transformation[1] is the process of changing the format, structure, or values of data. The transformed data is better-organized and easier to use than the raw data. Properly formatted and validated data improves data quality and protects applications from potential landmines such as *null* values, unexpected duplicates, incorrect indexing, and incompatible formats. Data transformation facilitates compatibility between applications, systems, and types of data. Data used for multiple purposes may need to be transformed in different ways.

Missing values imputation is simply the process of filling missing data. But it is not easy to find out an appropriate value. Data imputation takes a lot of effort to gain 'just' the acceptable result. There are many techniques[2] to impute the data as: using mean, median value or taking the predicted value from a trained model, etc.

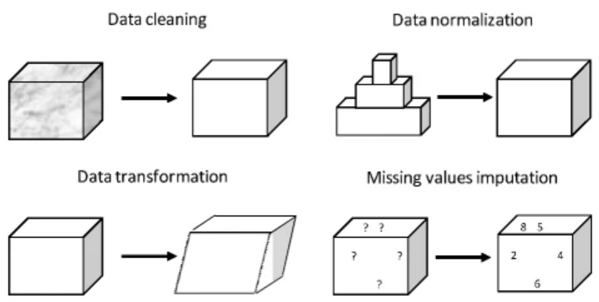


Fig. 1: Data pre-processing techniques [4].

[1] https://www.stitchdata.com/resources/data-transformation/
[2] https://scikit-learn.org/stable/modules/impute.html

3.2 Feature engineering

Feature engineering [7] is a terminology for the progress of preparing the proper input data set that is compatible with the machine learning algorithm requirements and improving the performance of models. There is a sea of techniques for engineering the data. Since the specific dataset used in this paper, the techniques such as: One-hot encoding, Date extraction are applied to observe the performance of the trained models.

One-hot encoding is one of the most common encoding methods in machine learning. This method spreads the values in a column to multiple flag columns and assigns 0 or 1 to them. These binary values express the relationship between grouped and encoded column. Subsequently, this also changes the categorical data to numerical format which is easy for the algorithms to understand.

Date extraction is a method to manipulate the date format data which is hard for algorithms to understand. There are three basic types of pre-processing as: extracting date into parts, extracting time period between the current date and extracting specific features (holiday, weekend).

3.3 Big data platform

In the era of big data, data is everywhere. Thanks to the development of technological devices, data can be collected from social media, sensors, IoT devices which make the amount of data become enormous that traditional tools and technique hardly handle. Big data has five specific characteristics, which are volume, variety, value, velocity and veracity. Big data has many applications ranging from banking and securities to communication or education, but how to utilize big data is a problem that traditional tools and techniques cannot process. This is the reason why some frameworks are born to deal with big data. One of the powerful frameworks is Apache Spark which we use for credit scoring problem in this paper.

Apache Spark [15] initially started in 2009 and it became a part of Apache Software Foundation in 2013. Apache Spark is an open-source cluster computing framework for real-time processing. Spark can be run in many clusters, which utilize the hardware of many devices for resolving a problem. Spark provides an interface for programming entire clusters with implicit data parallelism and fault-tolerant [14]. Thanks to saving data in multi clusters, when a cluster is broken, data can be retrieved from other clusters so it provide an efficient fault-tolerant. Spark has some important components that supply users all necessary tools from collecting data from streaming or other sources to processing data and modeling data. Spark Core is the base engine for large-scale parallel and distributed data processing. Further, additional libraries which are built atop the core allow diverse workloads for streaming, SQL, and machine learning. Spark Streaming helps to process real-time data from various sources such as Kafka, Flume, and Amazon Kinesis. After processing, this data can be pushed out to file systems or databases. Spark Streaming is used for real-time data which can be unstructured data such as images or text and Spark also provide a tool for dealing with structured data, which is Spark SQL [1]. In modeling, Spark has MLlib [6], a machine learning framework for Spark. This framework provides a lot of choice for building predictive model from regression algorithms to classification algorithms. Figure 2 below illustrate the components and structure of Spark.

