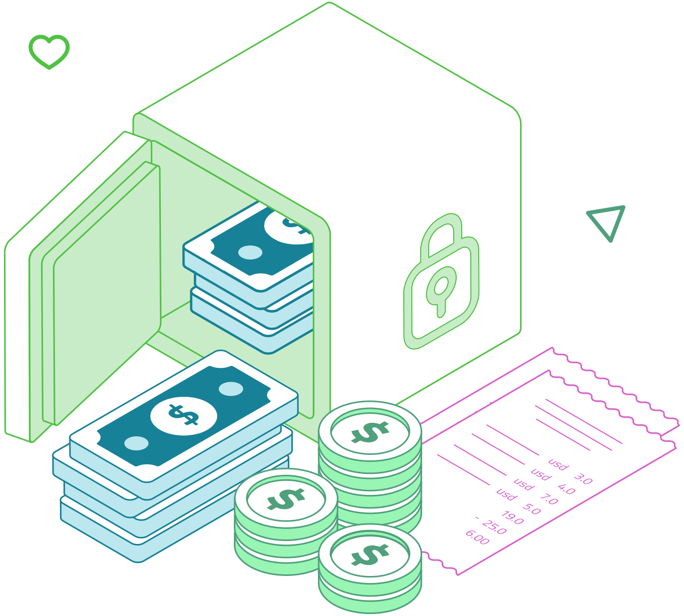


Banking Service Subscription Classification

Huỳnh Kim Hưng – 1952745
Lưu Chấn Hưng – 1952063
Trần Quốc Hoàn – 1952051
Lê Huy Hoàng – 1752209

Table of contents

Introduction

01

Motivation

In a modern economy, banks are to be considered not as dealers in money but as the leaders of development.
The banking system reflects the economic health of the country.

Telephonic marketing campaigns still remain one of the most effective way to reach out to people.

Motivation

Neural Network Architecture Optimization

Hyper parameters tuning for neural network ***is challenging***, especially in determine the architecture (number of layers, number of nodes for each layers)

Dataset

The data is related to the **direct marketing campaigns** of a Portuguese banking institution.

Target: Predict whether a customer will subscribe to the bank's campaign (**'yes'**) or not (**'no'**).

Dataset

The data folder contains two datasets:-

train.csv: 45,211 rows and 18 columns ordered by date (from May 2008 to November 2010)

test.csv: 4521 rows and 18 columns with 10% of the examples (4521), randomly selected from train.csv

Dataset

1. age (*numeric*)
2. job : type of job
(*categorical*:
"admin.","unknown","unemployed","management","housemaid","entrepreneur","student", "blue-collar","self-employed","retired","technician","services")
3. marital : marital status
(*categorical*: "married","divorced","single";
note: "divorced" means divorced or widowed)
4. education
(*categorical*: "unknown","secondary","primary","tertiary")
5. default: has credit in default?
(*binary*: "yes","no")

Dataset

6. balance: average yearly balance, in euros (*numeric*)
7. housing: has housing loan? (*binary*: "yes","no")
8. loan: has personal loan? (*binary*: "yes","no")

9. contact: contact communication type
(*categorical*: "unknown","telephone","cellular")
10. day: last contact day of the month (*numeric*)
11. month: last contact month of year (*categorical*: "jan", "feb", "mar", ..., "nov", "dec")
12. duration: last contact duration, in seconds (*numeric*)

Dataset

13. campaign: number of contacts performed during this campaign and for this client (*numeric*, includes last contact)
14. pdays: number of days that passed by after the client was last contacted from a previous campaign
(*numeric*, -1 means client was not previously contacted)
15. previous: number of contacts performed before this campaign and for this client (*numeric*)
16. poutcome: outcome of the previous marketing campaign
(*categorical*: "unknown","other","failure","success")

Output variable (desired target):
17. y - has the client subscribed a term deposit? (*binary*: "yes","no")

