# Comparison between different models in Credit Score Classification

1st Dat Nguyen Duy
*University of Information and Technology*
*Vietnam National University HCMC*
20520435@gm.uit.edu.vn

2nd Minh Le Doan Phuc
*University of Information and Technology*
*Vietnam National University HCMC*
20520243@gm.uit.edu.vn

3th Truong Pham Nguyen Xuan
*University of Information and Technology*
*Vietnam National University HCMC*
20520835@gm.uit.edu.vn

4th Huy Le Nhat
*University of Information and Technology*
*Vietnam National University HCMC*
20520056@gm.uit.edu.vn

5th Nam Le Nguyen Khanh
*University of Information and Technology*
*Vietnam National University HCMC*
20520073@gm.uit.edu.vn

6th Binh Phan Doan Thai
*University of Information and Technology*
*Vietnam National University HCMC*
20520043@gm.uit.edu.vn

***Abstract*—Gradient Boosting, Random Decision Forest, k-Nearest Neighbors (KNN), Perceptron Learning Algorithm (PLA or Perceptron for short), Support Vector Machine (SVM) and Naive Bayes Classifiers are some of common machine learning algorithms for classification. This paper aims to use these algorithms to predict credit score for comparison between these algorithms. The Credit Score Classification dataset from Kaggle will be used in this paper. This dataset consists 28 data fields (including result field) and 150000 rows of data from both train and test dataset, containing basic bank details and gathered a lot of credit-related information.**

***Index Terms*—machine learning, support vector machine, naive bayes, perceptron, random forest, gradient boosting, k-nearest neighbors, classification**

## I. Introduction

A credit score is a numerical expression based on a level analysis of a person's credit files, to represent the creditworthiness of an individual [8]. Each of credit score number ranges represent the creditworthiness class of an individual such as Poor, Standard, Good.

Credit Score have been an important factor for many organizations. Thanks to credit score, businesses can make more precise decision for their customers such as smarter products recommendation, better business strategy based on market scenario to make more profit from it. Many banks also use this data to predict the chance of the specific applicant to repay loan, allowing them to make decision regarding account creation which helps these banks avoid the risks of their applicants not paying them back the loan and mitigate losses due to bad debt.

Credit scoring is not limited to these situations only. Other organizations, such as mobile phone companies, insurance companies, landlords, and government departments employ the same techniques. Digital finance companies such as online lenders also use alternative data sources to calculate the creditworthiness of borrowers [8].

Because of this, it's important to predict the credit score. In the Information Age, many of machine learning models have been created for classification jobs recently. In this paper, we will compare 6 well known machine learning models for classification: Gradient Boosting, k-Nearest Neighbors (KNN), Perceptron, Support Vector Machine (SVM), Naive Bayer and Random Forest. To make thing simpler, we will ignore the credit score number and only focus on the class of its, thus avoid the misunderstand of this problem type as regression.

## II. Data Description

We use the Credit Score Classification dataset from Kaggle, it consists of 2 files: train.csv and test.csv. There are 28 columns and 27 columns in the train.csv file and test.csv file, respectively. Each column contains the following information:

1) ID: Represents a unique identification of an entry
2) Customer_ID: Represents a unique identification of a person
3) Month: Represents the month of the year
4) Name: Represents the name of a person
5) Age: Represents the age of the person
6) SSN: Represents the social security number of a person
7) Occupation: Represents the occupation of the person

8) Annual_Income: Represents the annual income of the person
9) Monthly_Inhand_Salary: Represents the monthly base salary of a person
10) Num_Bank_Accounts: Represents the number of bank accounts a person holds
11) Num_Credit_Card: Represents the number of other credit cards held by a person
12) Interest_Rate: Represents the interest rate on credit card
13) Num_of_Loan: Represents the number of loans taken from the bank
14) Type_of_Loan: Represents the types of loan taken by a person
15) Delay_from_due_date: Represents the average number of days delayed from the payment date
16) Num_of_Delayed_Payment: Represents the average number of payments delayed by a person
17) Changed_Credit_Limit: Represents the percentage change in credit card limit
18) Num_Credit_Inquiries: Represents the number of credit card inquiries
19) Credit_Mix: Represents the classification of the mix of credits
20) Outstanding_Debt: Represents the remaining debt to be paid (in USD)
21) Credit_Utilization_Ratio: Represents the utilization ratio of credit card
22) Credit_History_Age: Represents the age of credit history of the person
23) Payment_of_Min_Amount: Represents whether only the minimum amount was paid by the person
24) Total_EMI_per_month: Represents the Equated Monthly Installments payments (in USD)
25) Amount_invested_monthly: Represents the monthly amount invested by the customer (in USD)
26) Payment_Behaviour: Represents the payment behavior of the customer (in USD)
27) Monthly_Balance: Represents the monthly balance amount of the customer (in USD)
28) Credit_Score: Represents the bracket of credit score (Poor, Standard, Good) (**this column do not appear in test.csv**)

## III. Exploratory Data Analysis

### A. Credit Score

The primary objective of this dataset is to classify Credit Scores into three values: Good, Standard, and Poor. After observing, we see most people are in the Standard classification.