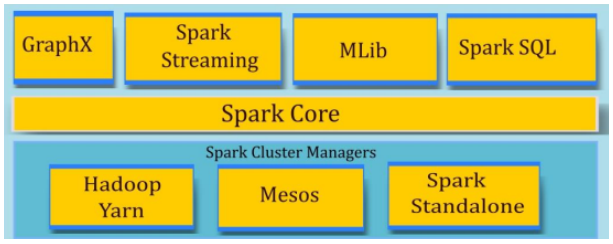


Fig. 2: Spark Components

3.4 Ensemble Learning

Ensemble learning is kind of machine learning that ensemble many machine learning models to get the final result. In this subsection, we will introduce a three tree-based models with 2 ensemble learning methods.

Decision Tree [8] is a typical tree-based model, which is constructed by many nodes. Decision tree can be used in both classification and regression problem. It has a root and many leaves and decision tree model bases on the rules in these leaves to give prediction. Decision tree use Gini index or Entropy to split leaves. The Gini index is computed as:

$$Gini = -\sum_{i=1}^{k} Pi(1 - Pi), \tag{1}$$

The Entropy is computed as follow:

$$Entropy = -\sum_{i=1}^{k} Pi \log_2 Pi, \tag{2}$$

The k is the number of cases and Pi is the probability of case i. Figure 3 below is an example of decision tree using entropy to build the tree:

Decision tree is the foundation for Random Forest algorithms and other ensemble learning methods.

Random Forest [5] is a combination of decision trees so it is an ensemble learning method. When a decision tree gives a result, this result will be aggregated with other trees to give the final result. Random Forest helps to reduce the

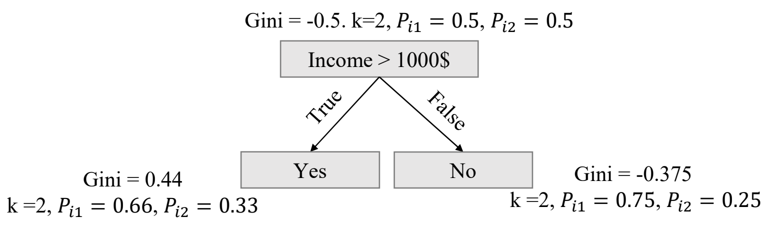

Gini = -0.5. k=2, $P_{i1} = 0.5$, $P_{i2} = 0.5$
Income > 1000$
True
False
Yes
No
Gini = 0.44
k =2, $P_{i1} = 0.66$, $P_{i2} = 0.33$
Gini = -0.375
k =2, $P_{i1} = 0.75$, $P_{i2} = 0.25$

Fig. 3: Example of Decision Tree using Gini.

overfitting problem in decision tree because it require many trees to give the result. Random Forest is an effective ensemble learning method thanks to its mechanism of trees combination. The Fig 4 below is an example of how a random forest work.

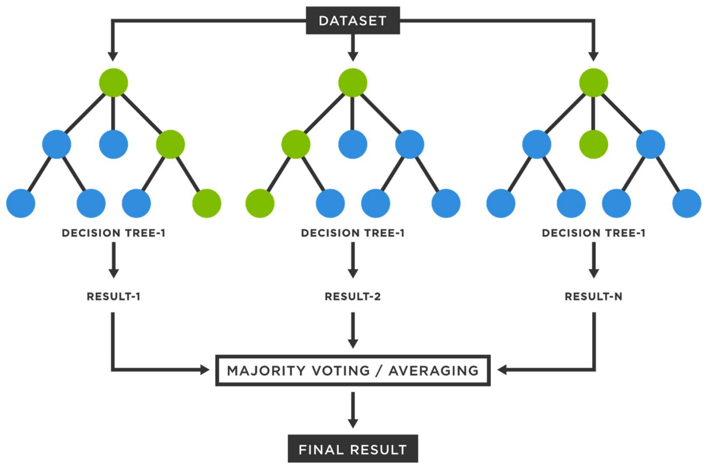


Fig. 4: Example of Random Forest.

