Comparing Multilayer Perceptron and Support Vector Machines methods applied on bank marketing dataset

Shiva taherzadehlari

Shiva.Taherzadehlari@city.ac.uk

Abstract

This paper aims to evaluate the performance of Multi-Layer Perceptron and Support Vector Machines applied on bank marketing dataset. The best possible models were achieved after applying grid search. The performance of the models was compared using confusion matrixes and ROC curve. A single hidden layer perceptron with number of neurons=13, learning rate =1 and the accuracy= 69% which was achieved in the previous research was considered as the baseline model. At the end, MLP performed better than SVM.

1. Introduction

Companies can better understand their historical performance, the market, and their competitors by analysing data from bank marketing campaigns. Using this information, suggestions may then be made for how to improve the bank's customer-attraction tactics and future marketing initiatives. In other words, studying bank marketing data can help identify efficient strategies and direct the creation of future profitable marketing initiatives for the bank. The purpose of this paper is to critically evaluate SVM and MLP models designed to solve a classification task being whether a customer will subscribe to a term deposit or not. In our paper, we have structured our findings into four sections. In Section 2, we introduce the dataset that we have used for training and testing our models. In Section 3, we discuss the different approaches and methods that we have implemented and compare them. In Section 4, we analyse the results of our implementation and critically compare the models. Finally, in Section 5, we mention the lessons learned and future work.

1.1 MLP

Artificial Neural Networks are used a lot in the field of artificial intelligence to solve problems that range from recognizing simple patterns to dealing with complex symbolic manipulation. The Multilayer Perceptron is one sort of artificial neural network that is frequently employed. It is used to address a variety of issues, including pattern recognition and forecasting using insufficient data. [1]. It performs information processing in a forward manner, taking a collection of inputs and producing a matching set of outputs. There are multiple layers of input nodes connected in a certain way that creates a directed graph which connects the input and output layers of an MLP. The network is trained using backpropagation, a popular deep learning technique that adjusts the weights of the connections between the nodes to enhance the accuracy of. MLP is a deep learning method that is widely utilised and very well-liked.[2]

Pros: 1) Flexibility: MLP models are very adaptable and can be used to address a variety of issues in different industries. 2) Non-linearity: MLP models can capture non-linear interactions between inputs and outputs, in order to accurately forecast complicated patterns in data. 3) Scalability: MLP models are advantageous for working with big data because they can manage large datasets and difficult problems by adding additional hidden layers and neurons.

4)Robustness: MLP models can handle noisy data and outliers to some extent, and can learn to generalize well on unseen data, making them useful for real-world application.

Cons: 1) Overfitting: there is a risk of overfitting for MLP model if the complexity of the model is not controlled, which can lead to poor generalization of new data. 2) Hyperparameter tuning: The performance of MLP models is highly dependent on the choice of hyperparameters such as the number of hidden layers, number of neurons in each layer, and learning rate. Tuning these hyperparameters can take a lot of time and requires expertise. 3) Slow convergence: MLP models can converge slowly, especially for large datasets, which can take a lot of time for training. 4)Difficult interpretability: MLP models can be difficult to interpret, making it hard to understand how the model arrived at its predictions. [3]

1.2 SVM

Support Vector Machines (SVM) are a type of machine learning algorithm that are used to sort and classify data. It's especially helpful when we want to categorize data but there isn't a clear line separating the categories. SVM works by creating a line, or hyperplane, that separates different categories of data in a high-dimensional space. The goal is to create a line that maximizes the distance between the closest data points from each category. SVM uses something called a kernel function to change the input data into a higher-dimensional space, making it easier to distinguish the categories. This is called kernel trick, and it's what makes SVM powerful when working with complicated and non-linear data.[4],[5].

Pros: 1) SVMs are very accurate and can perform well on a variety of datasets, including those with complex patterns and large feature spaces. 2) They perform well on datasets with a high number of features. 3) They can handle both linear and non-linear classification problems by using different kernel functions. 4) SVMs are generally robust to overfitting, as they minimize the margin error rather than the training error.