### B. Anually Income and Monthly Salary

People who has a high in-hand monthly salary have a good credit score and who has a low in-hand salary has a low credit score Fig 2, but in contrast to that, the Annual Income of the Customers doesn't affect on the credit score as we see that the variance on the annual income and the people can still have a good credit score whether the cutomer has a 100000 USD or 250000 USD Annually. Fig 3


Fig. 1. Customers Credit Scores.

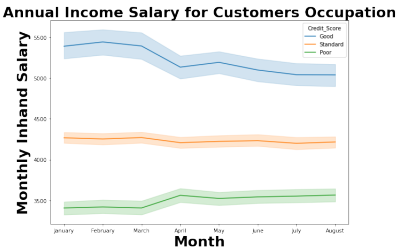


Fig. 2. Monthly Income Salary.

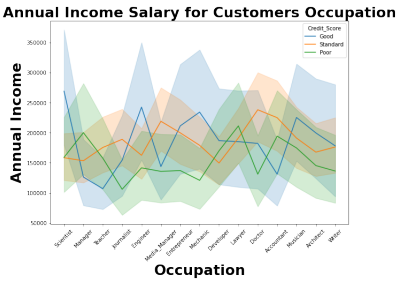


Fig. 3. Annual Income.

Customers between age 30 and 45 has the most Annual Income and the second more group age are customers between 14 and 25 which mean not people from 25 and 30 which indicate that there are people who can make money in a young age more than the old people but as the same time as indication that the 2 largest Categories most of their credit score are Standard or Poor but the as for the people between 45 and 55 have more good credit score than the young people from 14 to 25. Fig 4

### C. Credit Card Utiliztion Ratio, Investment and Debt

We can see in Fig 5 that more the people use the credit card it makes the credit score much better.

Along with that, most people who invested between 700 to 800 USD of their money have a good Credit Score and most people who have a standard credit score invested around 600 and 700 USD per month. Fig 6

On the other hand, in Outstanding Debt (Fig 7), people who don't use the credit card so much but also pay small portion

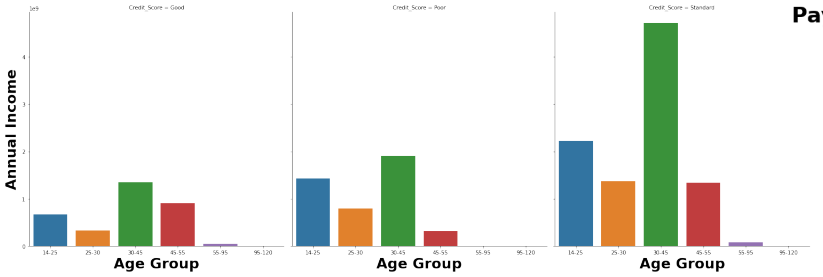


Fig. 4. Annual Income.

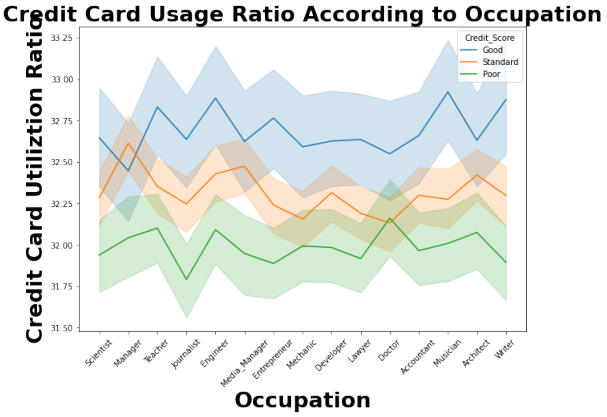


Fig. 5. Credit Card Utiliztion Ratio.

of the credit card has the majority on the outstanding debt (`Low_spent_Small_value_payments`) and the Category after that which has the second most outstanding debt the people who (`Medium_spent_Medium_value_payments`). And we can see that the people who have the least outstanding debt are `High_spent_Large_value_payments`.

