INM427 Neural Computing Individual Project

Yumi Heo (Msc Data Science / 230003122 / [yumi.heo@city.ac.uk](mailto:yumi.heo@city.ac.uk))

**A Comparison of Multilayer Perceptrons and Support Vector Machines**

**for Bank Churn Prediction**

**Abstract**

1. **Description and Motivation**

Analysing customer churn in the bank is important for maintaining profitability, customer satisfaction, and competitiveness against other banks. Therefore, banks must enhance their ability to identify potential customer churn. This can be achieved using a supervised classification model based on neural networks or support vector machines (SVM), using customer and churn data. In this paper, multilayer perceptrons (MLP) and SVM have been utilised to conduct an experiment to find which of the two models better performs in classifying customer churn. By comparing and evaluating the performance of the two models, we will learn about the characteristics of the two algorithms and determine which model is has the higher accuracy for this task and which model is appropriate for this case. Eventually, the best model will enable banks to predict future churn from customer data and proactively improve services to retain their customers.

1. **Initial Analysis of Dataset**

This bank churn dataset is obtained from Kaggle [1]. It is a tabular dataset which has 165,034 samples with 14 features including the target feature. The target is a column named with ‘Exited’, which is binary. ‘1’ means to close their account and ‘0’ means to remain their account. The class of customer churn is always imbalanced. In order to handle this issue, since the number of samples is not very large, stratified k fold cross validation is planned to be used to perform model training and validation using a training set and validation set that balance the classes on both sides. this part will be covered again in the methodology.

Also, there is a large proportion of '0' in 'Balance'. It should be examined whether to remove the data with '0' in 'Balance' or not, as this data might affect the sensitivity of the class distribution. As a dataset having only 0 in ‘Balance’ and a dataset without 0 in ‘Balance’ have a similar distribution (Figure 1), it implies that the dataset with 0 in 'Balance is not significantly different from the dataset without 0 in 'Balance' in terms of its impact on the target.

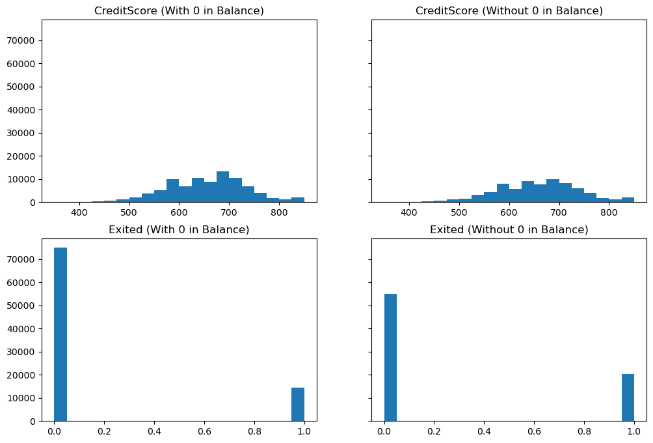


Figure 1. balance

Therefore, the data with the value of 0 in 'Balance' will be remained.In features, 'id', 'CustomerId' and 'Surname’ are deleted since those are irrelevant to predict whether customer will close their account or not.

Therefore, the total features used are 10. After checking Pearson correlation, 가장 높은 상관계수는 0.48로 모든 피쳐가 서로 낮은 상관 관계를 갖는다. 하여 추가적인 피쳐 삭제는 없이 총 10개의 피쳐를 모델링에 사용하기로 한다. 텍스트, 스크린샷, 사각형, 번호이(가) 표시된 사진

자동 생성된 설명

Figure 2. Pearson

1. **Hypothesis Statement**

Before comparing the two different models, the following three hypotheses were developed:

1. SVM takes longer to train than MLP because it is unsuitable for running multiple samples.
2. The accuracy of the SVM final model will be higher than that of MLP because SVM shows higher performance in classification tasks than MLP[MLP and SVM networks - a comparative study 참고문헌]
3. The SVM model will provide a more interpretable decision boundary using support vectors.

1. **Summary of the two ML methods with their pros and cons**
   1. **MLP**

Non-linearity: MLPs can learn complex non-linear relationships in data due to their ability to model intricate decision boundaries through multiple layers and activation functions.

Feature Learning: MLPs can automatically learn relevant features from raw data, reducing the need for manual feature engineering.

Versatility: MLPs can be applied to a wide range of tasks, including classification, regression, and even unsupervised learning tasks like clustering and dimensionality reduction.

Parallel Processing: Training of MLPs can be efficiently parallelized, especially with the use of modern computational frameworks and hardware accelerators like GPUs.

