HEC MONTRÉAL

MATH60607A.A2022

ALGORITHMS FOR OPTIMIZATION AND BIG DATA ANALYSIS

Presented to Gilles Caporossi

Prepared by:
Huang, Huanhuan — 11252307
Aliakbarlou, Hamid — 11291818
Vuong, Billy — 11169018

1. Introduction of the problem

Granting credit plays a key role in financial transactions. This task typically involves a lot of borrower-related variables. However, in many situations, having a large number of variables can introduce noise into the database when building predictive models. In this context, feature selection presents itself as a way to simplify a database by identifying key features, reducing computational costs, and improving prediction performance.

Feature selection is an NP-hard problem (Peng, Albuquerque, Kimura, & Saavedra, 2021), and there is no algorithm that can be used to solve it in polynomial time. Given the number of variables, we can find the number of variables(n) to the power of 2 = 2^n for our feature combinations.

In this project, we offer three metaheuristic methods for finding feasible solutions with a machine learning model (Support Vector Machine) for feature selection tasks: Simulated Annealing, Variable Neighborhood Search, and Genetic Algorithm. Finally, we summarise our findings and insights regarding the results through comparison and analysis.

2. Methodology

2.1 Dataset Description

The dataset we used for this project is the German Credit Data, which categorises loan applicants as good or bad credit risks based on a set of attributes. There are 1000 observations in this dataset, with 21 variables (7 numerical and 13 categorical), and no missing values. The target variable Y is a binary variable with two levels: good credit and bad credit. If a loan applicant is labelled as 0, it indicates that he or she has "bad credit" based on the 21 variables associated, or vice versa. Each variable is described in detail in the appendix (Table 1: Description of the dataset).

2.2. Standardization of Data

We standardised the features to have a mean of zero and a standard deviation of one. Because the features are measured on different scales, the standardisation step ensures that all features contribute equally, and that bias is avoided.

2.3 Dataset split

The data is divided into three categories: training (64%), validation (16%), and testing (20%). The machine learning model is trained using pairs of feature sets and labels generated from the

training data set. We use the validation set to compare the model's performance and the test set to see the generalisation performance of the model.

2.4 Feature representation

We use binary coding to generate feature representations, such that each feature set is represented by a set of 0s and 1s. A feature is encoded as 1 if selected, otherwise as 0. In this project, we have 21 variables under investigation. Hence, when all features are included, the feature set representation is [1,1, 1, ... 1,1,1], while none of the features selected is [0,0,0,…0,0,0].

2.5 ML classifier

SVM is the machine-learning model we use. We chose this model because SVM is known for overfitting in high dimensions and that the overall performance is heavily influenced by the variables used. If there were too many noisy features, SVM would take them into account, overfitting and increasing computation time. We are intrigued by the outcome on how SVM works with the feature selection. For simplicity, we use SVM from Scikit-learn with default parameters.

2.6 Evaluation criteria

There are several metrics for comparing the performance of different predictive methods. Among all, we use one of the most elementary and widely used: accuracy, which can be represented by the following equation:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

3.Models

In this session, we cover the setup for the base model as well as three metaheuristics: simulated annealing, variable neighbourhood search, and genetic algorithm.

3.1 Base model (SVM with all features)

To see how SVM's performance improved after feature selection, we first built a dummy model with SVM with the full feature set.

3.2 Transformation

We defined the following transformation rules for SA and VNS, where j is the number range between 1 and total number of the variables.