Customers Between age of 30 and 45 the most category who have a lot of outstanding debts, meaning that people in their youth age have a high purchase power and customers between 45 to 55 their outstanding debt is less than young

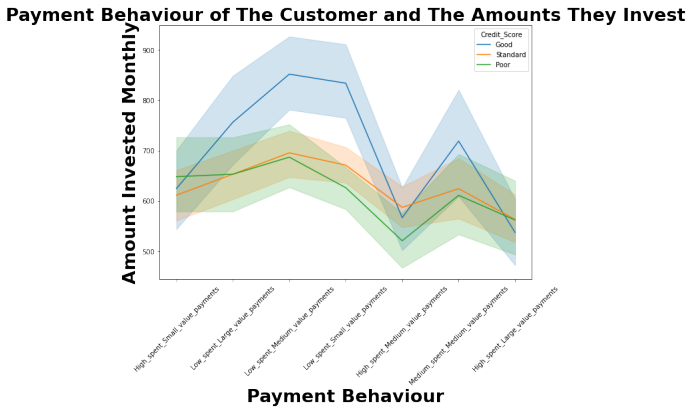


Fig. 6. Amount Invested Monthly.

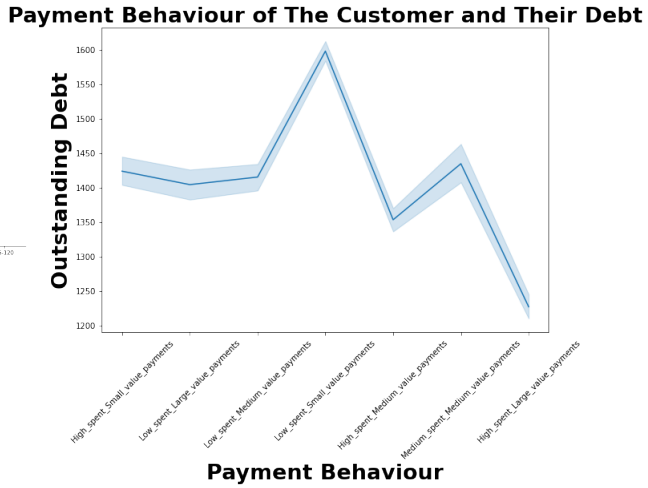


Fig. 7. Outstanding Debt.

people. Fig 8

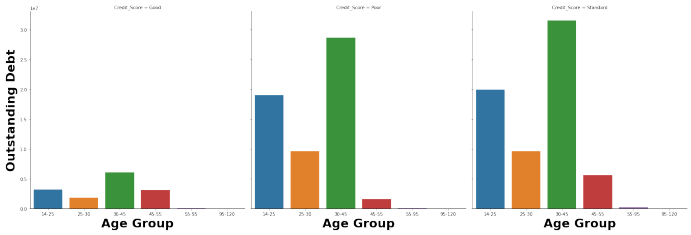


Fig. 8. Outstanding Debt in Categories.

### *D. Credit Mix, Payment and Bank Accounts*

Look at Fig 9 we can summarize:

Most of people who don't have a credit mix has a Standard Credit score and the second most category has a bad credit Score.

Most of people who have a good credit mix also have a good credit score and the second most category has a standard credit score.

As for people who have a standard mix, most of them has a standard credit score and the second most category have a bad credit score.

And with people who have a bad credit mix, most of them has a bad credit score and the second most category have a standard credit score.

Customers who pay the minimum amounts normally has a poor credit score. However, the people who don't pay the minimum amounts has a good credit score more than the others which mean that there are a lot of people who stay in debt for a long time as they don't pay the all amounts and they pay part of it which made an interest on them. Fig 10

The less Monthly in hand Salary that customer has, the more he/she delayed from the due date. But at the same time, there are people who delayed from the due date but also have a good credit score. Fig 11

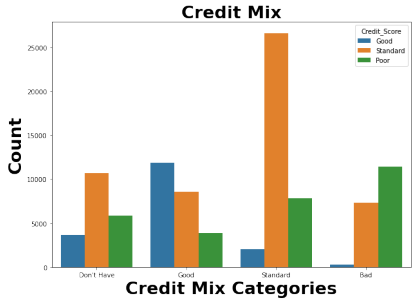


Fig. 9. Credit Mix.

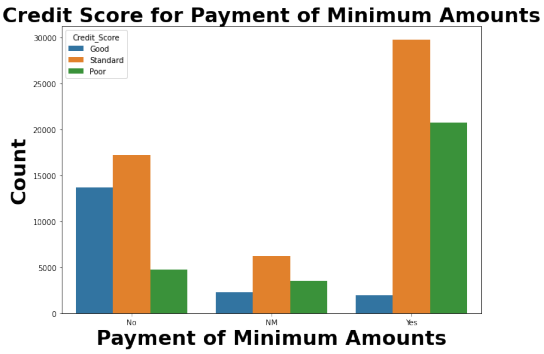


Fig. 10. Credit Score for Payment of Minimum Amounts

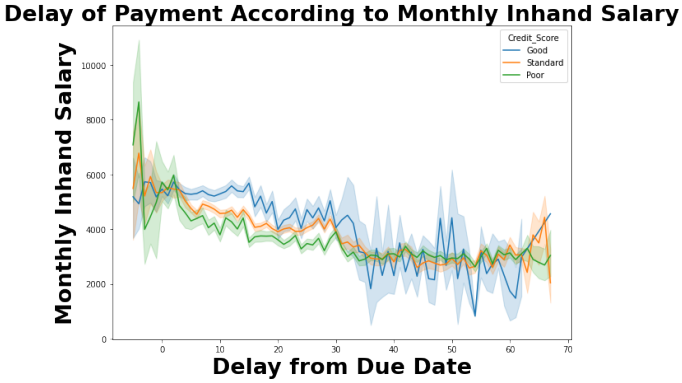


Fig. 11. Delay of Payment According to Monthly In-hand Salary.