Cons: 1) When dealing with large datasets or complex models, SVMs can be computationally expensive. 2) They can be sensitive to the choice of the kernel function, not selecting the appropriate kernel can lead to poor performance. 3) SVMs can be difficult to interpret the results and understand how the model is making its predictions. 4) Training an SVM can be difficult when dealing with noisy or overlapping data.[6]

Hypothesis statement: we assume both models will perform better than a random guess. As well as this we assume MLP will outperform SVM since MLP has the capability to recognize non-linear and complex relationships between input and output variables, while SVM is considered to be more effective for linearly separable data.[7]

2. Dataset and initial analysis

The information contained in a dataset pertains to a marketing campaign conducted by a financial institution. The purpose of analyzing this data is to determine which strategies were effective and which were not, so that this information can be utilized to shape future marketing efforts. This dataset consists of 11,162 rows and 17 columns, with both categorical and numerical features and no missing values. The target variable is 'deposit', which can be either 'yes' or 'no' and indicates whether a customer has signed up for a term deposit. The target feature is balanced with 47.4% of values being 'yes' and 52.6% being 'no'. The table below presents basic statistical information about the numerical variables.

	age	balance	day	duration	campaign	pdays	previous
count	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000
mean	41.231948	1528.538524	15.658036	371.993818	2.508421	51.330407	0.832557
std	11.913369	3225.413326	8.420740	347.128386	2.722077	108.758282	2.292007
min	18.000000	-6847.000000	1.000000	2.000000	1.000000	-1.000000	0.000000
25%	32.000000	122.000000	8.000000	138.000000	1.000000	-1.000000	0.000000
50%	39.000000	550.000000	15.000000	255.000000	2.000000	-1.000000	0.000000
75%	49.000000	1708.000000	22.000000	496.000000	3.000000	20.750000	1.000000
max	95.000000	81204.000000	31.000000	3881.000000	63.000000	854.000000	58.000000

Table 1: basic statistics of the numerical variables

Before implementing the models, after encoding the categorical variables using the label encoding method, we normalized the numerical features to put all of them on the same scale.

3. Methods

In this section, the choice of training, evaluation, and testing sets, hyperparameters, and architecture of the models is explained.

3.1. Methodology

We split the data into 70% train set and 30% test set. From the 70% train set, we separated 10% as a validation set for validating in the process of model selection. The 30% test set was kept unseen until the algorithm comparison step. We specified ‘shuffle=True’, to make sure that samples are randomly distributed between the train, validation, and test sets.

Model selection: We trained the MLP model using the MLP classifier and the SVM model using the SVC function provided by the Scikit-Learn library. Then, the specified hyperparameters of both models were tuned using the ‘GridSearchCV’ function which takes the cross-validation strategy as one of the inputs for evaluating the model. For classification tasks, ‘GridSearchCV’ uses a ‘StratifiedKFold’ which is a type of cross-validation method. It is used to ensure that each fold (a subset of the data) represents the entire dataset. The data is divided into k folds, with each fold having almost the same proportion of samples from each target class as the complete set.[8] As we have a fairly large dataset, we chose k=5 to have a good balance between computational efficiency and reliable performance estimates. In the model selection process, other than getting the accuracy of the train set, we checked the accuracy on the separated validation set to compare the accuracies of train and validations sets and see if there is any signs of overfitting and whether we need to apply regularization techniques or not.

Algorithm comparison: After applying grid search and getting the best hyperparameters, both models were retrained using the train set and tested first on the validation set to check the need for regularization. Since the accuracy of the train and validation sets were so close for both models, we did not apply any regularization techniques (other than the default weight decay regularization which is used in this classifier with alpha=0.0001 and was enough for our case), and the same model was selected as the final model (we did not retrain the model using the train plus validation set to avoid data leakage) and tested on the 30% unseen test set. The best possible results were used to compare the performance of models.