- Transformation t_1:
  - Randomly **invert j** number of features in the current solution set. For any features in the current solution set, we randomly sample **j** numbers and activate/deactivate those selected features in the current solution.
  - The respected neighborhood size N^{t_1}can be calculated as : $\frac{factorial(total\ number\ of\ all\ features)}{facotrial(total\ number\ of\ all\ features-j)*factorial(j)}$
  - Example: [0,1,0,1] → [1,1,0,1], [0,0,0,1], [0,1,1,1], [0,1,0,0]
- Transformation t_2:
  - Randomly **remove j** number of features to the current solution set. For any activated features in the current solution set, we randomly sample **j** numbers from the activated features set and remove those selected features from the current solution.
  - The respected neighborhood size N^{t_2} can be calculated as : $\frac{factorial(total\ number\ of\ activated\ features)}{facotrial(total\ number\ of\ activated\ features-j)*factorial(j)}$
  - Example: [0,1,0,1] → [0,0,0,1], [0,1,0,0]
- Transformation t_3:
  - Randomly **add j** number of features to the current solution set. For any inactivated features in the current solution set, we randomly sample **j** numbers from the inactivated features set and add those selected features to the current solution.
  - The respected neighborhood N^{t_3} size can be calculated as : $\frac{factorial(total\ number\ of\ inactivated\ features)}{factorial(total\ number\ of\ inactivated\ features-j)*factorial(j)}$
  - Example: [0,1,0,1] → [1,1,0,1], [0,1,1,1]

3.3 Simulated Annealing +SVM

Simulated annealing (SA) is a stochastic global search algorithm for function optimization. It is a probabilistic optimization method that is simple to implement and effective for complex problems. It considers most of the possible local optimums, occasionally choosing a worse solution to avoid the local optimum.

SA algorithm takes in three parameters, and the effectiveness of the algorithm depends on the choice of these parameters. Tuning these hyperparameters is critical for getting the most out of the model. They are as follows:

- The first parameter is α which is a factor that determines the rate of temperature reduction. In fact, alpha restricts the search space. A higher value of alpha leads the temperature to reduce at a faster rate. After tuning, we set it to 0.97.
- The second parameter is T_0, which is the initial temperature. we set to 1 after tuning.
- The third one is β as the normalizing constant. If the chosen value of beta is too high, i.e., probability of rejecting a set of parameters is too low in later iterations, this might lead to an infinite loop. we also tuned this parameter and chose 1 for it.

For its implementation in Feature selection, it can be described as the following five steps.

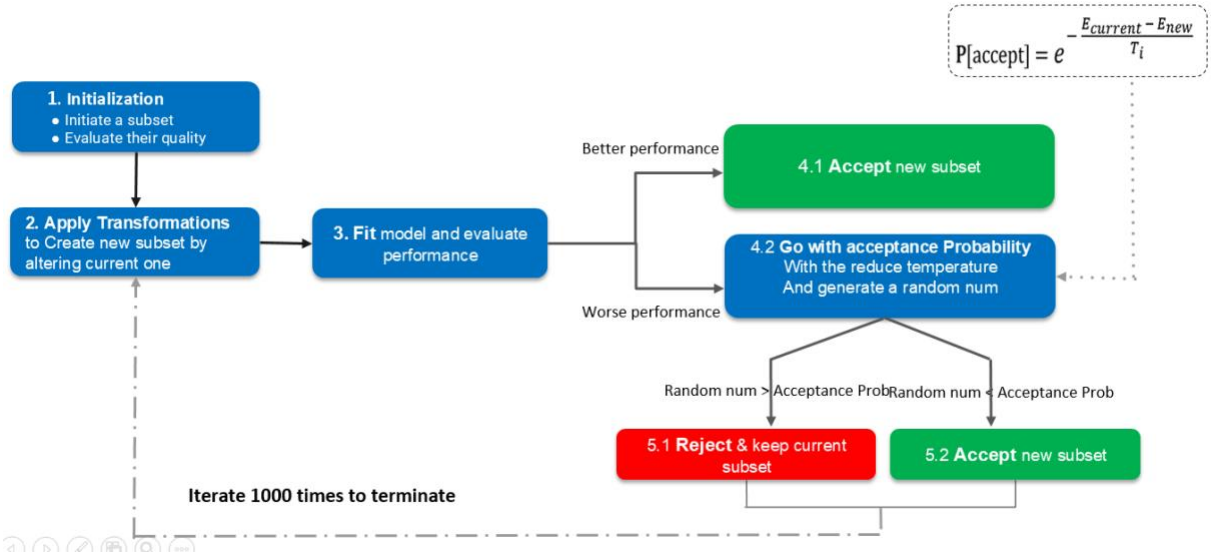


Figure 1: Simulated Annealing algo for Feature selection

1.1 Initialization

1) Generate an initial subset of features from which all features are selected. (Optional: choose 50% of the total set of features at random.)

