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 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 observations with 14 variables, which are 13 features and 1 target. The target is a column named ‘Exited’, which is binary. In the target, '1' means that a customer closed their account, and '0' means that a customer kept their account. The target is imbalanced since customer churn is 1/4 the rate of customer retention. Therefore, to resolve target imbalance, oversampling, undersampling, or stratified k-fold cross-validation can be considered before model training.

There is a large proportion of '0' in the feature variable '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. After checking the histogram of credit score and the target variable, a similar distribution has been found in a dataset containing only '0' in ‘Balance’ and a dataset without '0' in ‘Balance’ (Figure 1). This 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.

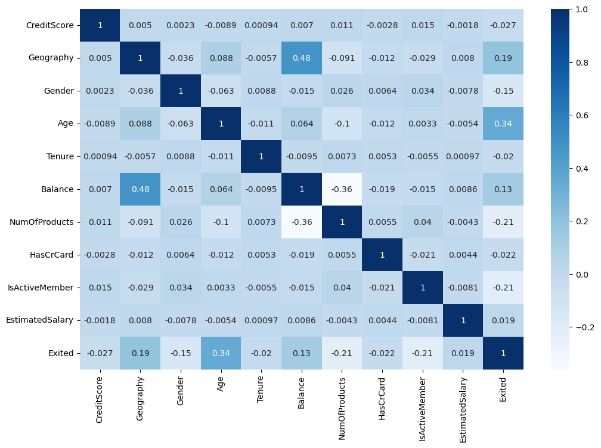


Figure 2. Correlation between all features and the target

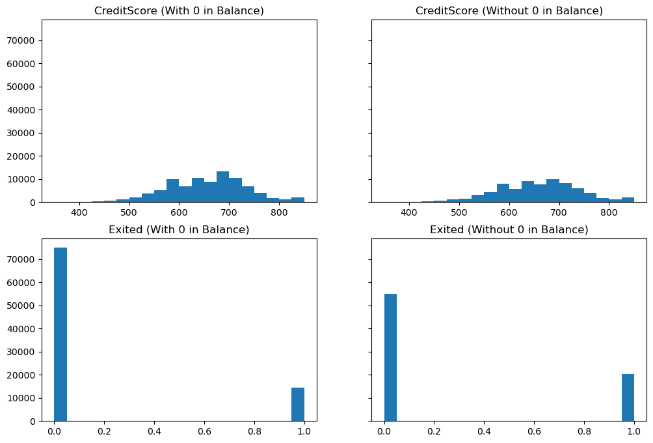


Figure 1. Histograms of credit score and the target

Therefore, the data with the value of 0 in 'Balance' has been retained. The features 'id', 'CustomerId', and 'Surname' have been removed as they are irrelevant to predicting whether a customer will close their account. To check the correlation coefficients between variables, Pearson correlation has been used (Figure 2). The highest correlation coefficient is 0.48, indicating that all features have low correlations with each other. Hence, 10 features will be used for modeling without any additional feature deletions.

Finally, considering the differing scale range of each feature, the data will be scaled between 0 and 1 after splitting it into train, validation, and test sets. This ensures stable results during the training of the MLP [2] and SVM models [3].

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 [4].
3. The SVM model will provide a more interpretable decision boundary using support vectors.

These three hypotheses will be confirmed through the process and results of the experiment.

1. **Summary of the MLP and SVM with their pros and cons**
   1. **MLP**

MLP is a supervised learning method and one of the neural networks used to address and solve non-linear problems [5]. This type of neural network is feedforward, consisting of fully connected layers with more than one hidden layer and units, the number of which can be controlled.

* + 1. **Pros**
* MLP learns and provides the result from various types of data such as tabular or time-series data, image or text [6].
* MLP is used to solve classification or regression tasks [7].
  + 1. **Cons**
* There's a risk of overfitting when the model contains a large number of weights, which can lead to it fitting the training data too closely [7].
* Determining and monitoring the values of weights is challenging [7]. If they're close to 0, the model may not learn effectively from the data, while excessively large weights can lead to poorer training performance.
* The interpretability of this model's reasoning is challenging due to its nonlinear activation functions and the opacity of its weights.
  1. **SVM**

SVM is a supervised learning method, based on statistical learning principles [8]. This method effectively addresses both linear and non-linear problems through the use of kernel tricks, which transform input data into higher-dimensional space to tackle the task.