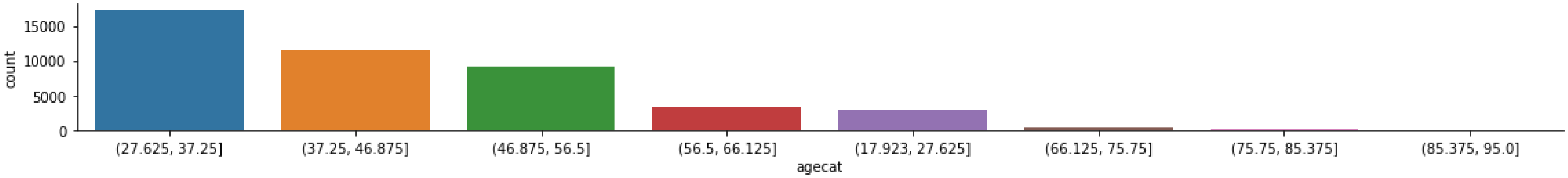
Explanatory Data Analysis

02

Data Pre-processing

Convert Numerical to Categorical data

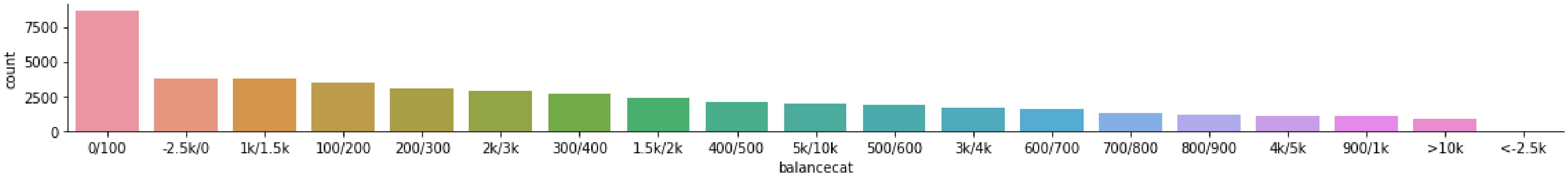
Age of the client data, we separate into 8 categories.



Data Pre-processing

Convert Numerical to Categorical data

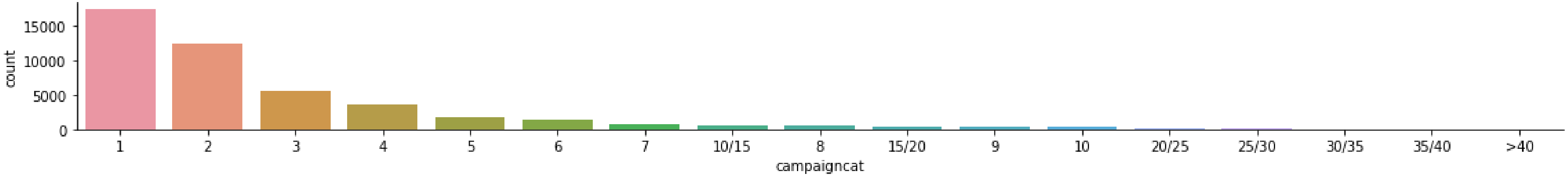
Data of ***customer's balance***, we separate into 19 categories



Data Pre-processing

Convert Numerical to Categorical data

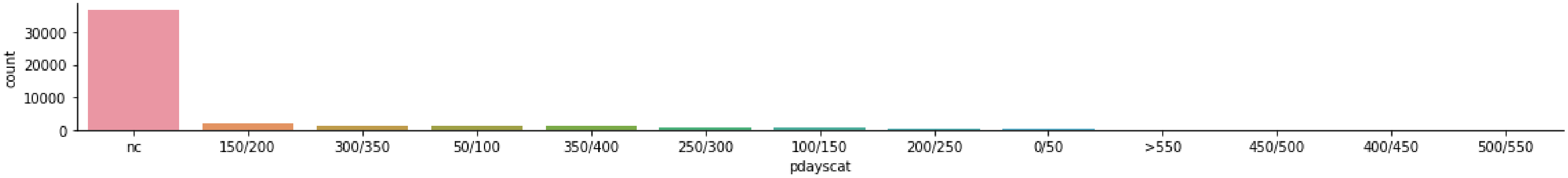
Number of ***contacts performed*** during this campaign to the client, we separate into 17 categories