2) Set the maximum iterations to run from step 1.2 to step 1.5. Here, we set it to 2000.

3) Evaluate SVM's performance on the validation data and calculate the initial accuracy.

1.2 Transformation:

1) Randomly modify the current subset's feature to create a new one by adding (t_3), replacing (t_1), or removing a feature(t_2).

    - Notes:
        - 1) If a subset has already been visited, we repeat this step to generate a new unseen subset.
        - 2) In this project, we just applied the above-mentioned transformations for one feature only (i.e., j=1).

1.3 Performance metric calculation

1) Refit SVM on the new transformed subset and calculate its respective accuracy.

1.4 Performance metric comparison

1) Compare with the current subset accuracy:
    - If the new subset performs better, accept the new subset and replace the current state, and stop the iteration.
    - If the performance is worse, go to step 1.5.

1.5 Apply the main metaheuristic characteristic of the algorithm:

1) Calculate $p_{accept} = e^{-\frac{E_{current}-E_{new}}{T_i}}$
2) Generate a probability p from uniform distribution and compare with p_{accept}
    - If p is greater than the calculated probability, keep the current subset and reject the new one.
    - Otherwise, accept the new subset and update it as current solution.

In this step, rejection rate is affected by the two following factors:

1) p_{accept} decreases as the performance of the new subset model deteriorates (i.e., more difference between $E_{current}$ and E_{new}). Correspondingly, it means that we are more likely to reject the new subset with poor performance.
2) As the T_i decreases, so does p_{accept}. This means that there is a greater chance of the probability p being larger than the p_{accept}. This also suggests that we are likely to reject the new subset, which has poor performance.

3.4. VNS – Variable Neighborhood Search

Variable neighborhood search (VNS) searches for the best solution by exploring predefined number of k neighborhoods (Hansen, Mladenović, Todosijević, & Hanafi, 2017).

Basic VNS contains two main components:

1) A local search allows successive improvement over the current best solution.

- For this project, the local search is limited by the three transformations (t_1, t_2, t_3) defined above.

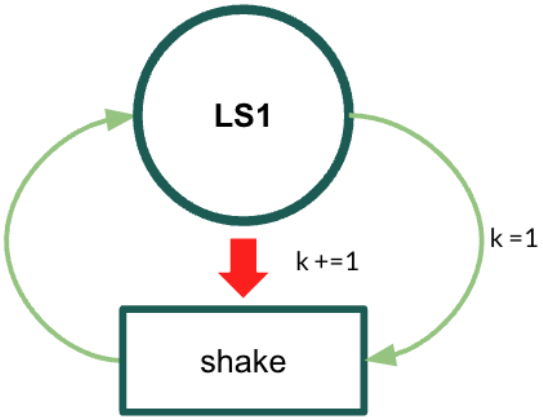
2) A perturbation allows the process to leave a local optimum.

- For this project, the shaking function is used in conjunction with the three transformations defined above (t_1, t_2, t_3). More specifically, we have three $S_t(s)$. During the shaking step, the three transformations are selected randomly to generate a new neighborhood $N_t(s)$. Once a shaken neighborhood is generated, we choose the first solution in this neighborhood as a new starting point for the local search (the first solution in this neighborhood is randomly generated).
- To allow the shaking intensity, we also include intensity parameter k to define k-shaking s_t^kk(s) by applying k times the shaking $S_t(s)$. We set kmax to 6 for VNS-LS as well as VNS-VND, which allow VNS to explore more neighbourhood during the transformation phase.