3.2. Architecture and parameters used for MLP

As mentioned before, we used MLPclassifier to train our neural network. At the beginning, we trained our model using the default parameters without any sophistication. This classifier utilizes the ‘adam’ optimizer as the default algorithm for optimization. Adam is an adaptive optimization approach that updates the weights of the neural network during training using estimates of the first and second moments of the gradients it is known for its efficiency in terms of computation and its ability to handle large datasets [9]. The loss function used is binary cross-entropy loss function which ‘’compares each of the predicted probabilities to actual class output which can be either 0 or 1. It then calculates the score that penalizes the probabilities based on the distance from the expected value’’.[10]. Since the Relu activation function on the hidden layers and Sigmoid activation function on the output layer usually has the best performance,[11], we used the same for the initial and final model architectures. In the initial model, the only parameter that we did not use as default was the number of neurons in the hidden layer. Since our dataset has 17 features, and “The number of hidden neurons should be between the size of the input layer and the size of the output layer. The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer”, [12], and also because in our baseline model the best accuracy was achieved using 13 neurons, we changed the default 100 number of neurons to 13. Then we experimented the performance on the train and validation sets by having one, two, three, and four hidden layers and keeping other parameters constant. Changing to two hidden layers from one hidden layer led to improvement in the accuracy of both sets however changing to three and four hidden layers did not affect the accuracy significantly. So, in the Gridsearch we only experimented with two hidden layers but different numbers of neurons.

3.3. Architecture and parameters used for SVM

Same as we did for MLP, we started by creating a basic SVM model using the default parameters provided in Scikit-learn. These defaults are usually a good place to start, and for SVM they are c=1, kernel=’rbf’, and gamma=’auto’. The kernel function is the first parameter to be specified in an SVM, and it determines how similarities are calculated between pairs of input data points. The box constraint (C) is a regularization parameter that determines the balance between minimizing training error and minimizing testing error, by regulating the amount of misclassification allowed. The gamma parameter, which is specific to the kernel function, determines the width of the kernel.[13] Once we trained the initial model, we used grid search to find the best hyperparameters for our model.

4. Results, findings, and evaluation

4.1 Model selection

The hyperparameters varied in the MLP model were the number of neurons in hidden layers, learning rate, momentum, number of epochs, and batch sizes. These hyperparameters have an impact on the model's capacity and ability to learn complicated patterns, the speed, and accuracy of the optimization process, the model's ability to overcome local minima and make the optimization process faster, the ability of the model to learn complicated patterns and the risk of overfitting, and stochasticity of the optimization process and the training speed respectively. In the case of SVM, we varied kernel, C, and gamma values which were explained in the section 3.3. In the SVM grid search, we used a logarithmic scale for C and gamma hyperparameters to reduce the number of values that needed to be tested, therefore making the

grid search faster and more efficient since in the linear scale, the grid search would test every single value within the given range. The best set of hyperparameters for MLP varied every time we run the grid search. This happens because the best combination of parameters can depend on the random initialization of the weights in the neural network. To avoid getting stuck in a local minimum during training, the weights are usually initialized randomly. As a result, different initializations can lead to different outcomes [14]. The hyperparameters that were used to get the following results were: batch_size=100, hidden_layer_sizes=(13,11), learning rate= 0.01, epoch=1000, momentum=0.5. For SVM however, the hyperparameters were the same after each time running the grid search being C=10, gamma=0.1, and kernel=rbf. Time wise, MLP gridsearch took 8.5 times more than SVM to run (MLP: 119 mins, SVM: 14 mins). This can be because of the range of the hyperparameters. MLP was searched over 6 hyperparameters with the total of 216 possible combinations, while the SVM was searched over 3 hyperparameters with overall 30 possible combinations.

4.2 Algorithm comparison

The accuracies of training, validation, and test sets of the best MLP model were 81%, 81%, and 80% respectively. In the case of the best SVM model, the training, validation, and test accuracies were 84%, 80%, and 78%. We can see that even though the accuracy of the SVM model is higher on the train set, with a small difference, the MLP model generalizes better on the unseen data. Moreover, both models are performing better than our baseline (69%). Figures 1 and 2 show the confusion matrix of the test set of both models. We can see that the number of false predictions (FP+FN) of SVM model is 729 and for MLP is 674 meaning that SVM is misclassifying almost 2% more than MLP. Table 2 illustrates the average value of performance metrices for the test set. Considering our dataset and the goal of this classification which is whether a customer will subscribe to a term deposit or not, other than accuracy, precision and recall values are also equally important. The reason is that we do not want to overestimate the number of customers saying ‘yes’ to the subscription because in this case, the bank will consider more money than it is needed for paying the interest rate which is not efficient especially in the cases where the resources are limited. Overestimating in this context refers to high number of false positives. Regarding the precision formula (TP/TP+FP), a higher precision indicates lower number of false positives which is what we look for. We can see that in this case MLP is performing better. On the other hand, we also do not want to underestimate the number of customers subscribing to a term deposit. This can lead to not having enough budget to pay the interest rate of all customers which can result in unsatisfactory of customers and losing them. Underestimating here is related to high number of false negatives. According to the recall formula (TP/TP+FN), a higher recall means lower number of false negatives which is of our interest. From table 2 we can see that in this case also MLP is performing better. As we said both values of precision and recall are important in our analysis, F1 score can be another appropriate performance metric to compare our models since it takes a harmonic mean of both precision and recall values. In this case again MLP is performing better with its F1 score value being 2% more than that of SVM.