Data Pre-processing

Convert Numerical to Categorical data

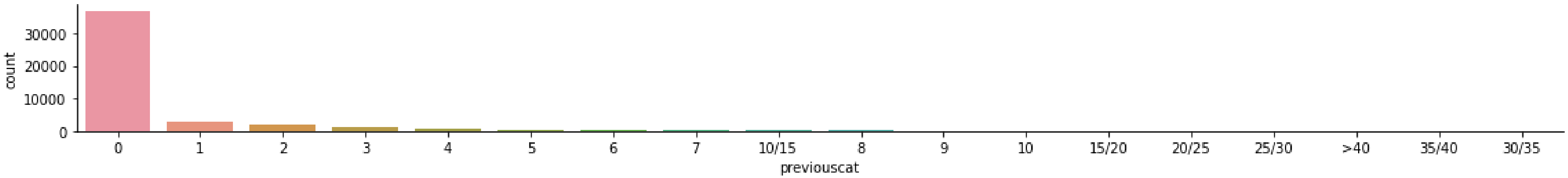
Number of ***days that passed by*** after the client was last contacted, we separate into 13 categories



Data Pre-processing

Convert Numerical to Categorical data

Number of ***contacts performed before this campaign***, we separate into 18 categories



Data Pre-processing

Treating Missing Data

The % of unknown in ***job*** is 0.63
The % of unknown in ***education*** is 4.10
The % of unknown in ***poutcome*** is 81.74
The % of unknown in ***contact*** is 28.79

Data Pre-processing

Fixing Job Unknown data

Primary education JOB mode is:
blue-collar 0.5485
retired 0.1160
housemaid 0.091
Secondary education JOB mode is:
blue-collar 0.2314
technician 0.2253
admin. 0.1818
Tertiary education JOB mode is:
management 0.5864
technician 0.1479
self-employed 0.0626

If job is unknown and education is primary, we will assign with **blue collar**

If job is unknown and education is secondary, we will **random the job**

If job is unknown and education is tertiary, we will assign with **management**

Data Pre-processing

Fixing Education Unknown Data

If job is in this list ['blue-collar', 'technician', 'admin.' ,'student', 'education', 'retired', 'services' ,'unemployed'] we assign **secondary.**

If job is in this list ['management','entrepreneur','self-employed'] we will assign **tertiary**