3.4.1 VNS-LS

Basic VNS-LS has three main steps:

Figure 2: VNS-LS



1.1 Initialization

1) Define the transformations/LS as: Transformation t_1 with j = 1.
2) Generate an initial subset of features from which all features are selected.
3) Set the maximum number of iterations to run Steps 1.2–1.4: We set it to 100.
4) Evaluate SVM's performance with this solution set on the validation data and calculate the initial accuracy.

1.2 Transformation:

1) Generate transformed solution with the LS1 from the current solution.

1.3 Performance metric calculation

1) Refit SVM on the new transformed neighborhood with all the neighbors.
2) Calculate their respective accuracies and select the best one among the neighbors.

1.4 Performance metric comparison

1) Compare with the current subset accuracy:
    - If the new subset performs better, apply shaking k =1.
    - Otherwise, increase k to shake the current solution with larger intensity.

3.4.2 VNS –VND

VND use a meta local search, previously we have one LS, now we have implemented three LSs (t_1, t_2, t_3). We set the features to be 2 in t_2, t_3. Since if we were to add and remove a feature, the neighborhood would be the same as using t_1.

VND can be described as the following:

1.1 Initialization

1) Define the transformations/LS as: Transformation t_1 with j = 1, transformation t_2 with j = 2 and transformation t_3 with j = 2.

2) Generate an initial subset of features from which all features are selected.

3) Evaluate SVM's performance with this solution set on the validation data and calculate the initial accuracy.

1.2 Transformation:

1) Generate transformed solution with the LS from the current solution. The first iteration starts from LS1.

1.3 Performance metric calculation

1) Refit SVM on the new transformed neighborhood with all the neighbors.

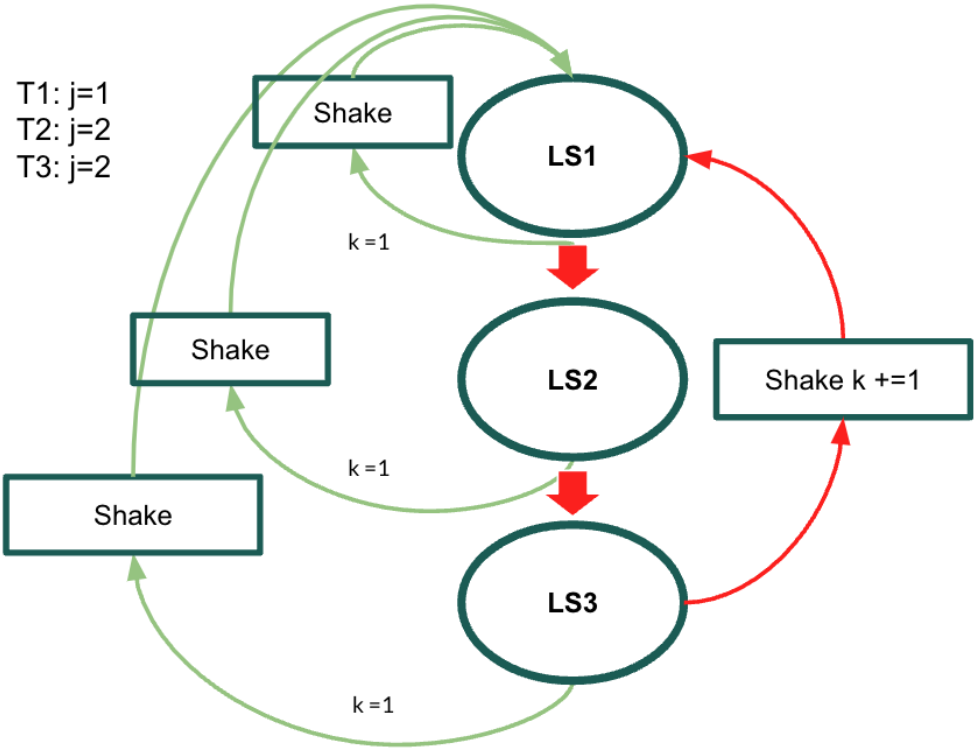
2) Calculate their respective accuracies and select the best one among the neighbors.