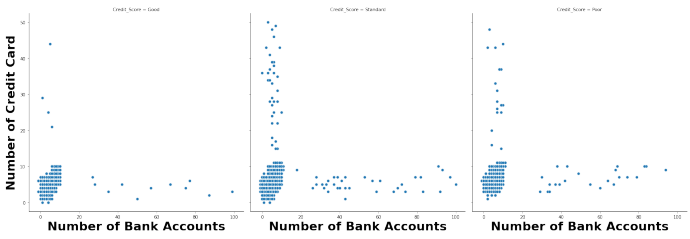


Fig. 12. Bank Accounts

Look into Fig 12 we see most people have Accounts from 0 to 10 Accounts and the number of credit cards also from 0 to 10 which mean each account has at least one credit card.

## IV. Prepare Data for Modeling

### A. Processing TypeOfLoan feature

TypeOfLoan has 9 types: AutoLoan, Credit-BuilderLoan, DebtConsolidationLoan, HomeEquityLoan, MortgageLoan, NotSpecified, PaydayLoan, PersonalLoan and StudentLoan.

With Split String function, we will separate TypeOfLoan column into 9 columns above. If record has any type of loan, we will track by 1. If it do not have that type of loan, we will track by 0.

### B. Label Encoding

CreditMix has 3 variables: Bad, Don't Have, Good and Standard. PaymentOfMinAmount and PaymentBehaviour include NM, No, Yes and HighSpentLargeValuePayments, HighSpentMediumValuePayments, HighSpentSmallValuePayments, LowSpentLargeValuePayments, LowSpentMediumValuePayments, LowSpentSmallValuePayments, MediumSpentMediumValuePayments, respectively.

The main target for the classification in this dataset is CreditScore. It contains 3 types that are Good, Poor and Standard.

### C. Drop unimportant features

In train dataset, we will drop Month, Age, occupation, TypeOfLoan, AgeGroup, CreditScore. However, CreditScore will be used to make target column.

In test dataset, we will remove Month, Age, Occupation, TypeOfLoan.

### D. Remove outliers and normalize the Data

In this project, we use IsolationForest algorithm with contamination parameter and there is no bootstrap. Therefore, the number of entries decreases from 100,000 to 80,000.

We often have two solutions for balanced labels that are undersampling and oversampling. In this case, we use oversampling from SMOTE library. After resampling, 3 labels are balanced with 34728 according to Standard.

Before modelling step, we use RobustScaler for xtrain and xtest dataset to get expected result.

## V. Modeling

### A. Settings

The raw dataset contains many noises that affect the performance of the models. So we applied some processes for handling with wrong and null values.

For training, we split the dataset with the ratio train:test equals to 8:2.

For evaluation, we use the metrics F1-Score and Accuracy. There are several hyper-parameters in our models. We set them empirically by the development performances or tune it with Grid-search.

Due to some problem with assigning label into confusion matrix, the class 0 represent the Good class, class 1 represent the Poor class and class 2 represent the Standard class.

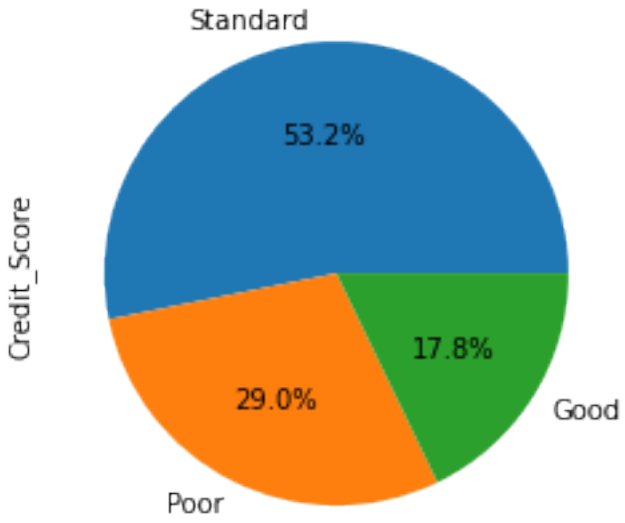


Fig. 13. Percentage of all labels in CreditScore feature.

```
xgb = XGBClassifier(
    eval_metric="mlogloss",
    objective="multi:prob",
    use_label_encoder=False,
    max_depth=9,
    learning_rate=0.3,
    n_estimators=500,
)
```

Fig. 15. Config original parameter.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.72 | 0.73 | 0.73 | 2963 |
| 1 | 0.78 | 0.79 | 0.78 | 4249 |
| 2 | 0.81 | 0.80 | 0.81 | 8788 |
| accuracy | | | 0.79 | 16000 |
| macro avg | 0.77 | 0.77 | 0.77 | 16000 |
| weighted avg | 0.79 | 0.79 | 0.79 | 16000 |

Fig. 16. Classification report.