* + 1. **Pros**
* SVM effectively captures linear or non-linear relationships in data by mapping it to higher dimensions, making it suitable for tasks with texts or images.
* It can be used for both classification [9] and regression [10] tasks
  + 1. **Cons**
* SVM is computationally expensive and impractical when dealing with large datasets because it calculates the kernel as a matrix, requiring significant memory storage [11].
* This method is sensitive to variations in feature scales, which can negatively affect its performance [3].

1. **Methodology**

All data used for developing the MLP model is converted into PyTorch tensors, while for the SVM model, the training and test sets in tensor state are converted to arrays using new variables. The original dataset is divided into 80% for training and 20% for testing. To handle the class imbalance issue, repeated stratified K-fold cross-validation is employed for model training and validation. This process repeats the cross-validation with different randomisations multiple times, ensuring each class's ratio is maintained. The training set is divided into three sets, creating a validation set using 20% of the training set, and the process is repeated twice, resulting in a total of 6 different models fitted and evaluated.

Subsequently, baseline models are created for each method. During training and evaluation, both models use normalised data, as both algorithms require data scaling. Normalisation is applied to the training set samples, while validation and test samples are transformed accordingly.

Following normalisation, hyperparameter tuning is performed through grid search or manual search. The best model is selected based on test accuracy and other evaluation metrics such as precision, recall, and F1 score during hyperparameter tuning.

Finally, the comparison of the best models from both algorithms is examined through evaluation metrics such as test accuracy, ROC curve, AUC range, Precision-Recall curve, and Average Precision score.

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

As the target classes are imbalanced, and repeated stratified K-fold cross-validation is applied, the accuracies of models created through this process are combined and averaged. Then, they are compared with the accuracy of each model using the holdout test set after tuning hyperparameters.

**MLP hyperparameter tuning**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Multilayer Perceptrons | | | | | | | | | | | | |
| No. | Hidden layers | Hidden units | Epochs | Learning rate | Weight decay | | Test Accuracy  (%) | | | | | Training & Validation Time (sec) |
| Baseline (one single hidden layer) | | | | | | | | | | | | |
| 1 | 1 | 6 | 50 | 0.001 | 0 | | 86.28 | | | | | 441.65 |
| Grid search regarding hidden units & learning rate | | | | | | | | | | | | |
| 2 | 1 | 6 | 10 | 0.001 | 0 | | 85.86 | | | | | 89.02 |
| 3 | 1 | 4 | 10 | 0.0001 | 0 | | 83.55 | | | | | 86.78 |
| Grid search regarding weight decay | | | | | | | | | | | | |
| 4 | 1 | 6 | 10 | 0.001 | 0 | | 85.87 | | | | | 88.86 |
| 5 | 1 | 6 | 10 | 0.001 | 0.0001 | | 85.88 | | | | | 89.38 |
| Application of step learning rate scheduler | | | | | | | | | | | | |
| 6 | 1 | 6 | 10 | 0.001 | 0.0001 | | 85.44 | | | | | 207.54 |
| Application of exponential learning rate scheduler | | | | | | | | | | | | |
| 7 | 1 | 6 | 10 | 0.001 | 0.0001 | | 86.28 | | | | | 205.29 |
| Application of linear learning rate scheduler | | | | | | | | | | | | |
| 8 | 1 | 6 | 10 | 0.001 | 0.0001 | | 86.20 | | | | | 201.99 |
| Baseline (two hidden layers) | | | | | | | | | | | | |
| 9 | 2 | (4, 4) | 50 | 0.001 | 0 | 86.47 | | | | | | 499.48 |
| Grid search regarding hidden units & learning rate | | | | | | | | | | | | |
| 10 | 2 | (6,4) | 12 | 0.01 | 0 | 86.31 | | | | | 119.94 | |
| Increase the epochs | | | | | | | | | | | | |
| 11 | 2 | (6,4) | 20 | 0.01 | 0 | 85.89 | | | | | 199.45 | |
| Grid search regarding hidden units & learning rate | | | | | | | | | | | | |
| 12 | 2 | (4, 4) | 12 | 0.001 | 0 | 86.28 | | | | 283.89 | | |
| 13 | 2 | (4, 4) | 12 | 0.01 | 0 | 85.66 | | | | 271.33 | | |
| Grid search regarding weight decay | | | | | | | | | | | | |
| 14 | 2 | (6,4) | 12 | 0.01 | 0.1 | 78.93 | | | | 283.90 | | |
| 15 | 2 | (6,4) | 12 | 0.01 | 0.001 | 86.28 | | | | 293.12 | | |
| Application of step learning rate scheduler | | | | | | | | | | | | |
| 16 | 2 | (6,4) | 12 | 0.01 | 0 | 78.93 | | | 273.42 | | | |
| Application of exponential learning rate scheduler | | | | | | | | | | | | |
| 17 | 2 | (6,4) | 12 | 0.01 | 0 | 86.31 | | 271.62 | | | | |
| Application of linear learning rate scheduler | | | | | | | | | | | | |
| 18 | 2 | (6,4) | 12 | 0.01 | 0 | 86.32 | | 263.23 | | | | |