1.4 Performance metric comparison

1) Compare the best solution from the LS1 with the current subset accuracy:
    - If the new subset performs better, apply shaking k =1 and go back to LS1
    - Otherwise, move on to the next neighborhood and repeat the step 1.2 to 1.4.

The VNS-VND overall whole architecture can be described as the following:

Figure 3: VNS-VND



1.1 Initialization

1) Set the maximum number of iterations to run the following Steps 1.2: We set it to 100 max counts.

1.2 VND and shaking step:

1) Start the VND process described above.
2) If LS3 is reached,
    a. If the solution from the whole VND process is not improved, increase k by 1 to generate the shaken solution.
    b. Otherwise, set k to 1 and generate the shaken solution
3) Return to LS1 again with the shaken solution.
    a. Each time VND goes back to LS1, we increase the count by 1.

3.5. Genetic Algorithm

Genetic algorithm has four basic steps: 1) Population initiation; 2) Selection; 3) Crossover; and 4) Mutations. Steps 2-4 alternate until a termination criterion is reached; this criterion we set for this project is a maximum number of generations.

The overall Genetic Algorithm can be described as:

1.1 Initialization

1) An initiated nation's population is made up of 1000 people. During the initiation, it is totally random when one feature is presented or not to these 1000 individuals.
2) We calculate the accuracy of each individual initiated using SVM and choose the best one among these 1000 as our initial quality baseline.

1.2 Reproduction

1) For each generation, we apply the elitism strategy to directly select the best candidate from the next population, as we did in the current generation. This ensures that we do not lose the best solution during evolution.
2) Selection
    a. In this phase, we randomly selected 100 individuals from the current population. Among these 100 individuals, we choose the two best (individual 1, individual 2) to proceed with crossover and mutation based on their quality.
3) Cross-over
    a. For each feature of the individual, draw a random number from uniform distribution
        i. If the random number drawn is larger than 0.5, replace the new offspring with that feature from individual 1.
        ii. Otherwise, we replace the new offspring with that feature from individual 2.
        (Note: This step will produce the offspring who will have the features from individuals 1 and 2 by swapping the elements between them based on the random value compared with 0.5. The offspring will then take the place of their parents in the new population.)
4) Mutation
    1) We select a random number between 0 and the total number of variables for each offspring, with the selected number corresponding to the gene or feature to be inverted.
    2) Sample another random number from uniform distribution to decide whether we mutate or not. The probability of mutations is set to a rather large value (80%) to encourage evolutionary.

a. If the feature is selected and the random number is larger than 0.8, we invent its value from 0 to 1 or from 1 to 0.
b. Otherwise, do not mutate.

After selection, crossover, and mutations in Reproduction for 999 times, the 999 individuals along with the best individual in the current population form the new generation that replaces the previous one.

This process is repeated 100 times for 100 generations. The best solution is the best individual in the 100th generation.

1.3 Performance metric calculation

1) Refit SVM on each individual from the new population.
2) Calculate their respective accuracies, select the best one among the population, and update it as the current best solution.

Repeat the reproduction phase till 100 generation is reach. The best solution is the best individual in the last generation.

4. Result analysis

4.1 Performance comparison

Table 1: All models Performance on Valid/Test set

Accuracy	Base model	SA	VNS-LS	VNS-VND	GA
Validation set	0.70625	**0.742**	0.739375	**0.75**	**0.75**
Test set	0.700	**0.7175**	0.709	0.705	0.705

(**Note**: For SA, VNS-LS, VNS-VND, the results are based on the 10 iterations average — see Appendix Table 2.)

From the result table above, we can observe the following insights.