Gradient-boosted Tree [3] is similar to Random Forest. It also uses a set of decision trees to predict the target label. However, Gradient-boosted tree usually outperforms Random Forest because Random forests use bagging mechanism to build independent decision trees and combine them in parallel. On the other hand, Gradient-boosted Tree uses an algorithm called boosting [9]. It means that Gradient-boosted Tree builds one tree at a time. This additive model works in a forward stage-wise manner to improve the gradient of existing weak learners (tree) [3], sequentially, each new tree corrects the errors of the previous one. In

[3] http://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf

order to evaluate the performance of a tree, the loss function is used. Here is the cross-entropy loss function:

$$L(\hat{y}, y) = -\sum_i y_i \log(\hat{y}), \tag{3}$$

After having the loss of the first tree, the second tree is added so as to lower the loss compared to the first one alone. The loss has to move in direction of lowering its value fastest and the current tree's output is enhanced by that of the previous one as we can see in the Fig 5 below. Mathematically, this is given by using the negative derivative of loss with respect to the previous tree's output. This is the reason why this is called Gradient-boosted tree.

$$F(k) = F(k-1) + \eta \times -\frac{\partial L}{\partial F(k-1)} \tag{4}$$

where, k annotates the current tree $(0 < k \leq N)$, η is the learning rate.

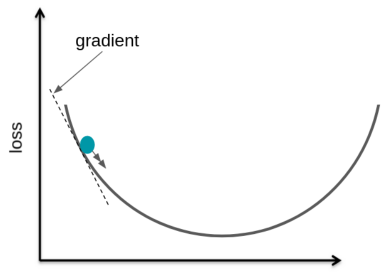


Fig. 5: Gradient descent.

3.5 Important Features Extraction

Important Features Extraction is one of the useful techniques to increase the performance of model. In this paper, important features are extracted from the ensemble learning models on Spark. Some tree-based model calculates the node/feature probability to find out the important features from data set. Normally, the node probability is calculated by the number of samples that reach the node, divided by the total number of samples. When a feature gets a high probability, it has an enormous importance to the model.

Tree-based models extract calculation the importance indicator of a feature by summing the gain, scaled by the number of samples passing through the node. To be more specific, the important feature extraction formula is as in the equation 5 below:

$$f_i = \sum_j^k s_j C_j, \tag{5}$$

- f_i: the probability of feature i
- s_j: number of samples reaching node j
- C_j: the impurity value of node j
- k: nodes j splits on features i

4 Experiment

4.1 Data set and Pre-processing

The raw data set consists of 193 attributes but there are 117 attributes having missing rate over 50%. Due to the encrypted attribute names, it is hard to understand the relationship between attributes and their values. The attributes with missing rate grater or equal than 90% are removed. Besides, the complication in datatype need paying attention on. Therefore, so many investigations have been done to find out the best way to deal with each feature. Some pro-processing techniques and feature engineering methods mentioned in Section 3 is applied to clean the data set.

The attributes are divided into three groups (based on the datatype and empirical investigation). They are group 1: date and datetime attributes, group 2: unicode ones and group 3: the remaining features. There are 28 attributes in group 1 (e.g: *Field_ 1, ngaySinh, A_ startDate, A_ endDate, etc.*). All the values in each one in this group need normalizing to date or datetime form. The group 2 including 30 attributes (e.g: *Field_ 18, maCv, homeTownCity, brief, etc.*). Values in this group are Vietnamese or English words, sentences, and abbreviations. The concern is the inconsistent expression of the same meaning words or sentences. So, the solution is to lower all characters and remove the unnecessary characters or words from the values (e.g: *'thủ đô Hà Nội'* or *'tp. Hà nội'* $\rightarrow$ *'hà nội'*, etc.). The inconsistency of values in each attribute are also normalized (eg: *'I'* $\rightarrow$ 1, *'II'* $\rightarrow$ 2, *'Ngoài quốc doanh Quận 7'* $\rightarrow$ *Null*, etc.), then the datatype normalization may be applied if needed. In order to deal with group 3, the numeric datatype correction is used because the attributes in this group mostly have numeric values. The attributes with categorical values are also processed as those in group 2.