**SVM**

**커널 선택 하이퍼파라미터로 감마 레귤러라이제이션 씨 설명**

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

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 계산된 weight와 bias는 찾을 수 있음

허나 왜 그러한 계산을 내렷는지 모델 설명이 어려움

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

도표, 그래프, 라인, 텍스트이(가) 표시된 사진

자동 생성된 설명

스크린샷, 텍스트, 그래프, 라인이(가) 표시된 사진

자동 생성된 설명

텍스트, 도표, 스크린샷, 그래프이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 그래프, 도표이(가) 표시된 사진

자동 생성된 설명

1. **Lessons Learned**
2. **MLP is faster to train when the number of samples is large compared to SVM.**

**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**

**Iteration과 정확도 관계성**

**Svm에 weight 적용**

**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]

얼리스타핑을 코드로도 구현하면 좋을 듯

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.

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

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 로버스트이니 모멘텀은 적용하지 않음

Multi core machine learning in python

Repeated k-fold cross validation for model evaluation in python

**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>

[2] Scikit-learn: Machine Learning in Python. ‘Neural network models (supervised), Tips on Practical Use’. *scikit-learn*. Accessed: Apr. 16, 2024. [Online]. Available: <https://scikit-learn.org/stable/modules/neural_networks_supervised.html>

[3] Scikit-learn: Machine Learning in Python. ‘Support Vector Machines, Tips on Practical Use’. *scikit-learn*. Accessed: Apr. 16, 2024. [Online]. Available: <https://scikit-learn.org/stable/modules/svm.html>

[4] S. Osowski, K. Siwek, and T. Markiewicz, ‘MLP and SVM Networks – a Comparative Study’. *Proceedings of the 6th Nordic Signal Processing Symposium.*, pp. 37-40, Jun. 2004.

[5] G. Cybenko, ‘Approximation by superpositions of a sigmoidal function’, *Math. Control Signal Systems*, vol. 2, no. 4, pp. 303–314, Dec. 1989, doi: [10.1007/BF02551274](https://doi.org/10.1007/BF02551274).

[6] J. Brownlee, ‘When to Use MLP, CNN, and RNN Neural Networks’, Machine Learning Mastery. Accessed: Apr. 16, 2024. [Online]. Available: <https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/>

[7] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, Second edition, Corrected at 12th printing 2017. in Springer series in statistics. New York, NY: Springer, 2017. doi: [10.1007/b94608](https://doi.org/10.1007/b94608).

[8] V. N. Vapnik, *The nature of statistical learning theory*, 2nd ed. in Statistics for engineering and information science. New York: Springer, 2000.

[9] C. Cortes and V. Vapnik, ‘Support-vector networks’, *Mach Learn*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: [10.1007/BF00994018](https://doi.org/10.1007/BF00994018).

[10] V. N. Vapnik, ‘The Support Vector method’, in *Artificial Neural Networks — ICANN’97*, vol. 1327, W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, Eds., in Lecture Notes in Computer Science, vol. 1327. , Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 261–271. doi: [10.1007/BFb0020166](https://doi.org/10.1007/BFb0020166).

[11] J. Cervantes, X. Li, W. Yu, and K. Li, ‘Support vector machine classification for large data sets via minimum enclosing ball clustering’, *Neurocomputing*, vol. 71, no. 4–6, pp. 611–619, Jan. 2008, doi: [10.1016/j.neucom.2007.07.028](https://doi.org/10.1016/j.neucom.2007.07.028).

**Appendix Ⅰ – Glossary**

|  |  |
| --- | --- |
| **Term** | **Definition** |
| MLP | Multilayer Perceptrons |
| SVM | Support Vector Machines |
| Observation | A single data point in a dataset |
| Feature | An input variable to make a model. |
| Target | A variable that a model will predict. |
| Variable | Here, this is an independent variable that can be controlled. |
| Balance | A feature in the dataset. This is a continuous scaled value since it is the balance of a bank account. |
| id | A feature in the dataset. This is the index of the dataset. |
| CustomerId | A feature in the dataset. This is the ID given to each customer. |
| Surname | A feature in the dataset. This is the last name of each customer. |
|  |  |
|  |  |
|  |  |
|  |  |

**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**