- Firstly, with all features activated, the base accuracy of the validation set is 0.70625. We can see that SVM with feature selection has increased the accuracy from 0.706 to 0.74 and higher.

- Secondly, based on the accuracy of the validation set, we can see that VNS-VND and GA have the best performance since VNS-VND is not limited by only one transformation rule and has more room to explore during meta-local search, and GA is unlimited by any rule and has a large space to improve through the evolutionary (i.e., reproduction phrase).

- Thirdly, what is surprising is that SA with three transformations has outperformed VNS-LS and generalised the best in the test set.
    - The possible explanation for this outstanding performance could be the following:
        - Multiple transformation increases the neighborhood size for SA to explore
        - By default, SA accept worse solutions with a given probability and hence helpful to get out of the local optima.
        - When iteration goes infinitive, VNS will remain at the local optima, where nothing can be improved by the predefined shaking function.

- Finally, the accuracy on the test set is lower than the validation set, which means that it's not able to generalise well with unseen data. This is to be expected given the small size of the dataset (1000 total observations).

4.2 Feature selection comparison

Table 2: Feature Selected -All models

Models	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	sum
SA	1	1	1	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1	1	1	0	9
GA	1	1	1	0	1	1	0	1	1	0	0	0	0	0	1	0	0	1	1	0	0	9
VNS-VND	1	1	1	0	1	1	0	1	1	0	0	0	0	0	1	0	0	1	1	0	0	9
VNS-LS	1	1	1	0	1	0	1	1	1	0	1	0	0	0	1	0	0	0	0	1	0	10

From the figure above, we can observe the following insights.

- Firstly, we can see from the table above that SA, GA, VNS-VND use fewer features, totaling nine, whereas VNS-LS uses ten.
- Secondly, among the features chosen, they all agree half of the time.
- Thirdly, by looking at the features that VNS-VND selected and those of GA, we can see they agree all the time. This explains the same performance of the GA and VNS-VND.
- Fourthly, by looking at the features that SA selected and those of GA and VNS-VND, we can see they agree most of the time. (7 out of 9 times). This explains the very close performance between the SA, VNS-VND, and GA in the validation set.

5. Challenges

The scope of this project was deemed worthy of a team effort, here we summarize our challenges into the following aspects:

1. Coding experience
   - Coding collaboration within a team is challenging due to difference style in code structure and naming convention.
   - As intermediate beginners in Python coding, we struggle with debugging, such as implementing a stopping condition using computational time with a while loop that leads to failure to stop VNS-VND, and modifying our code in a more professional manner, such as by building modular functions, classes, and so on.
   - We are faced with challenges in optimising the code's performance, such as reducing the input of each function and the number of times each list is called, among other things.

2. Metaheuristics method implementation
   - Finding an appropriate database with a suitable number of features to run more demanding metaheuristics in a reasonable amount of time with the genetic algorithm and

VNS-VND, the compute time can be as long as 2 hours with the SPAM dataset using 57 features. This also varies with the machine-learning models used.

- Defining the transformation rules, the local searches, the sequence of LS execution, the neighbourhood size calculation, and how to use LS to build correct VNS and VND
- Implement innovation in SA and define three transformations to find different solutions.
- Trial and error in tuning parameters, in selecting the best scheme for all methods. For example, setting initial input parameters for SA to increase its performance. Regarding the temperature reduction scheme). There is a trade-off between a better solution and time consumption. Such as a linear scheme with a suboptimal solution or a gradual reduction strategy with a better result.
- There have been attempts to implement GA with MPI, but the bottleneck issue of the master remains unsolved.

Overall, it was challenging and very rewarding to code most of the algorithm without using any libraries. All the challenges we had have a positive impact in advancing our coding skill and understanding on metaheuristics and feature selection.

Conclusion

To summarise, the metaheuristics SA, VNS-LS, VNS-VND, and GA outperformed the base model with all features activated. SA outperformed the VNS-LS and had similar performance to GA and the VNS-VND, and, more importantly, it generalised very well in unseen data.