Cons:

Overfitting: MLPs are prone to overfitting, especially when dealing with small datasets or when the model is too complex relative to the amount of available training data.

Hyperparameter Sensitivity: Proper tuning of hyperparameters such as learning rate, number of hidden layers, and number of neurons per layer can be challenging and time-consuming.

Computationally Intensive: Training large MLPs with many layers and neurons can be computationally intensive, requiring substantial computational resources and time. **MLP is faster to train when the number of samples is large compared to SVM.**

Black Box Nature: Interpretability of MLP models can be challenging due to their complex, nonlinear nature, making it difficult to understand the reasoning behind predictions.

* 1. **SVM**

Effective in High-Dimensional Spaces: SVMs are effective in high-dimensional spaces, making them suitable for tasks with a large number of features, such as text classification and image recognition.

Margin Maximization: SVMs aim to maximize the margin between classes, which often results in models that generalize well to unseen data and are less prone to overfitting.

Kernel Trick: SVMs can efficiently handle non-linear decision boundaries through the use of kernel functions, allowing them to capture complex relationships in the data.

Sparsity: SVMs typically use only a subset of training data points (support vectors) in the decision function, making them memory efficient, especially for large datasets.

Cons:

Limited Scalability: SVMs can become computationally expensive and memory-intensive, especially when dealing with large datasets, as the time complexity of training SVMs is typically quadratic in the number of samples.

Sensitive to Noise: SVMs can be sensitive to noisy data and outliers, which can negatively impact their performance if not properly handled or preprocessed.

Difficulty in Parameter Tuning: SVMs have several hyperparameters that need to be tuned, such as the choice of kernel and regularization parameter, which can be challenging and require careful optimization.

Limited Interpretability: Similar to MLPs, SVMs can be considered as black box models, as it's often difficult to interpret the learned decision function, especially when using complex kernel functions.

1. **Methodology**

The dataset is divided into features and target. Afterwards, all data to be used for developing the MLP model is converted into PyTorch tensors and divided into 80% for training and 20% for testing. Since the SVM model must use an array, the training and test sets in the tensor state are converted to an array using new variables.

Since the target is imbalanced and the training samples are approximately 132,000, repeated stratified k fold cv is used. The difference between repeated stratified and stratified is that randomly split data can produce different results, so cv is repeated to produce an average result. And this average value provides a more accurate estimate of model performance. Here we break it into three sets and repeat twice. Thus, a total of 6 different models are fitted and evaluated. [Applied modeling reference]

Since cross validation has been decided, a base model for each algorithm is created.

During subsequent training and evaluation, both models use normalized data.

Because both models require data scaling.

For reference, normalization must be applied to the training set sample, and validation and test samples are performed only on transformers accordingly.

Find the optimal model through accuracy and other evaluation indicators through hyperparameter tuning.

Afterwards, the optimal model comparison of the two algorithms is examined through evaluation indicators such as ROC curve, AUC range, and accuracy using the test set.

Svm모델은 어레이를 써야 하므로 텐서 상태에서 나눠진 트레이닝 테스트셋은 다시 어레이로 바꾼 새로운 변수를 사용하여 트레이닝 및 평가한다

타겟이 imbalanced이고 트레이닝 샘플이 대략 132,000정도 이므로 repeated stratified k fold cv를 사용한다 repeated와 일반 stratified의 차이점은 일반이 무작위로 쪼개진 데이터는 각 다른 결과를 내올 수 있으므로 cv를 repeated하여 평균 결과를 내오는 것이다. 그리고 이 평균 값은 모델 성능을 더 정확하게 추정한다. 여기서 우리는 3개의 셋으로 쪼갠 후 2번 반복한다. 하여 총 6번의 각 다른 모델에 피팅 및 evaluation을 하는 것이다.[applied modeling 참고문헌]