Housemaid job will be assign with **primary** education

Data Pre-processing

Fixing Contact Unknown Data

cellular	29236
unknown	12966
telephone	2882

We will assign unknown value with **cellular**

Data Pre-processing

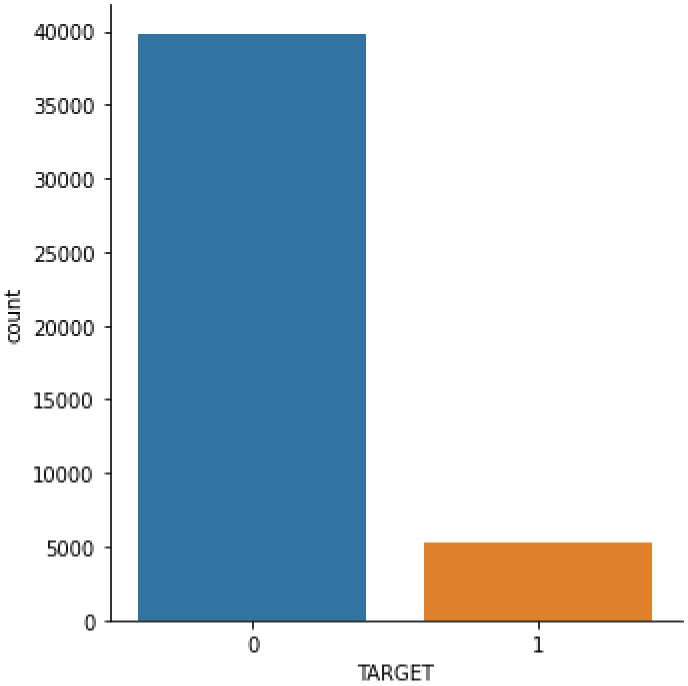
Fixing poutcome Unknown Data

unknown	81.72
failure	10.86
other	4.08
success	3.34

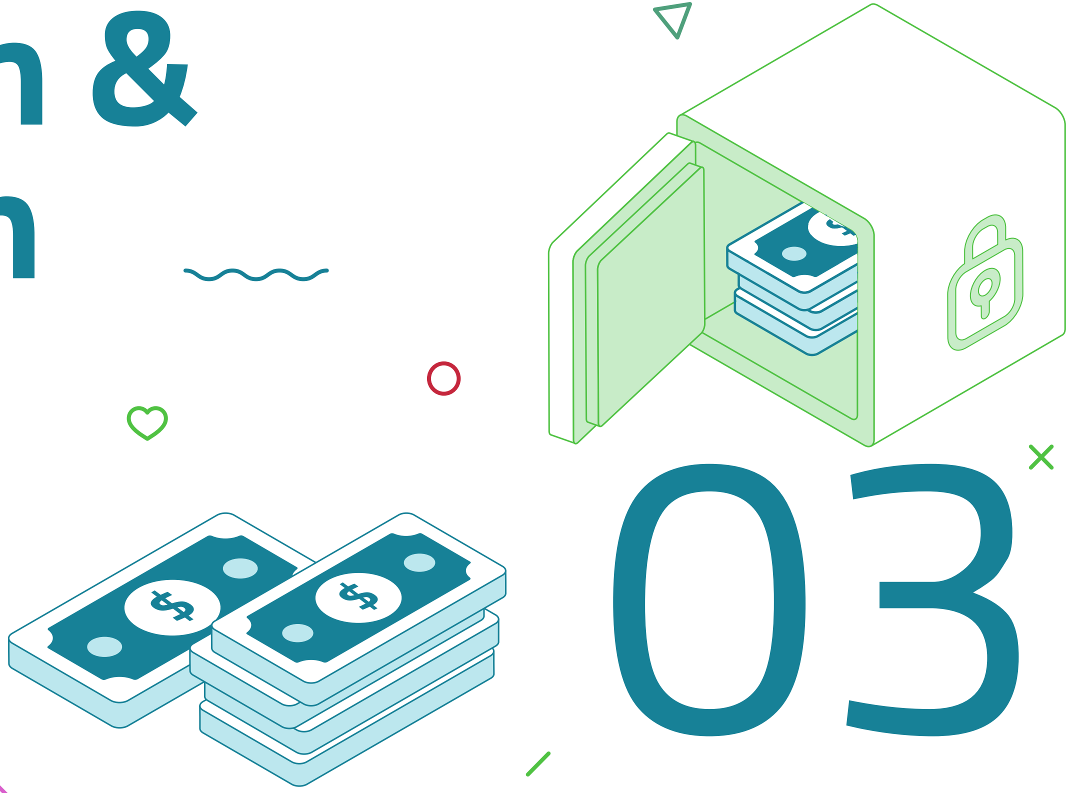
Since most of the poutcome is 'unknown' any attempt to replace them will bring a lot of BIAS so we'll **discard the column**