### B. *GradientBoosting*

This is a classification problem, we use a specified model called XGBClassifier.

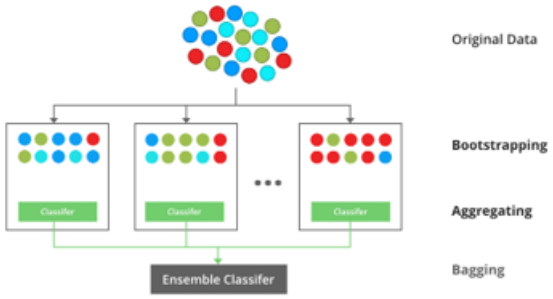


Fig. 14. XGB model.

Main idea is instead of building good model, we will combine many predictive models that have less accuracy (weak learner) when used individually but give high accuracy when combined.

The result of gradient descent is a weighted combination of gradients. The goal is minimize expected loss function.

F* = $argmin_F E_x[E_y(L(y, F(x)))|x]$

with: L(y, F(x)) = $(y - F)^2$

or: L(y, F(x)) = $|y - F|, y \in \mathrm{R}^1(regression)$

or: L(y, F(x)) = $\log(1+e^{(} - 2yF))$, y∈-1,1

F*(x) is mapping function that transform x into y.

Back to the dataset problem, we call XGBClassifier from xgboost package. The following model is experimented in Google Colab.

The accuracy score on train and test are 1.0 and 0.79, respectively.

More details about evaluation, we export the confusion matrix among 3 classes (good, poor, standard).

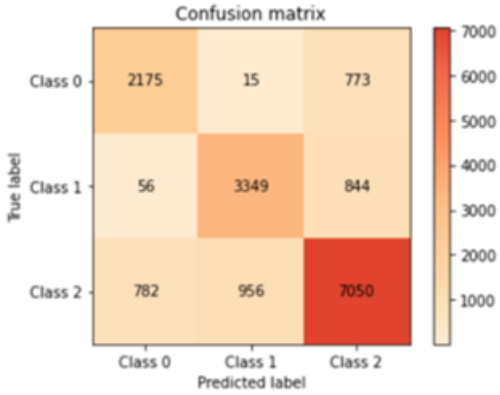


Fig. 17. Confusion matrix.

*Hyperparameter Tunning

### C. *RandomForest*

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. In this problem, we use it for classifying three classes of Credit Score (good, standard, poor).

Bagging, also known as Bootstrap Aggregation is the ensemble technique used by Random Forest. Bagging chooses a random sample from the data set. Hence each model - a decision tree, is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement known as row sampling. This step of row sampling with replacement is called bootstrap. Now each model is trained independently which generates results. The final output is

based on majority voting after combining the results of all models. This step which involves combining all the results and generating output based on majority voting is known as aggregation.

Step 1: In Random forest n number of random records are taken from the data set.

Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.

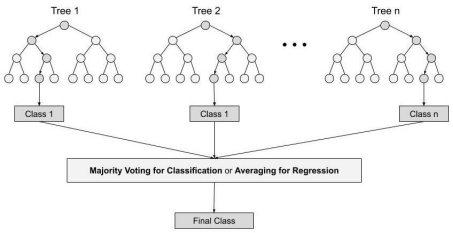


Fig. 18. Random Forest Idea.

To increase the performance of Random Forest Algorithm, we adjust these hyperparameters:

1. `n_estimators` – number of trees the algorithm builds before averaging the predictions. Higher number of trees give better performance but makes it slower.

2. `criterion` - measure the quality of a split.

3. `max_depth` - maximum number of levels allowed in each decision tree: to avoid overfitting.

4. `min_samples_split` - if node have number of samples larger than this value, split node. It could reduce overfitting if it large enough, too large will lead to underfitting.

5. `mini_sample_leaf` – determines the minimum number of leaves required to split an internal node. A smaller leaf makes the model more prone to capturing noise in train data.

6. `max_features` – maximum number of features random forest is allowed to try in individual tree.