The missing values, in categorical and numeric attributes, are treated separately. The missing values in categorical feature are fulfilled with *'unknown'* value and a very large number is used for imputing missing values in numeric features. Those imputing values are chosen because of their balanced meaning for the real data. By using this method, it is hopefully believed that the data set is fulfilled and the data imputation does not impact too much to the original data.

4.2 Feature Engineering

After the data set cleaned, it is believed that the information of each attribute can be made use of as much as possible by generating new attributes. So that, some new attributes are generated based on the investigations on the data set. The phase is divided into two separated phases. Phase 1, the attributes in group 1 and date datatype attributes in group 3 are processed: the two new features *age* and *birth_ month* are generated by extracting information from the feature *ngaySinh* (in which, $age = 2021-$ year of *ngaySinh*), the features in form of *x_ startDate* and *x_ endDate* are used to generate new features in form of *x_ start_ end* by subtracting *x_ startDate* to *x_ endDate*. By using the same method, the features *x_ y_ startDate* and *x_ y_ endDate* are created, where x and y in list of *[A, C, E, F, G]*. Additionally, some date attributes may be the beginning date of a debt, so it is good to calculate the duration of that debt by subtracting the current processing date to the attribute's values. With other attributes, the time period between two features, in day, is stored in a new feature. Some new features are generated by checking whether the date values in the attribute are valid. Also, some new features (*is_ WD_ n*) which is created by checking whether the date (in *Field_ n*) is weekend or not. In phase 2, it is the turn for group 2 attributes and the features of group 3: the feature *gender* is born from the combination of *ngaySinh* and *info_ social_ sex*, two new features are created by slicing the two or three letters from the values of *Field_ 45* because this is may be the code that the bank assigned to the customer. Finally, there are 81 new features added and 47 ones deleted. So, after the feature engineering, there are 227 attributes in the processed data set.

4.3 Input preparation

In order to train the models, the input must be compatible to the algorithm to understand. Therefore, the values in features must be converted to a single vector so to implement the conversion, the MLlib[6] in Spark is used with estimators such as StringIndexer, OneHotEncoder, StandardScaler, VectorAssembler. There are different conversion methods for each data type. There are two types of data that need to be converted: category and numeric. The categorical data group includes data in the form of string, boolean and some other properties of ordinal (Field_82 and birth_month). For this group, StringIndexer is used to mark index each value in each attribute depending on the number of unique values, then we use OneHotEncoder to convert the index into a one hot vector(1). For the numerical data group, which includes all the remaining attributes. To transform this type of data, Skew and Kurtosis coefficients are used to filter out the attributes that need to be normalized according to the normal distribution. For attributes that do not need to be normalized, VectorAssembler aggregates them together with the vector of the categorical attribute group into a single vector (2). StandardScaler applies on the attributes to be normalized, then aggregates the conversion to vector(3) using VectorAssembler. Finally, executing the VectorAssembler again for (1),(2),(3) to aggregate into a large vector to feed into the model. Figure 6 below shows the progress of the vector conversion.