Data Pre-processing

“y” column information

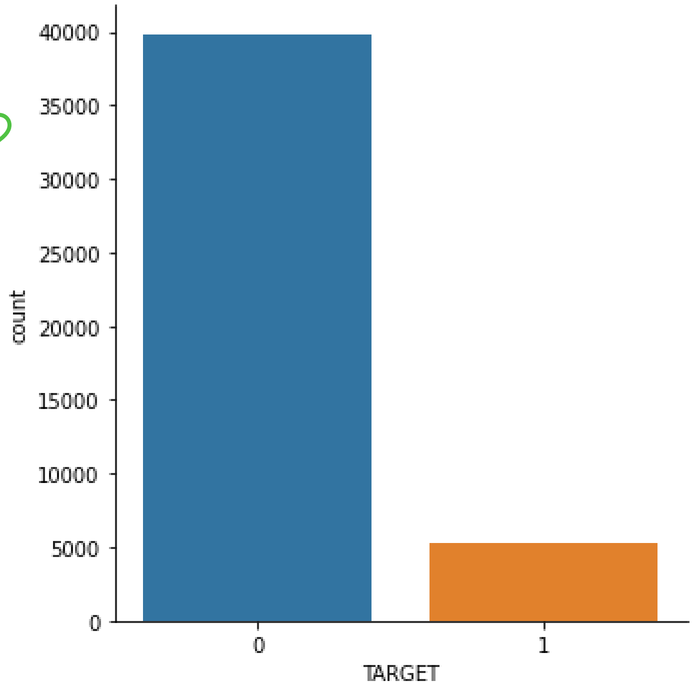


Problem & Solution



03

Unbalance Dataset

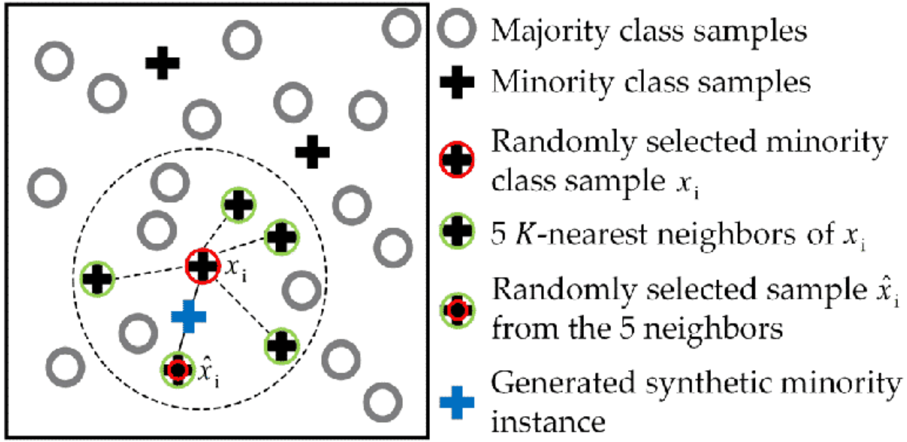


Only 10% of the data indicates a **1** outcome in the target column.