Because we lack of resource, so Random Forest Model will be tuned with values:

`n_estimators` = [20,50,100]
`max_features` = [’auto’, ’sqrt’]
`max_depth` = [None, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
`mini_sample_leaf` = [1, 2, 4]

After finding best set of hyperparameter by using Grid Search, these are the best hyperparameters for this dataset.

`n_estimators` = 100
`max_features` = ’auto’
`max_depth` = 50
`mini_sample_leaf` = 1

Using those hyperparameters to train Random Forest model, the accuracy score on train and test are 0.99 and 0.77, respectively. Fig 19

More details about evaluation, we export the confusion matrix among 3 classes (good, poor, standard). Fig 20

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.65 | 0.74 | 0.69 | 2963 |
| 1 | 0.77 | 0.75 | 0.76 | 4249 |
| 2 | 0.82 | 0.78 | 0.80 | 8788 |
| accuracy | | | 0.77 | 16000 |
| macro avg | 0.74 | 0.76 | 0.75 | 16000 |
| weighted avg | 0.77 | 0.77 | 0.77 | 16000 |

Fig. 19. Random Forest report


Fig. 20. Random Forest Confusion matrix

Random Forest have many advantages:
- Missing values aren’t an issue.
- It can capture nonlinear relationships.
- Accuracy of random forest is generally very high.
- It notably efficient in large data sets.

### *D. KNN*

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. [9] While it can be used in either classification or regression problem, it’s mostly well known in solving classification problems. In this paper, we will explain the KNN Classifier only.

KNN classifier works by determining the class of a data point using majority voting principle. If k is set to 3, the classes of 3 closest points are checked. Same thing can go with different k value. The only special case is when k is set to 1 which make this algorithm become the nearest neighbor instead of k of them. Prediction is done according to the majority class.

For our test, we used Google Colab as a testing platform. As for our model, we used KNearestNeighbors from Scikit-learn with parameter tuning around these values:

`n_neighbors` = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ]
`weights` = [ ’uniform’, ’distance’ ]

### *E. Perceptron*

Perceptrons were one of the first algorithms discovered in the field of AI. Its big significance was that it raised the hopes

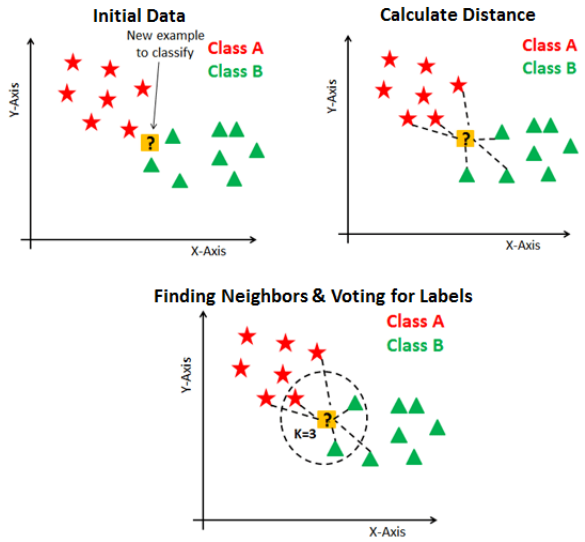


Fig. 21. KNN Model

Number of n_neighbors: 3

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Good | 0.57 | 0.57 | 0.57 | 2963 |
| Poor | 0.74 | 0.68 | 0.71 | 4249 |
| Standard | 0.75 | 0.77 | 0.76 | 8788 |
| accuracy | | | 0.71 | 16000 |
| macro avg | 0.68 | 0.67 | 0.68 | 16000 |
| weighted avg | 0.71 | 0.71 | 0.71 | 16000 |

Fig. 22. KNN report

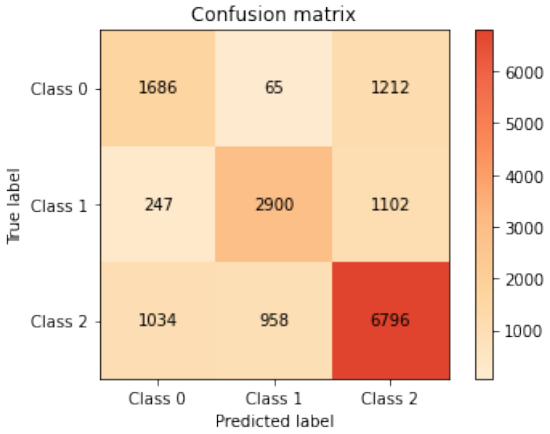


Fig. 23. KNN confusion matrix

and expectations for the field of neural networks. Inspired by the neurons in the brain, the attempt to create a perceptron succeeded in modeling linear decision boundaries.