In the future, it might be interesting to revisit these metaheuristics with larger datasets and more features, as well as to experiment with other ML classifiers to see if their power in feature selection can also be beneficial.

Appendix

Table 1: Dataset information

Attribute 1: (qualitative)
Status of existing checking account

0: ... < 0 DM
1: 0<=...< 200DM
2: ... >= 200 DM /salary assignments for at least 1 year
3: no checking account

Attribute 2: (numerical)
Duration in month

Attribute 3: (qualitative)
Credit history

0: no credits taken/all credits paid back duly
1: all credits at this bank paid back duly
2: existing credits paid back duly till now
3: delay in paying off in the past
4: critical account/other credits existing (not at this bank)

Attribute 4: (qualitative)
Purpose

0: car (new)
1: car (used)
2: furniture/equipment
3: radio/television
4: domestic appliances
5: repairs
6: education
7: (vacation - does not exist?) 8: retraining

Attribute 7: (qualitative)
Present employment since

0: unemployed
1: ... < 1 year
2: 1 <=...<4years
3: 4 <=...<7yearS
4: .. >= 7 years

Attribute 8: (numerical)
Instalment rate in percentage of disposable income

Attribute 9: (qualitative)
Personal status and sex

0: male: divorced/separated
1: female: divorced/separated/married
2: male: single
3: male: married/widowed
4: female: single

Attribute 10: (qualitative)
Other debtors / guarantors

0: none
1: co-applicant
2: guarantor

Attribute 11: (numerical)
Present residence since

Attribute 12: (qualitative)
Property

0: real estate
1: if not 1 : building society

Attribute 16: (numerical)
Number of existing credits at this bank

Attribute 17: (qualitative)
Job

0: unemployed/unskilled - non-resident
1: unskilled - resident
2: skilled employee / official
3: management/ self-employed/ highly qualified employee/ officer

Attribute 18: (numerical)
Number of people being liable to provide maintenance for

Attribute 19: (qualitative)
Telephone

0: none
1: yes, registered under the customers name

Attribute 20: (qualitative)
foreign worker

0: yes
1: no

Attribute 21: (qualitative)
Sexe

0: male
1: female

9: business
10: others

Attribute 5: (numerical)
Credit amount

Attribute 6: (qualitative)
Savings account/bonds

0: ... < 100 DM
1: 100<=...< 500DM
2: 500 <= ... < 1000 DM
3: .. >= 1000 DM
4: unknown/ no savings account

savings agreement/life insurance
2: if not 1/2 : car or other, not in attribute 6
3: unknown / no property

Attribute 13: (numerical)
Age in years

Attribute 14: (qualitative)
Other instalment plans

0: bank
1: stores
2: none

Attribute 15: (qualitative)
Housing

0: rent
1: own
2: for free

Target Variable: (qualitative)
Credit

0: bad credit
1: good credit

Table 2: VNS-LS, VNS-VND, SA performance with 10 iterations

(note: VNS-VND always in the local optima in the same accuracy as the result table)

	Validation set accuracy			Test set accuracy		
N	VNS-LS	VNS-VND	SA	VNS-LS	VNS-VND	SA
1	0.7375	0.75	0.74375	0.71	0.705	0.725
2	0.74375	0.75	0.74375	0.715	0.705	0.72
3	0.7375	0.75	0.7375	0.71	0.705	0.72
4	0.74375	0.75	0.74375	0.725	0.705	0.715
5	0.73125	0.75	0.74375	0.695	0.705	0.72
6	0.7375	0.75	0.75	0.715	0.705	0.715
7	0.7375	0.75	0.74375	0.7	0.705	0.715
8	0.74375	0.75	0.73125	0.715	0.705	0.715
9	0.75	0.75	0.7375	0.715	0.705	0.715
10	0.73125	0.75	0.74375	0.69	0.705	0.715
avg	0.739375	**0.75**	**0.742**	0.709	**0.705**	**0.7175**