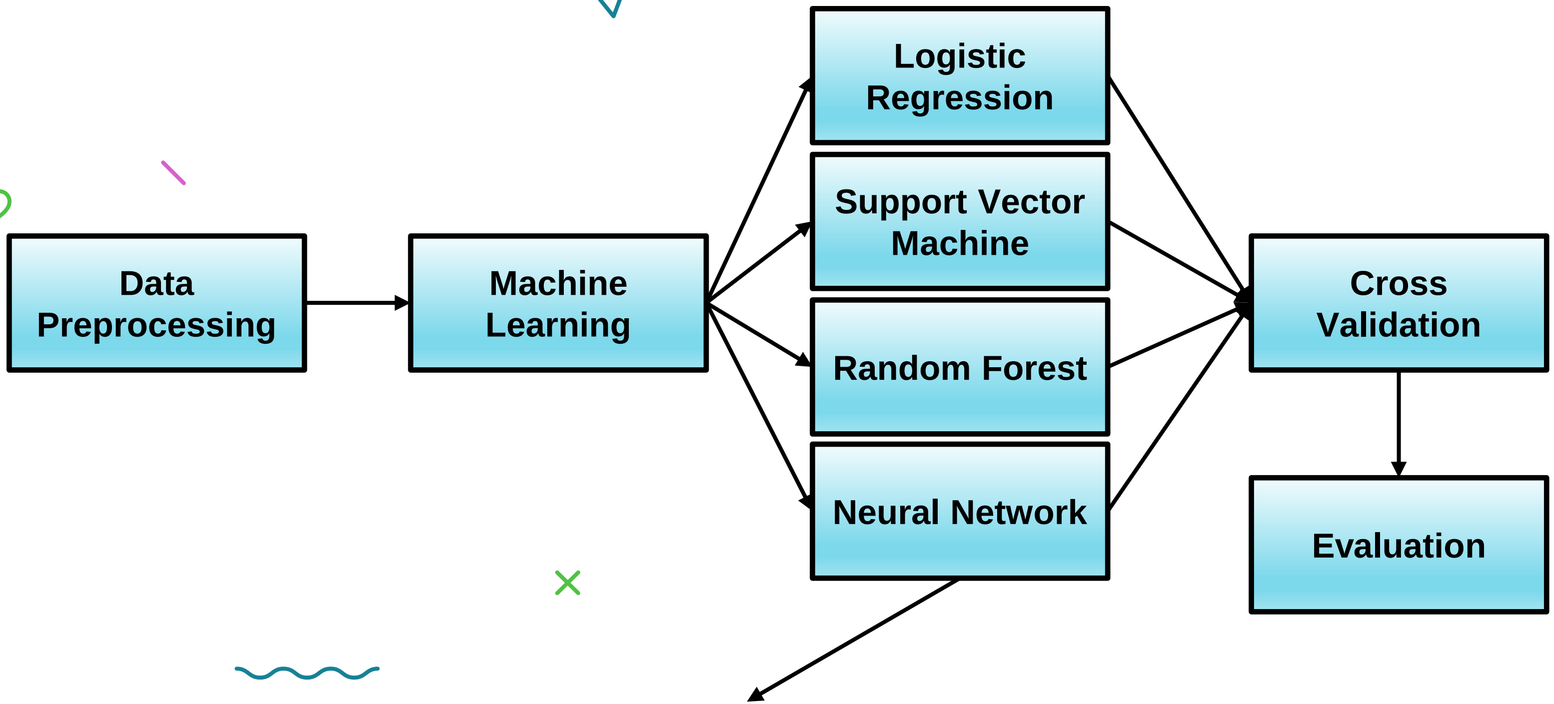
SMOTE

Synthetic Minority Oversampling Technique (SMOTE) is a statistical technique for increasing the number of cases in your dataset in a balanced way.

The component works by generating **new instances from existing minority cases** that supply as input**.**



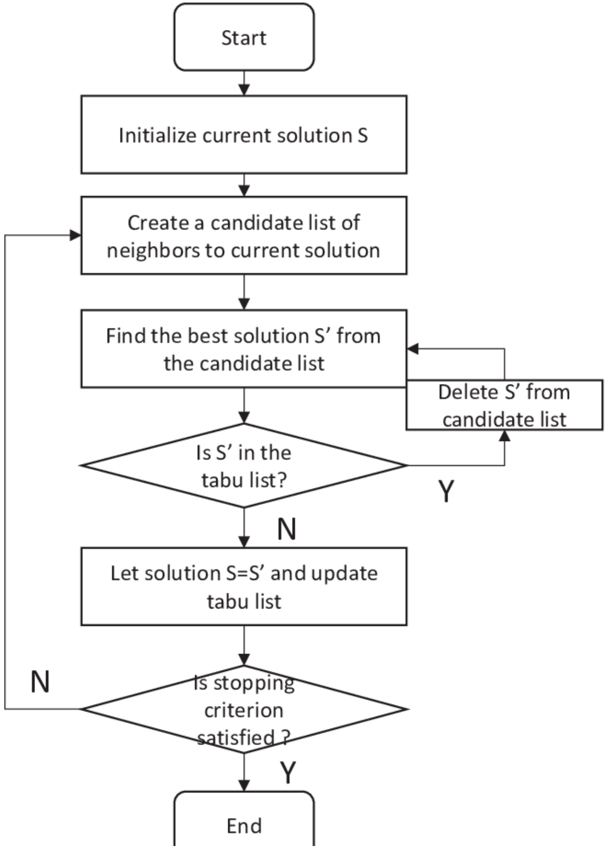
Propose Solution



Tabu search for Architecture Optimization + Cross validation for Model Selection

Tabu search

- A meta-heuristic search that overcome local optimum problems
- Is not problem specific
- **General Idea**:
    - First find an **optimum** in a local scope → Generate a new neighbor scope and find in them
⇒ Not stuck in local optimum
    - Have a **Tabu list**→ To ensure that we not visited the move we have already made