	Average Precision	Average Recall	Average F1 Score	Accuracy
MLP	0.80	0.80	0.80	0.80
SVM	0.78	0.785	0.78	0.78

Table 2: AVG performance metrices for test set

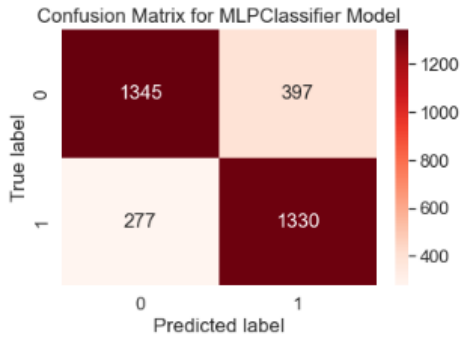


Figure 1

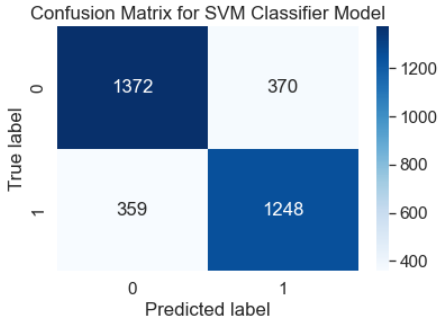


Figure 2

ROC curves were created as another way to evaluate the performance of our classification models. The closer the ROC curve is to the top left corner of the plot (indicating high true positive rates and low false positive rates for various threshold values), the better is the performance of the classifier. To have a better comparison we plotted the ROC curves of both models for class 1. We can see that the performance of both models is good and they both performed better than a random guess as we mentioned in the hypothesis. However, MLP model performed better since it has 0.88 area under the curve (AUC) being 0.02 more than that of SVM. The higher the AUC is the better the model at predicting the class 0 as 0 and the class 1 as 1.[15].

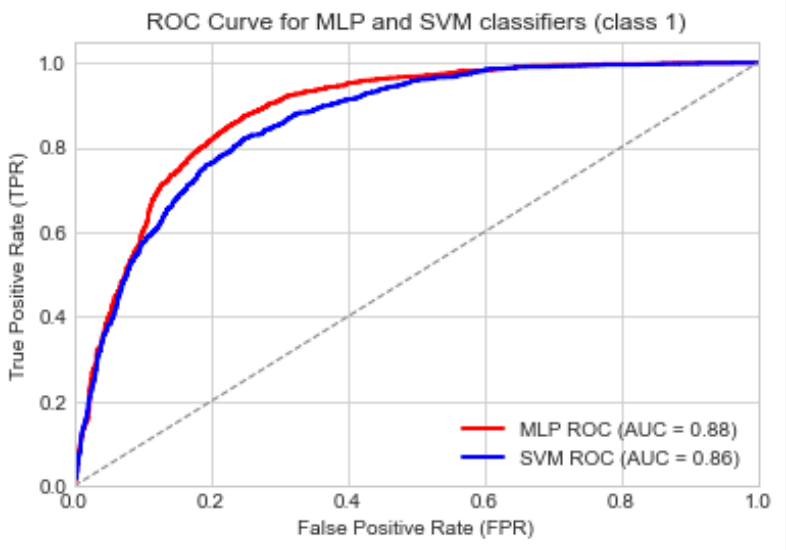


Figure 3

5. Conclusion

Our study aimed to compare which of the SVM or MLP models can better perform on the unseen data and predict whether a customer would subscribe to a term deposit or not. In conclusion, our hypothesis statement was true meaning both models performed better than a random guess with MLP performing better (80% accuracy). Moreover, both models had higher accuracy on the train set as expected.