Let's consider the structure of the perceptron. The perceptron has four key components to it:

- Inputs
- Weights
- Weighted Sum
- Thresholding using the unit-step function

(Fig 24)

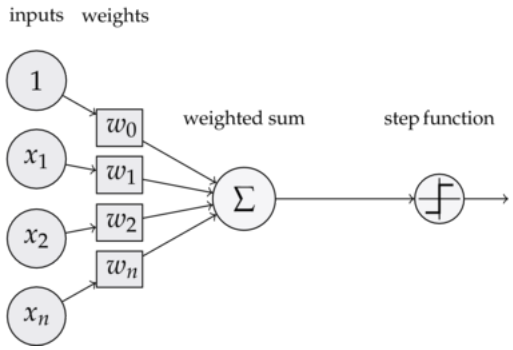


Fig. 24. Perceptron structure

We will consider the update rule. That is, the algorithm computes the difference between the predicted value and the actual value. The difference is defined as an error. If the predicted value is the same as the real value, then the error is 0; otherwise, it's a non-zero number. Once the errors have been computed for all the data samples, then the parameters are updated. The parameters define the learning model, and in this case, it's the weights.

The learning update rule is given as follows:

$$weights_j := weights_j + \alpha(y^{(i)} - h_\theta(x^{(i)})x_j^{(i)}$$

We use Grid Search to find the best learning rate in the set [0.0001, 0.0003, 0.001, 0.003], and we get the best result is accuracy equal to 0.43 for learning rate = 0.0001 (Fig 25, Fig 26).

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.24 | 0.35 | 0.29 | 2963 |
| 1 | 0.34 | 0.36 | 0.35 | 4249 |
| 2 | 0.61 | 0.50 | 0.55 | 8788 |
| accuracy | | | 0.43 | 16000 |
| macro avg | 0.40 | 0.40 | 0.39 | 16000 |
| weighted avg | 0.47 | 0.43 | 0.45 | 16000 |

Fig. 25. Perceptron report

### *F. Support Vector Machines - SVM*

Support vector machine is an algorithm which is highly used as it produces significant accuracy with less computation power. Support Vector Machine, abbreviated as SVM can be

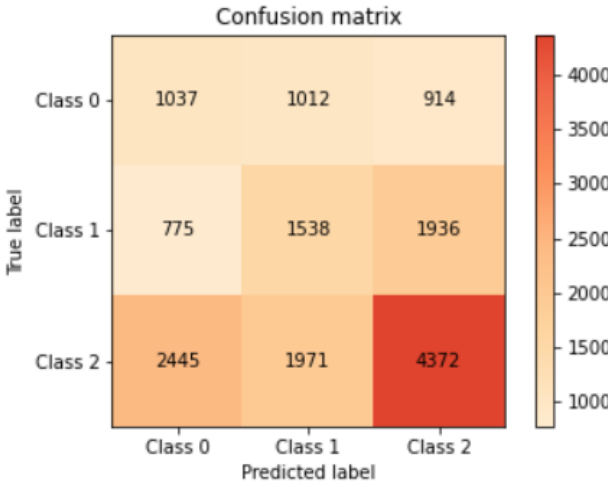


Fig. 26. Perceptron confusion matrix

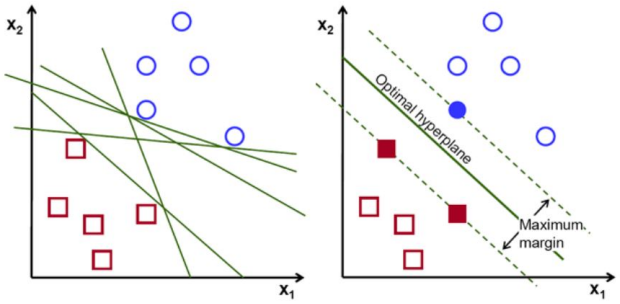


Fig. 27. SVM Algorithms: Possible hyperplanes(left) and optimal hyperplane(right)

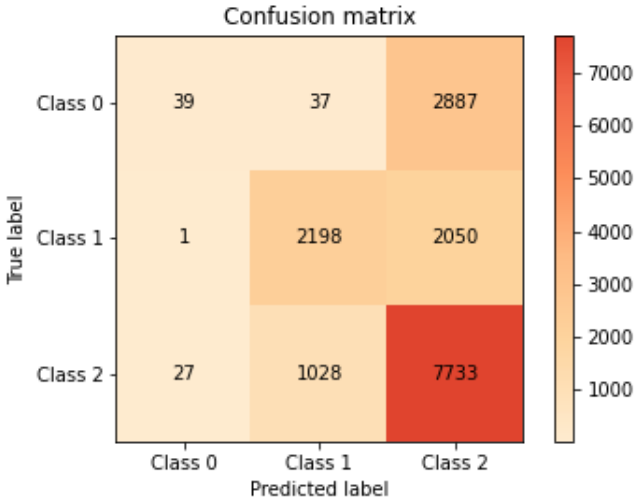


Fig. 28. Support Vector Machine confusion matrix

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.58 | 0.01 | 0.03 | 2963 |
| 1 | 0.67 | 0.52 | 0.59 | 4249 |
| 2 | 0.61 | 0.88 | 0.72 | 8788 |
| accuracy | | | 0.62 | 16000 |
| macro avg | 0.62 | 0.47 | 0.44 | 16000 |
| weighted avg | 0.62 | 0.62 | 0.56 | 16000 |

Fig. 29. Support Vector Machine report

used for both regression and classification tasks. But, it is widely used in classification objectives.