Architecture Optimization

```
INPUT: # input neurons, # output neurons, max, Iter
For H_L = 1 to max
{
    Input ← #input neurons
    Output ← # output neurons
    N_List[ ] ← NULL
    Tabu_List[ ] ← NULL
    For H_N = 1 to H_L
    {
        a ← (Input+Output)/2
        b ← ((Input+Output)×2)/3
        N_List[H_N] ← random(a,b)
        Input ← N_List[H_N]
    }
    s_0 ← calculate_fitness(H_L, N_List)
    s_0 is the initial solution update with s_best
    Tabu_List [ ] ← s_best
    For x = 1 to Iter
    {
        s' ← Generate_Neighbor(s_{x-1})
        s' is best from neighbor of s_{x-1}
        if f(s') < f(s_best)
        {
            s_best ← s'
            s_x ← s'
        }
        else
        {
            Tabu_List[next] ← s'
            s_x ← s'
        }
    }
    optimal[H_L] ← s_best    //list contains best architecture
}
Return best of optimal[H_L]
```

```
INPUT: P_max, p, K
Candidate_List[P_max] ← NULL
For i = 1 to (P_max/2)
{
            Ft ← NULL
    For j = 1 to H_L
    {
            ω = random(0,1)
            if ω ≥ p          // p is probability
     Increase number of neurons by 'K' at that layer
            else
                 No change with neurons at that layer
    }
    Update N_List[ ] for Candidate_List[i]
    Ft ← calculate_fitness(Candidate_List[i])
            Update Ft of Candidate_List[i]
}
For i = 1 to (P_max/2)
{
            Ft ← NULL

    For j = 1 to H_L
    {
            ω = random(0,1)
            if ω ≥ p       // p is probability
                 decrease number of neurons by 'K' at that layer
            else
                 No change with neurons at that layer
    }
    Update N_List[ ] for Candidate_List[i + P_max/2]
    Ft ← calculate_fitness(Candidate_List[i + P_max/2])
    Update Ft of Candidate_List[i + P_max/2]
}
Return best of candidate_List[P_max]
```

Implementation

Experiment setup

- Max number of layers: 5
- Number of iteration: 10
- Number of training for each neighbors: 20

Tabu search for best architecture for each number of layers
Cross validation to choose the best architecture

Run others classification method with Cross Validation

- Logistic Regression
- SVM
- Random Forest

Evaluation

Search Result

	Architecture	Accuracy	F1	Precision	Recall
1 Layer	45	0.9106	0.7458	0.6982	0.8634
2 Layers	59, 57	0.9334	0.7895	0.7669	0.8558
3 Layers	40, 36, 40	0.9305	0.7877	0.7585	0.8627
4 Layers	53, 48, 53 58	0.9427	0.8222	0.7805	0.9171
5 Layers	60, 51, 55, 55, 58	0.9432	0.8281	0.7951	0.9074

Evaluation

Cross Validation result for each Neural Network Architectures

	Accuracy(%)	**F1(%)**	**Precision(%)**	**Recall(%)**
1 Layer	89.93	67.72	65.07	77.76
2 Layers	91.42	69.76	66.31	80.83
3 Layers	91.13	68.96	66.13	78.93
4 Layers	91.90	70.04	67.06	79.38
5 Layers	92.20	71.04	68.16	80.68

Evaluation

Evaluation between Algorithms

	Accuracy	F1	Precision	Recall
Best NN Architecture	0.922	0.7105	0.682	0.807
Logistic Regression	0.838	0.534	0.399	0.805
SVM	0.829	0.527	0.386	0.83
Random Forest	0.997	0.986	0.998	0.975

Conclusion

For this dataset, after some preprocessing, the number of features is still really small for a neural network solution
→ Therefore, **Random Forest** out-perform **Neural Network**.

In fact, Random Forest is more commonly used in these kind of problems because its light-weight and can perform exceptionally.

However, **Tabu Search for Neural Network Architecture Optimization** have a big advantage in terms of scalability
⇒ If the problems are more complex, the number of features is huge ⇒ This method could potentially perform well.

Demo

04

References

Banking Dataset - Marketing Targets
https://www.kaggle.com/datasets/prakharrathi25/banking-dataset-marketing-targets

Gupta, T.K., Raza, K. (2018) *Optimizing Deep Feedforward Neural Network Architecture: A Tabu Search Based Approach*: https://link.springer.com/article/10.1007/s11063-020-10234-7

Thanks!

Do you have any questions?