Lessons learned: 1) In the case of MLP, it is important to notice that increasing the number of hidden layers would not necessarily lead to an increase in accuracy. 2) MLP generalized better than SVM on the unseen data even though the accuracy of SVM on the train set was higher. 3) because of the random weights that are initialized every time, running the MLP model each time would give different accuracies and different best parameters as a result of grid search. 4)When evaluating the performance of a model, apart from getting the accuracy, it is always important to consider what our data is about and what goal of it is and evaluate the model based on the relevant performance metric.

Future work: 1) voting classifier can be applied to combine the optimized SVM model with other models such as the decision tree and k nearest neighbours, to make a final prediction which can lead to a better performance than using only one classifier. 2)Exploring the use of 'polynomial' kernel in the case of SVM model. it was not used in our project to make the grid search run faster.

6. References

[1]. Noriega, L. (2005). Multilayer perceptron tutorial. School of Computing. Staffordshire University, 4, 5.

[2]. Multilayer Perceptron (https://www.techopedia.com/definition/20879/multilayer-perceptron-mlp)

[3]. LeCun, Y., Bengio, Y., & Hinton, G. (2015), Deep learning Nature, 521(7553), 436-444. (https://www.nature.com/articles/nature14539)

[4]. Vapnik, V. N. (1995), The nature of statistical learning theory, Springer Science & Business Media.

[5]. Schölkopf, B., & Smola, A. (2002). Learning with kernels: Support vector machines, regularization, optimization, and beyond. MIT press.

[6]. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning-Data Mining, Inference, and Prediction Springer.

[7]. Byvatov E, Fechner U, Sadowski J, Schneider G. Comparison of support vector machine and artificial neural network systems for drug/nondrug classification. J Chem Inf Comput Sci. 2003 Nov-Dec;43(6):1882-9. doi: 10.1021/ci0341161. PMID: 14632437.

[8]. The scikit-learn documentation on StratifiedKFold: (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html)

[9]. Kingma, D. P., & Ba, J. (2014). Adam, A method for stochastic optimization, arXiv preprint arXiv,1412.6980.

[10]. Binary Cross Entropy/Log Loss for Binary Classification, (https://www.analyticsvidhya.com/blog/2021/03/binary-cross-entropy-log-loss-for-binary-classification/).

[11]. How to Choose the Right Activation Function for Neural Networks(https://towardsdatascience.com/how-to-choose-the-right-activation-function-for-neural-networks-3941ff0e6f9c#:~:text=In%20a%20binary%20classifier%2C%20we,with%20one%20node%20per%20clas)

[12]. Heaton, Jeff. "The Number of Hidden Layers." Heaton Research, 28 Dec. 2018, (https://www.heatonresearch.com/2017/06/01/hidden-layers.html)

[13]. Scikit-learn documentation on SVM: (https://scikit-learn.org/stable/modules/svm.html)

[14]. Weight initialization techniques in neural network (https://towardsdatascience.com/weight-initialization-techniques-in-neural-networks-26c649eb3b78)

[15]. Understanding AUC-ROC curve (https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5)

Appendix 1, Glossary:

Perceptron: is a type of neural network that can make decisions according to the input data it receives.

Confusion matrix: is a table being used to evaluate how well a machine learning model is performing. In this table, the actual values to the predicted values can be compared to see how accurate the model is.

ROC curve: is a graph showing how accurate a binary classification model is. It is used to compare the true positive rate and false positive rate.

Overfitting: This happens when a model is too complicated and memorizes the training data instead of learning the patterns which results in poor performance when we use new unseen data.

Generalization: is the ability of a model to perform well on new, similar data to what it was trained on.

Hyperparameters: are parameters that we set before training the model. hyperparameters control the learning process, such as the number of hidden layers.

Regularization: is a technique that is used to prevent overfitting. We add a penalty term to the loss function to keep the model from memorizing the training data.

Hidden layers: are layers in a neural network that learn complex patterns.

Epoch: is a complete pass through the training data during the model's training process.

Batch size: is the number of training examples used in one iteration of the training process.
Learning rate: is the rate at which a model adjusts its parameters during training.