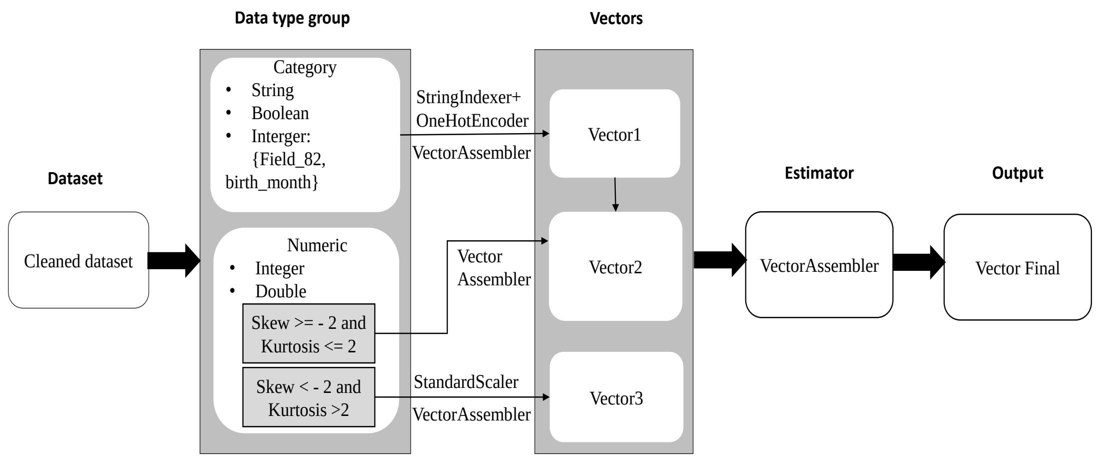


Fig. 6: Transformation pipeline

4.4 Experimental Procedure

This paper aims to investigate which feature engineering method and ensemble learning method are the best so we design the experimental procedure as the following structure. From the Kalapa Credit Score data set, we do some initial pre-processing such as handling missing values and normalizing numeric features in Spark framework. After that, we divide into 2 data sets, in which one data set is remained categorical features and denoted as Scenario A and another data set is removed categorical features and denoted as Scenario B. We apply 2 phases of feature engineering into these two data sets to construct new data sets and then we obtain four new processed data sets corresponding to A and B above, and two original data sets. We use three ensemble learning methods as we said in 3.4 to build the predictive model. Then we use some features which are computed as important of Gradient-boosted Tree in the data set 2 to obtain a two new data sets important features. We retrain these two data sets with three models and finally we compare the results of different combination between data sets and ensemble learning methods. In summary, we have 2 main branches, data having raw categorical features and data removing raw categorical features, demoting as Scenario A and Scenario B, respectively. In each branch we have four data sets, namely original data set, data set after phase 1, data set after phase 2 and data important features. We use three models to evaluate and compare the results.

Note: Data important features is the data set belonging to the best case of extracting important attributes. A total of six cases are set with the condition that the probability of feature is greater than a threshold - multiple of the probability at which each attribute has the same probability.

$$\begin{aligned} threshold &= \tfrac{X}{Y} \\ X &\in \{0,1,2,3,4,5\} \end{aligned} \tag{6}$$

- threshold: probability corresponding to x (threshold)
- Y: number of important attributes extracted from the model

Table 1: Experimental data set statistics

Data set	Num of features in Scenario A	Num of features in Scenario B
Data set1: Original	117	91
Data set2: data set 1 + feature engineering for datetime features	184	158
Data set3: data set 2 + feature engineering for categorical features	196	168
Data set important features	23	20

4.5 Model Hyperparameter

We use MLlib [6] in Spark to build and evaluate models. In the Decision Tree model we use the set of hyperparameter as follow: maxDepth=5, maxBins=32, minInstancesPerNode=1, minInfoGain=0.0, maxMemoryInMB=256, cacheNodeIds=False, checkpointInterval=10, impurity='gini', seed=42.

In the Random Forest, we use 20 trees to build the random forest classifier and the maxDepth is 5, other hyperparameters are as follow: maxBins=32, minInstancesPerNode=1, minInfoGain=0.0, maxMemoryInMB=256, cacheNodeIds=False, checkpointInterval=10, impurity='gini', featureSubsetStrategy= 'auto', seed=42, subsamplingRate=1.0.

In the Gradient-boosted Tree, we using lossType is logistic, the maxDepth is 5, the number of max iteration is 20 and, and other hyperparameters are as follow: maxBins=32, minInstancesPerNode=1, minInfoGain=0.0, maxMemoryInMB=256, cacheNodeIds=False, checkpointInterval=10, stepSize=0.1, seed=42, subsamplingRate=1.0, impurity='variance', featureSubsetStrategy='all', validationTol=0.01.