크로스 밸리데이션까지 정해졌으므로 각 알고리즘의 베이스 모델을 만든다

이후 트레이닝 및 평가 시 **두 모델 모두 normalised 된 데이터를 사용한다**

**왜냐하면 두 모델 모두 데이터 스케일링이 필요한 모델이기 떄문**

**참고로 노멀라이제이션은 트레이닝셋 샘플에 적용해야 하며 밸리데이션과 테스트 샘플은 이에 맞게 트랜스포머만 진행한다**

하이퍼 파라미터 튜닝을 통해 정확도 및 이외 평가 지표를 통하여 최적의 모델을 찾는다

이후 두 알고리즘의 최적 모델 비교는 테스트 셋을 이용하여 ROC 커브 및 AUC 범위, 정확도 등 평가 지표를 통해 알아본다

1. **Choice of Parameters and Experimental Results**

**(e.g. cross-validation, choice of parameters and experimental results)**

**앞서 언급햇듯이 클래스가 imbalanced 하고 보다 노이즈가 적은 stratitifed cv를 위해 repeated stratified가 적용됨**

**이를 트레이닝셋에 적용하여 다시 트레이닝과 validation set divided**

**3 cross validation with 2 repetition total 6 running**

**MLP hyperparameter tuning**

**SVM**

**커널 선택 하이퍼파라미터로 감마 레귤러라이제이션 씨 설명**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Multilayer Perceptrons | | | | | | | | |
| No. | Hidden layers | Hidden units | Epochs | Learning rate | Weight decay | Test Accuracy | Training & Validation Time (sec) | Stage |
| 1 | 1 | 6 | 50 | 0.001 | 0 | 86.28 | 441.65 | baseline |
| 2 | 1 | 6 | 10 | 0.001 | 0 | 85.86 | 89.02 | Unit&LR  Grid search |
| 3 | 1 | 4 | 10 | 0.0001 | 0 | 83.55 | 86.78 | Unit&LR  Grid search |
| 4 | 1 | 6 | 10 | 0.001 | 0 | 85.87 | 88.86 | Weight decay  Grid search |
| 5 | 1 | 6 | 10 | 0.001 | 0.0001 | 85.88 | 89.38 | Weight decay  Grid search |
| 6 | 1 | 6 | 10 | 0.001 | 0.0001 | 85.44 | 207.54 | Step  Learning Scheduler |
| 7 | 1 | 6 | 10 | 0.001 | 0.0001 | 86.28 | 205.29 | Exponential  Learning Scheduler |
| 8 | 1 | 6 | 10 | 0.001 | 0.0001 | 86.20 | 201.99 | Linear  Learning Scheduler |
| 9 | 2 | (4, 4) | 50 | 0.001 | 0 | 86.47 | 499.48 | baseline |
| 10 | 2 | (6,4) | 12 | 0.01 | 0 | 86.31 | 119.94 | Unit&LR  Grid search |
| 11 | 2 | (6,4) | 20 | 0.01 | 0 | 85.89 | 199.45 | Increase the epochs |
| 12 | 2 | (4, 4) | 12 | 0.001 | 0 | 86.28 | 283.89 | Unit&LR  Grid search |
| 13 | 2 | (4, 4) | 12 | 0.01 | 0 | 85.66 | 271.33 | Unit&LR  Grid search |
| 14 | 2 | (6,4) | 12 | 0.01 | 0.1 | 78.93 | 283.90 | Weight decay  Grid search |
| 15 | 2 | (6,4) | 12 | 0.01 | 0.001 | 86.28 | 293.12 | Weight decay  Grid search |
| 16 | 2 | (6,4) | 12 | 0.01 | 0 | 78.93 | 273.42 | Step  Learning Scheduler |
| 17 | 2 | (6,4) | 12 | 0.01 | 0 | 86.31 | 271.62 | Exponential  Learning Scheduler |
| 18 | 2 | (6,4) | 12 | 0.01 | 0 | 86.32 | 263.23 | Linear  Learning Scheduler |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Support Vector Machines | | | | | | | |
| No. | Kernel | C | Gamma | Iteration | Test Accuracy | Training & Validation Time (sec) | Stage |
| 1 | rbf | 1.0 | scale | 49821 | 85.71 | 1149.42 | baseline |
| 2 | rbf | 1.0 | scale | 10000 | 56.51 | 656.45 | Manual Search |
| 3 | rbf | 1.0 | scale | 5000 | 47.47 | 395.31 | Manual Search |
| 4 | rbf | 1.0 | scale | 1000 | 73.20 | 156.19 | Manual Search |
| 5 | rbf | 1.0 | auto | 25112 | 84.57 | 2155.29 | Manual Search |
| 6 | rbf | 100 | 0.1 | 460215 | 85.73 | 1352.12 | **Halving grid search** |

1. **Analysis and Critical Evaluation of Results**

**각 최종 모델 러닝 커브 올리기**

**두 모델 성능 비교는 confusion matrix와 ROC로 가능할 듯**

**+ 각 모델의 프레시젼 리콜 커브도**

1. SVM takes longer to train than MLP because it is a model that is not suitable for running multiple samples.

정말 시간이 많이 걸림 시간 비교

하여 하이퍼파라미터 튜닝시 그리드/랜덤/하빙그리드 매뉴얼 서치 모두 다 적용함

허나 해당 모델은 이번 연구에 사용된 샘플의 수가 많다고 여기는지 sklearn의 svc로 시간이 너무 걸림

그래서 linearsvc를 이용하기도 함

Svc와 linearsvc의 차이는 정확도와 시간이 잇음

정확도는 rbf를 사용한 svc가 높앗지만 트레이닝은 00만큼 걸림

허나 linearsvc는 정확도가 낮앗지만 트레이닝 시간이 00으로 몇 배 단축됨

두 쪽 모두 하이퍼파라미터 튜닝을 하였으나 svc는 맥스 이터를 줄엿음에도 불구하고 유의미한 최적화된 하이퍼파라미터를 구하는데 00만큼의 시간이 걸림