Momentum: is a technique being used to speed up training by adding a fraction of the previous parameter update to the current update.

Baseline model: This is a simple model that is used to compare the performance of more complicated models.

Local minima: These are points in the model's parameter space where although loss function is locally minimized, it may not be the global minimum that we are looking for.

Appendix 2, implementation details, and intermediate results:

Implementation details: For MLP grid search the range of hyperparameters that we experimented with were as follows:

'hidden_layer_sizes': [(11, 13,),(11,15,),(13,15,),(13, 11,),(15,11,),(15,13,)],

'learning_rate_init': [0.01,0.1,1],

'momentum': [0.1,0.5,0.9],

'max_iter': [100,500,1000,2000],

batch_size': [100, 200, 500,1000]

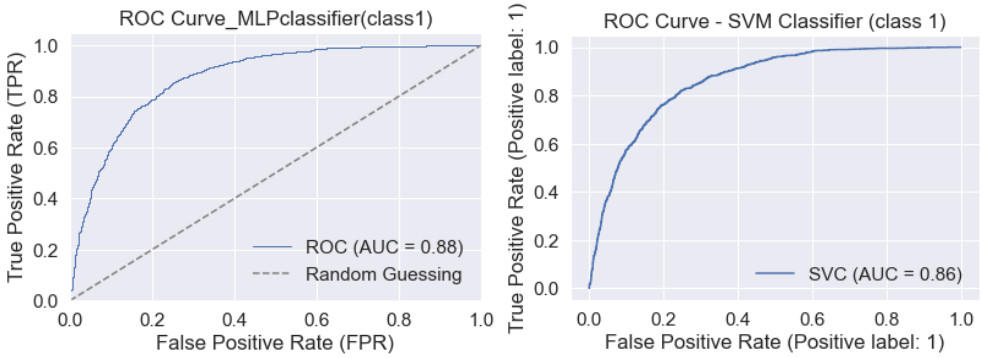
for SVM:

C= 0.001,10,100

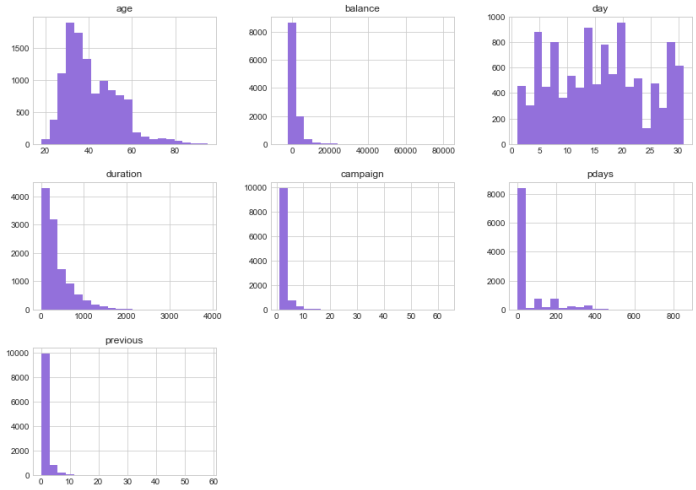
Gamma=0.001,10,100

Kernel='rbf','linear','sigmoid'

Intermediate results: 1) For MLP grid search, at first, we applied two different grid searches, one of which involved trying with hidden layer sizes, learning rate, momentum, and solver and the other one trying with number of epochs, batch sizes and activation function. We did it in one gridsearch took a lot of time to run but when we did it separately it did not take that long. However, we realised that doing two separate grid searches may not be the best approach because these hyperparameter affect each other and they should be assessed together. For example, learning rate and batch size, a higher learning rate might need a smaller batch size otherwise there would be the risk of overshooting the optimal weights during training. As a result, we combined these two grid searches but tried with fewer hyperparameters and smaller ranges so that it run faster.



2) At first, the above two ROC curves were made separately, but then to be able to compare better we combined them into one graph.



3) in the preprocessing step, these histograms were created to show the distribution of the numerical features.

4) before doing the grid search for SVM, we played around the parameters manually, trying with different kernels and C values. The code for this part is at the last section of the notebook.

The code at the last section of the notebook is related to the intermediate implementations and results that were not used.