4.6 Results and Discussion

Table 2: The result of models on data sets of Scenario A

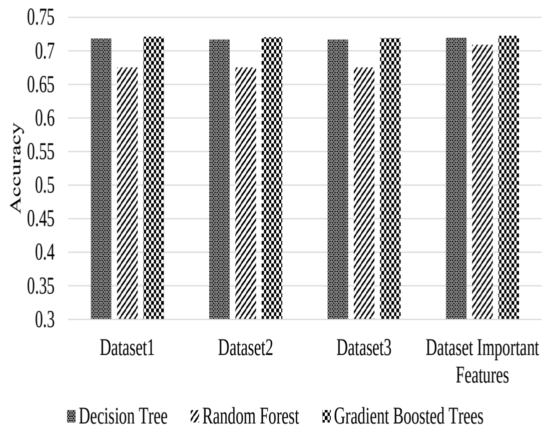
Dataset	Model	Accuracy	Recall	Precision	F1 score
Data set1	Decision Tree	0.7183	0.5935	0.7068	0.5870
	Random Forest	0.6754	0.5000	0.3377	0.4031
	Gradient Boosted Tree	0.7210	0.5979	0.7114	0.5931
Data set2	Decision Tree	0.7170	0.6029	0.6888	0.6025
	Random Forest	0.6754	0.5000	0.3377	0.4031
	Gradient-boosted Tree	0.7203	0.6020	0.7017	0.5999
Data set3	Decision Tree	0.7170	0.6029	0.6888	0.6025
	Random Forest	0.6754	0.5000	0.3377	0.4031
	Gradient-boosted Tree	0.7192	0.6010	0.6988	0.5988

Table 3: The result of models on data sets of Scenario B

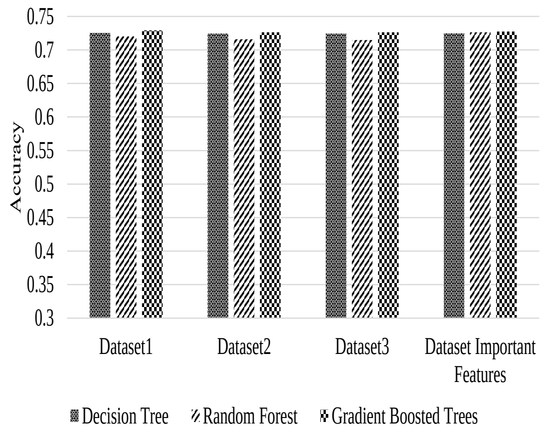
Data set	Model	Accuracy	Recall	Precision	F1 score
Data set1	Decision Tree	0.7254	0.5930	0.7028	0.5891
	Random Forest	0.7202	0.5791	0.7021	0.5677
	Gradient Boosted Tree	0.7292	0.6008	0.7075	0.6001
Data set2	Decision Tree	0.7244	0.5987	0.6921	0.5984
	Random Forest	0.7158	0.5685	0.6997	0.5504
	Gradient Boosted Tree	0.7263	0.6004	0.6968	0.6004
Data set3	Decision Tree	0.7244	0.5987	0.6921	0.5984
	Random Forest	0.7147	0.5575	0.7271	0.5278
	Gradient Boosted Tree	0.7263	0.6004	0.6968	0.6004

As we can see on the table 2 and 3, the overall performance of these models in classifying credit scoring status of customers is around 70% accuracy and 60% f1-score. In general, Gradient-boosted Tree gives the best performance with 72.1% in the data set 1a and 72.92% accuracy in the data set 1b. This is easily understood because the boosting algorithm in Gradient-boosted Tree is very effective in improving the performance. By contrast, Random Forest shows the worst results in all data sets.

The data set 1, which is the original data set give the better results than other data sets. The best results in data set 1a is 72.1% accuracy. This proves that the original data set provides sufficiently data for the model. In contrast, adding more features is not sure to make the performance increasing. However, adding also do not decrease the results much, it is only a small decrease.