또한 그 결과로 나온 하이퍼파라미터를 적용하여 테스트를 하였지만 디폴트인 c, gamma 값을 사용했을 때보다 낮은 수치가 나옴

Linear도 마찬가지였음

그에 비해 mlp는 트레이닝 시간이 상대적으로 덜 들었고 하이퍼파라미터 튜닝 시 빠르게 최적 하이퍼파라미터를 찾아냄

1. The accuracy of the SVM final model will be higher than that of MLP because SVM shows higher performance in classification tasks than MLP[MLP and SVM networks - a comparative study 참고문헌]

그러하지 않은 결과가 나옴

1. The SVM model will provide a more interpretable decision boundary using support vectors

C와 감마를 찾을 수 있음

Max iter가 어느정도인지도 알 수 있음

Mlp역시 weight와 bias는 찾을 수 있음

허나 왜 그러한 예측을 내렷는지 모델 설명이 어려움

그에 비해 svm은 디시젼 바운더리가 있음 서포트 벡터가 정해지므로 왜 그러한 예측이 나오는지 어느정도 설명이 가능해짐

1. **Lessons Learned**

**하이퍼 파라미터 튜닝 시 모델링에 사용한 라이브러리에 따라 맞는 라이브러리 안에서 튜닝하는게 안정성이 잇는 것 같음**

**파이토치로 짜고 scikit learn에서 다시 그리드 서치 하기엔 비슷한 정확도가 나오지 않음**

물론 데이터셋과 차원에 따라 mlp와 svm의 트레이닝 속도와 정확도는 차이가 날 수 있고 보편적으로 어느 쪽이 더 높은 정확도와 빠른 속도를 보이는지 얘기하긴 어려우나 해당 데이터셋으론 mlp가 우수한 속도와 정확도를 보여줌

허나 모델의 예측 결과에 대한 설명이 필요한 비즈니스에서 사용 시 mlp 모델은 상대의 이해도를 위해 한 번 더 사용을 고려해 봐야 할 모델임

또한 모델 해석이 더 중요하다면 굳이 테이블형 데이터에 뉴럴네트워크가 필요하지 않다

Svm의 경우 샘플이 피쳐에 비해 많으면 확실히 rbf 커널의 svm의 속도가 느리다

커널별 속도 분석

Linear가 가장 빠르고 sklear의 linsvc는 대용량 데이터셋을 위한 코드이므로 여타 커널 변경하여 만든 svm보다 훨 빠름 허나 rbf 커널에 비해 낮은 정확도를 보여줌

하지만 정확도와 속도를 trading off 관계로 하여 빠른 속도를 원한다면 linearSVM 사용도 고려해 볼 만함

1. **Future Work**

**Mlp의 액티베이션 펑션도 여러가지 적용해보기로**

Svm 트레이닝 및 하이퍼파라미터 튜닝에서 다소 긴 시간이 소요됨

하여 Pytorch 이용하여 내장 GPU를 사용하여 컴퓨팅 속도를 빠르게 해서 결과를 보기로

좀더 Fair comparison을 위해 skearn의 mlp 모델링도 고려해 볼 수 있음

Irrelevant라 여겼던 cutomer ID, surname을 넣었을때 안넣을때 ㅂ교 분석해 볼 수 있음

다른 모델을 썼으나 두 피쳐를 넣엇던 결과물이 더 좋음 [캐글 참고]

그러므로 두 경우의 수로 나누어 모델 정확도 분석해도 좋음

두 모델 모두 non linear을 띄었을 때 조금 더 높은 정확도를 보여주었으므로 두 모델을 합친 앙상블 기법을 사용하는 것도 좋음

Mlp로 피쳐 뽑고 svm 돌리기[참고문헌]

**Reference**

**[1] kaggle**

Comparison plots: lr curve & accuracy / prevision & recall curve(https://ai-com.tistory.com/entry/ML-%EB%B6%84%EB%A5%98-%EC%84%B1%EB%8A%A5-%EC%A7%80%ED%91%9C-Precision%EC%A0%95%EB%B0%80%EB%8F%84-Recall%EC%9E%AC%ED%98%84%EC%9C%A8)