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points (Fig 27).

To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

That is the binary-classifier idea, to apply it for this 3-class problem, we use One-versus-rest method.

After training, we can see in Fig 28 that the SVM model performs very well in classify Class 2 (F1-score: 0.72), but as a trade-off, it almost could not detect Class 0. The model predicted many instances of Class 0 as Class 2.

The reason for this issue is: The SVM algorithm is effective for balanced classification, and it does not perform well on *imbalanced* datasets. Which is that we can see in Fig 29: Class 0 has only 2963 instances, while Class 2 has 8788 instances!

### G. Bayesian

We use Naive Bayes classification with Bernoulli distributions for this problem. Because the data is scale between -1 and 1 so we assume that the data is distributed according to multivariate Bernoulli distributions with the binary threshold equal to 0.

The decision rule for Bernoulli naive Bayes is based on:
$P(x_i \mid y) = P(x_i = 1 \mid y)x_i + (1 - P(x_i = 1 \mid y))(1 - x_i)$

We use the BernoulliNB class with the binary threshold equal to zero from naive bayes libaray of package sklearn. The following model is experimented in Google Colab.

```
nb = BernoulliNB(binarize = 0)
```

Fig. 30. Bernoulli Naive Bayes Classification

The accuracy score on train and test are 0.62 and 0.63, respectively.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.47 | 0.72 | 0.57 | 2963 |
| 1 | 0.59 | 0.64 | 0.61 | 4249 |
| 2 | 0.76 | 0.59 | 0.66 | 8788 |
| accuracy | | | 0.63 | 16000 |
| macro avg | 0.61 | 0.65 | 0.61 | 16000 |
| weighted avg | 0.66 | 0.63 | 0.63 | 16000 |

Fig. 31. Bernoulli Naive Bayes report

More details about evaluation, we export the confusion matrix among 3 classes (good, poor, standard).

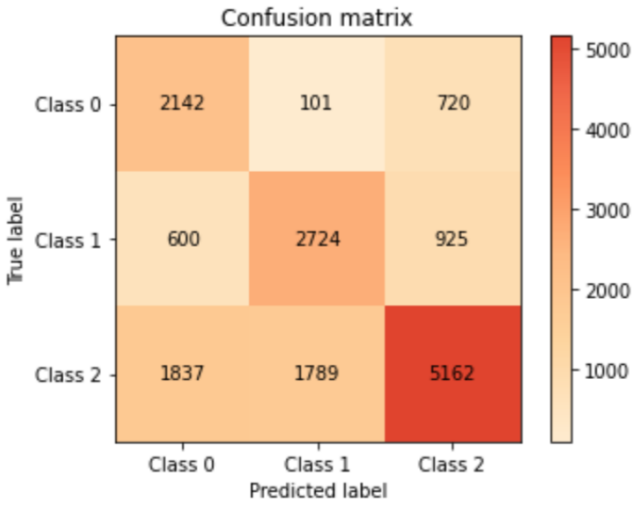


Fig. 32. Bernoulli Naive Bayes Confusion matrix

## VI. Conclusion

Here is the summary of six machine learning algorithms for classification we use to solve the Credit Score Classification problem:

| Methods List | | |
|---|---|---|
| Method | Macro F1 | Accuracy |
| GradientBoosting | 0.77 | 0.79 |
| RandomForest | 0.75 | 0.77 |
| KNN | 0.68 | 0.71 |
| Perceptron | 0.39 | 0.43 |
| SVM | 0.44 | 0.62 |
| Bayesian | 0.61 | 0.63 |

Most of our algorithms (KNN, SVM, Beyesian) have accuracy in the range of 0.6 to 0.7. But SVM have Marco F1 score 0.44, which is quite low, because SVM does not perform well on imbalanced datasets. The two best performance algorithms are GradientBoosting and RandomForest, those accuracy is almost 0.8. This is not surprising, cause XGBoost is known as a very good classification algorithm that is used in many contest, and Random Forest can capture nonlinear relationship and perform well with missing values, large data, like our Credit dataset. Perceptron has a low accuracy just 0.43, since it is bad at large, imbalnced, multi-dimensional dataset.

## References


[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, "Title of paper if known," unpublished.

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[8] https://en.wikipedia.org/wiki/Credit$_score$

[9] https://www.ibm.com/topics/knn