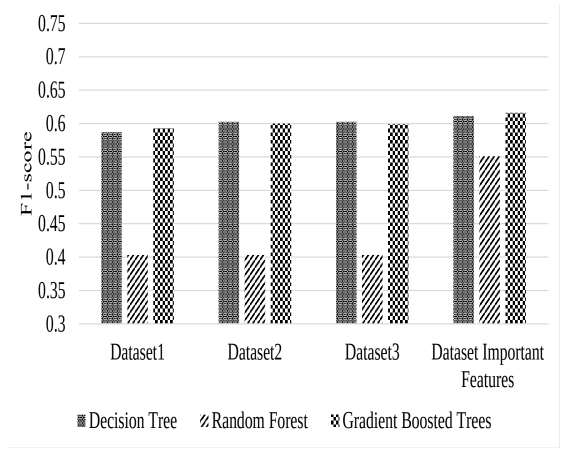


(a) Scenario A

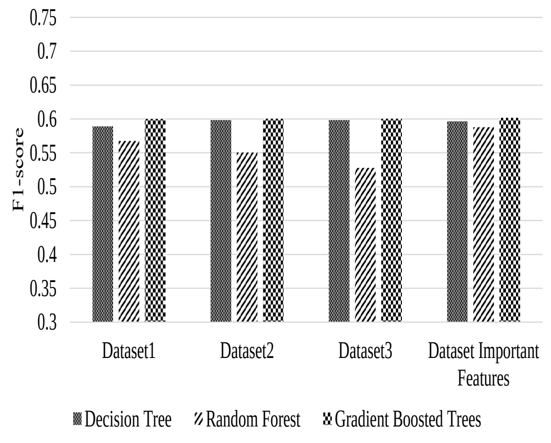


(b) Scenario B

Fig. 7: Accuracy of different models on 4 data sets



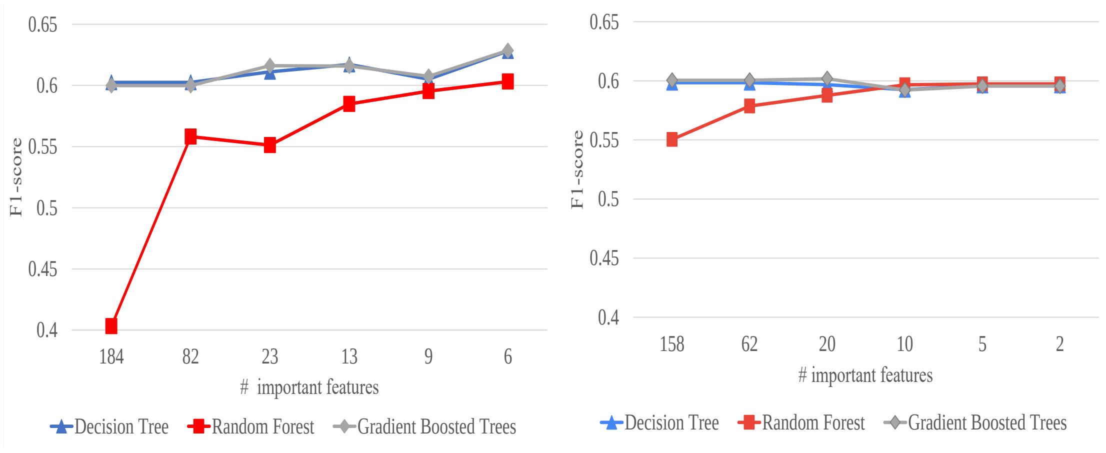
(a) Scenario A



(b) Scenario B

Fig. 8: F1-score of different models on 4 data sets

As we can see in the Fig. 7 and 8, the Scenario B, which is the data set deleting raw categorical features shows the higher results than the Scenario A. To be specific, the best result in Scenario B is 72.9% accuracy while the best result in Scenario A is only 72.1%. In addition, the performance of Random Forest in Scenario B is better than that in the Scenario A. This is a huge improvement because the f1-score of Random Forest in Scenario A is about 40% while the f1-score of Random Forest in Scenario B is always over 50%.



(a) Scenario A (b) Scenario B

Fig. 9: F1-score on different numbers of important features in data set 2

In the Fig. 9, we show the performances of three models on the data set 2 with many number of important features. We decrease the number of features from all features to the most important features. In Scenario A, there is 184 features in data set 2 and we use the formula in 3.5 to calculate the important of each feature and we select a set of important feature to train model. The data for Scenario B is similar. When we decrease the number of feature in Scenario A, the f1-score increase in all models, especially in Random Forest, from approximately 40% to about 60%. However, the performances of Decision Tree and Gradient-boosted Tree are fell in Scenario B.