Precision(정밀도)는 얼마나 정확하게 유저 이탈이라고 예측하는지에 대한 지표입니다. Recall(재현율)은 실제 이탈자에 대해서 얼마나 정확하게 이탈자라고 예측하는지에 대한 지표입니다.

lr어떻게 변화하는지 print 뽑기

시그모이드가 왜 output layer에 쓰이는지, tanh에 대한 설명(Activation Functions: Comparison of Trends in Practice and Research for Deep Learning)

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

https://machinelearningmastery.com/data-preparation-without-data-leakage/  
k fold cross validation batches train validation test set 설명

<https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/>

loss plot과 오버피팅 설명

pytorch로 모델링을 햇지만 gridsearch를 sklearn 패키지를 이요해서 햇으므로 정확성이 양 패키지 사이에 얼마나 있을지 모르겠음 future work로 냅둬

어차피 아담의 특징이 00이니 러닝레이트 스케줄러는 적용하지 않음

대신 그리드 서치 정도 적용

Multi core machine learning in python

Repeated k-fold cross validation for model evaluation in python

Support Vector Machine algorithms are not scale invariant, so **it is highly recommended to scale your data**. For example, scale each attribute on the input vector X to [0,1] or [-1,+1], or standardize it to have mean 0 and variance 1. Note that the same scaling must be applied to the test vector to obtain meaningful results. (<https://scikit-learn.org/stable/modules/svm.html>)

“

**Complexity of the Data: MLPs are known for their ability to learn complex nonlinear relationships in data, especially when dealing with high-dimensional datasets. If the data has intricate patterns that cannot be captured by linear decision boundaries, MLPs might perform better.**

**Interpretability: SVMs tend to provide more interpretable models, especially in cases where the kernel functions used are interpretable (e.g., linear kernel). MLPs, on the other hand, with their multiple hidden layers, might produce more complex models that are harder to interpret.**

**Training Time: Depending on the size of the dataset and the complexity of the model architecture, training an MLP can be more computationally intensive compared to training an SVM. SVMs are often more efficient when dealing with high-dimensional sparse data.**

**Regularization: SVMs naturally incorporate regularization through the choice of the margin parameter (C) and the kernel parameters. MLPs require explicit regularization techniques such as dropout or L2 regularization to prevent overfitting.**

**Robustness to Noise: SVMs can be more robust to noise and outliers in the data due to the margin-based optimization criterion. MLPs might be more susceptible to overfitting in the presence of noise if not properly regularized.**

“

**Reference**

[1] W. Reade and A. Chow, ‘Binary Classification with a Bank Churn Dataset’. Kaggle, Jan. 02, 2024. Accessed: Feb. 05, 2024. [Online]. Available: <https://www.kaggle.com/competitions/playground-series-s4e1>

**Appendix Ⅰ – Glossary**

**Appendix Ⅱ – Implementation Details**

Manual Search

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Multilayer Perceptrons | | | | | | | |
| No. | Hidden layers | Hidden units | Epochs | Learning rate | Weight decay | Test Accuracy | Training & Validation Time (sec) |
| 1 | 1 | 4 | 10 | 0.001 | 0 | 86.19 | 87.15 |
| 2 | 1 | 5 | 10 | 0.001 | 0 | 85.74 | 87.31 |
| 3 | 1 | 2 | 10 | 0.0001 | 0 | 85.60 | 86.92 |

Manual Search

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | Kernel | C | Gamma | Validation Accuracy | Training & Validation Time (sec) |
| 1 | linear | 1.0 | scale | 82.97 | 723.47 |
| 2 | poly | 1.0 | scale | 85.46 | 925.50 |
| 3 | sigmoid | 1.0 | scale | 68.10 | 1144.60 |
| 4 |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**LinearSVC**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | penalty | loss | C | max\_iter | Validation Accuracy | Training & Validation Time (sec) |
| 1 | *l2* | *squared\_hinge* | *1.0* | 1000 | 82.87 | 6.82 |
| 2 | l1 | *squared\_hinge* | *1.0* | 1000 | 82.87 | 4.12 |
| 3 | *l2* | hinge | *1.0* | 1000 | 83.06 | 1.41 |
| 4 | *l2* | *squared\_hinge* | 0.01 | 1000 | 82.80 | 1.32 |
| 5 | *l2* | hinge | 0.01 | 1000 | 79.41 | 1.32 |
| 6 | *l2* | *squared\_hinge* | 0.001 | 1000 | 82.30 | 0.95 |

**각 모델별 튜닝 어픛로치 설명**

**SVM 커널별 어큐러시 설명**

**테이블로 정리**