Detailed complexity analysis

The step we apply for the transformation's rules can be described using the following pseudo code:

- Initiated neighborhood list
- **j** is the number range between 1 and total number of the variables(k); worse case j =k, while in this project, it is not possible.
- **n** is the neighborhood size

1. For i in range of the neighborhood size:
2. Copy the current solution
3. Randomly sample the j indexes from current solution to apply to add/remove/invert
4. For all j:
5. Add/remove/invert its value
6. Append the transformed solution to the neighborhood list

Complexity

Line	Complexity	time	Total complexity
1	O(n)	O(1)	O(n)
2	O(n)	O(1)	O(n)
3	O(1)	O(n)	O(n)
4	O(1)	O(n)	O(n)
5	O(1)	O(n)	O(n)
6	O(1)	O(n)	O(n)

Because the transformation (add, remove, or invert) simply reassigns the current value to 0 or 1. It takes a constant amount of time, O (1). Here, we set any operation with j or k as O(1) since, compared to the neighbourhood size, they are relatively small. Therefore, the bottleneck of these transformations is the neighbourhood size.

References

SVM

sklearn.svm.SVC https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

For SA

Delahaye, D., Chaimatanan, S., & Mongeau, M. (2019). Simulated annealing: From basics to applications. In Handbook of metaheuristics (pp. 1-35). Springer, Cham. https://hal-enac.archives-ouvertes.fr/hal-01887543/document

Lin, S. W., Lee, Z. J., Chen, S. C., & Tseng, T. Y. (2008). Parameter determination of support vector machine and feature selection using simulated annealing approach. Applied soft computing, 8(4), 1505-1512. https://www.csie.ntu.edu.tw/~b92103/sdarticle.pdf

For VNS

- Boughaci, D., & Alkhawaldeh, A. A. (2018). Three local search-based methods for feature selection in credit scoring. Vietnam Journal of Computer Science, 5(2), 107–121. https://doi.org/10.1007/s40595-018-0107-y
- Helder, V. G., Filomena, T. P., Ferreira, L., & Kirch, G. (2022). Application of the VNS heuristic for feature selection in credit scoring problems. Machine Learning with Applications, 9, 100349. https://doi.org/10.1016/j.mlwa.2022.100349
- Caporossi, G., Hansen, P., & Mladenovic, N. (n.d.). Variable Neighborhood Search. 26.
- Hansen, P., Mladenović, N., Todosijević, R., & Hanafi, S. (2017). Variable neighborhood search: basics and variants. EURO Journal on Computational Optimization, 5(3), 423–454.
- Moumni, H. E. (2022). Scheduling problem in operational research. [Python]. https://github.com/SanELmoumni/Scheduling-problem-in-operational-research-Dynamic-programming-Metaheuristic-VNS-solutions. (Original work published 2019)
- Pereira, V. (2022). PyCombinatorial [Python]. https://github.com/Valdecy/pyCombinatorial (Original work published 2021)

For GA

- Leardi, R., Boggia, R., & Terrile, M. (1992). Genetic algorithms as a strategy for feature selection. Journal of chemometrics, 6(5), 267-281.
- Prakhargurawa/Feature-Selection-Using-Genetic Algorithmhttps://github.com/prakhargurawa/Feature-Selection-Using-Genetic-Algorithm
- mtbisca/genetic-feature-selection. https://github.com/mtbisca/genetic-feature-selection

Database

- Hofmann, H. (2022). UCI Machine Learning Repository–Statlog (German Credit Data)(1994). from https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)

- Hopkins, M., Reeber, E., & Forman, G. (n.d.). (2022), Index of /ml/machine-learning-databases/spambase. SPAM E-Mail Database., from https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/
- Peng, Y., Albuquerque, P. H. M., Kimura, H., & Saavedra, C. A. P. B. (2021). Feature selection and deep neural networks for stock price direction forecasting using technical analysis indicators. Machine Learning with Applications, Article 100060.