5 Conclusion and Future Work

In this paper, we compare the performances of different feature engineering techniques and various ensemble learning methods for credit scoring problem. We use Spark framework, which shows the effectiveness on handling big data. We use a real world data set about Vietnamese customers and the results on this data set are remarkable. Gradient-Boosted tree shows the highest performance at about 0.6% f1-score and 72.92% accuracy. In addition, removing raw categorical features gives a higher result than retaining them. Selecting important features is also an efficient way that improving the model performance that has been proving in this paper.

We have conduct several experiment to improve the performance in classifying customer's credit score status, but many aspects can research in the future. Using streaming data about customers, especially real-time financial activities or social data is an useful way to boost model's performance. Deep learning models proves their robustness in resolving many problems so apply deep learning models also should be considered to improving the performance.

References

[1] Michael Armbrust et al. "Spark SQL: Relational Data Processing in Spark". In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. SIGMOD '15. Melbourne, Victoria, Australia: Association for Computing Machinery, 2015, pp. 1383–1394. ISBN: 9781450327589. DOI: `10.1145/2723372.2742797`. URL: `https://doi.org/10.1145/2723372.2742797`.

[2] Mirela Danubianu. "Step by step data preprocessing for data mining. A case study". In: *Proc. Of the International Conference on Information Technologies (InfoTech-2015)*. 2015, pp. 117–124.

[3] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. "Special invited paper. additive logistic regression: A statistical view of boosting". In: *Annals of statistics* (2000), pp. 337–374.

[4] Salvador Garcıa et al. "Big data preprocessing: methods and prospects". In: *Big Data Analytics* 1.1 (2016), pp. 1–22.

[5] Andy Liaw, Matthew Wiener, et al. "Classification and regression by randomForest". In: *R news* 2.3 (2002), pp. 18–22.

[6] Xiangrui Meng et al. *MLlib: Machine Learning in Apache Spark*. 2015. arXiv: `1505.06807 [cs.LG]`.

[7] Fatemeh Nargesian et al. "Learning Feature Engineering for Classification." In: *Ijcai*. 2017, pp. 2529–2535.

[8] Lior Rokach and Oded Maimon. "Decision Trees". In: vol. 6. Jan. 2005, pp. 165–192. DOI: `10.1007/0-387-25465-X_9`.

[9] Robert E. Schapire. "A Brief Introduction to Boosting". In: *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*. IJCAI'99. Stockholm, Sweden: Morgan Kaufmann Publishers Inc., 1999, pp. 1401–1406.

[10] Youssef Tounsi, Larbi Hassouni, and Houda Anoun. "An Enhanced Comparative Assessment of Ensemble Learning for Credit Scoring". In: *Journal of Intelligent Computing* 10 (Mar. 2019), p. 15. DOI: `10.6025/jic/2019/10/1/15-33`.

[11] Youssef Tounsi, Larbi Hassouni, and Houda Anoun. "Credit scoring in the age of Big Data -A State-of-the-Art". In: *International Journal of Computer Science and Information Security,* 15 (July 2017).

[12] Khanh Quoc Tran et al. "Machine Learning-Based Empirical Investigation for Credit Scoring in Vietnam's Banking". In: *Advances and Trends in Artificial Intelligence. From Theory to Practice*. Ed. by Hamido Fujita

et al. Cham: Springer International Publishing, 2021, pp. 564–574. ISBN: 978-3-030-79463-7.

[13] Gang Wang et al. “A comparative assessment of ensemble learning for credit scoring”. In: *Expert Systems with Applications* 38.1 (2011), pp. 223–230. ISSN: 0957-4174. DOI: `https://doi.org/10.1016/j.eswa.2010.06.048`. URL: `https://www.sciencedirect.com/science/article/pii/S095741741000552X`.

[14] Matei Zaharia et al. “Resilient Distributed Datasets: A Fault-Tolerant Abstraction for in-Memory Cluster Computing”. In: *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*. NSDI’12. San Jose, CA: USENIX Association, 2012, p. 2.

[15] Matei Zaharia et al. “Spark: Cluster Computing with Working Sets”. In: *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*. HotCloud’10. Boston, MA: USENIX Association, 2010, p. 10.