Session – 5

Cross Validation

Why Do Models Lose Stability?

Any machine learning model needs to consistently predict the correct output across a variation of different input values, present in different datasets. This characteristic of a machine learning model is called stability. If a model does not change much when the input data is modified, it means that it has been trained well to generalize and find patterns in our data. A model can lose stability in two ways:

1. Underfitting: It occurs when the model does not fit properly to training data. It does not find patterns in the data and hence when it is given new data to predict, it cannot find patterns in it too. It under-performs on both known and unseen data.

2. Overfitting: When the model trains well on training data and generalizes to it, but fails to perform on new, unseen data. It captures every little variation in training data and cannot perform on data that does not have the same variations.

The figures depicted below show unfit, overfit, and optimally fit models:

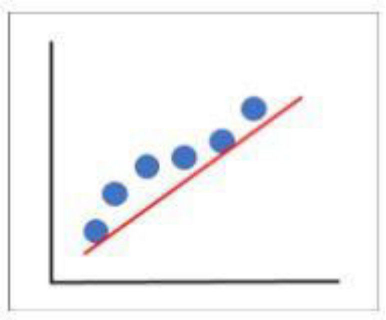


Figure 1: Underfitting

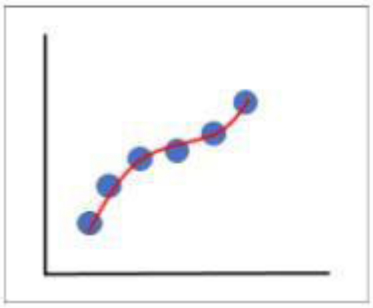


Figure 2: Overfitting

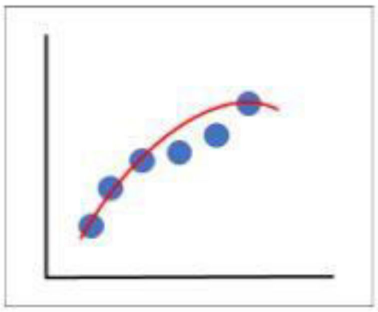


Figure 3: Optimal Model

In Figure 1, we can see that the model does not quite capture all the features of our data and leaves out some important data points. This model has generalized our data too much and is under fitted. In Figure 2, our model has captured every single aspect of the data, including the noise. If we were to give it a different dataset, it would not be able to predict it as it is too specific to our training data, hence it is overfitted. In figure 3, the model captures the intricacies of our model while ignoring the noise, this model is our optimal model.

What is Cross-Validation?

While choosing machine learning models, we need to compare models to see how different models perform on our dataset and to choose the best model for our data. However, data is usually limited, our dataset might not have enough data points or may even have missing or wrong data. Further, if we have fewer data, training, and testing on the same portion of data does not give us an accurate view of how our model performs. Training a model on the same data means that the model will eventually learn well for only that data and fail on new data, this is called overfitting. This is where cross-validation comes into the picture.

Cross-Validation in machine learning is a technique that is used to train and evaluate our model on a portion of our database, before re-portioning our dataset and evaluating it on the new portions.

This means that instead of splitting our dataset into two parts, one to train on and another to test on, we split our dataset into multiple portions, train on some of these and use the rest to test on. We then use a different portion to train and test our model on. This ensures that our model is training and testing on new data at every new step.

This also exposes our model to minority classes which may be present in the data. If we split our data into two and train only on one part, there is a chance that the test data contains a minority class that was not present in the testing data. In this case, our model will still perform well as the class constitutes only a small portion of the dataset but it will be desensitized to that data.

Consider the block below to represent the entirety of our data. We partition the dataset into training and testing data. The training data will be used by our model to learn. The testing dataset will be used by our model to predict unseen data. It is used to evaluate our model's performance.

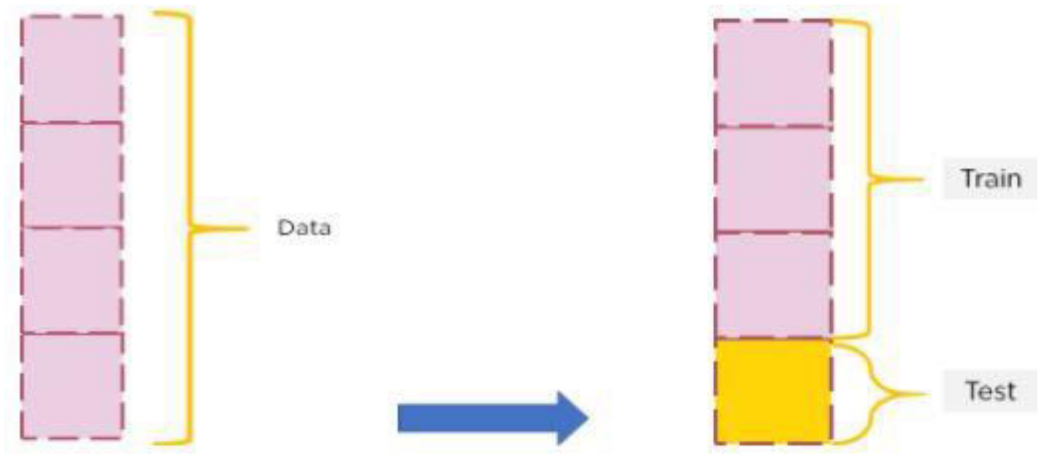


Figure 4: Partitioning dataset for cross-validation

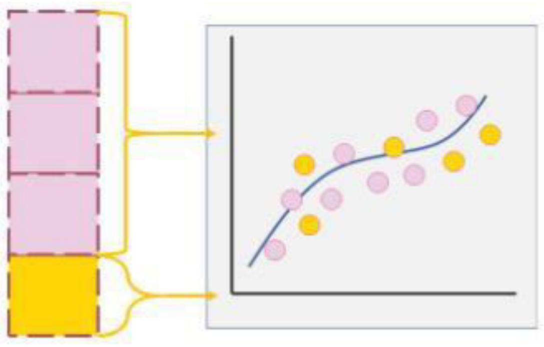


Figure 5: Training and testing with our portioned dataset

We then choose a different portion to test on and use the other portions for training. Then, the model performance is re-evaluated with the results obtained from the new portioned dataset to get better results.

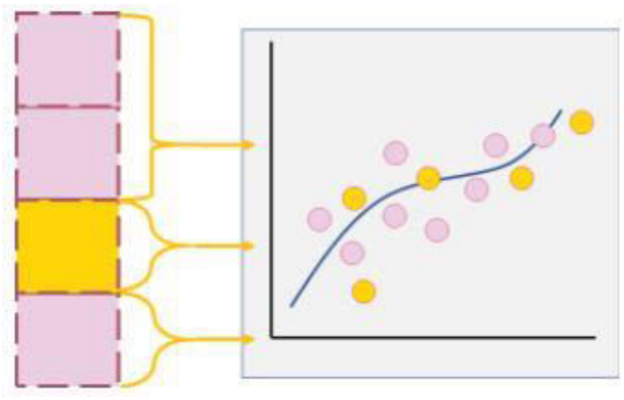


Figure 6: Training and testing on new portions

Steps in Cross-Validation

Step 1: Split the data into train and test sets and evaluate the model's performance

The first step involves partitioning our dataset and evaluating the partitions. The output measure of accuracy obtained on the first partitioning is noted.

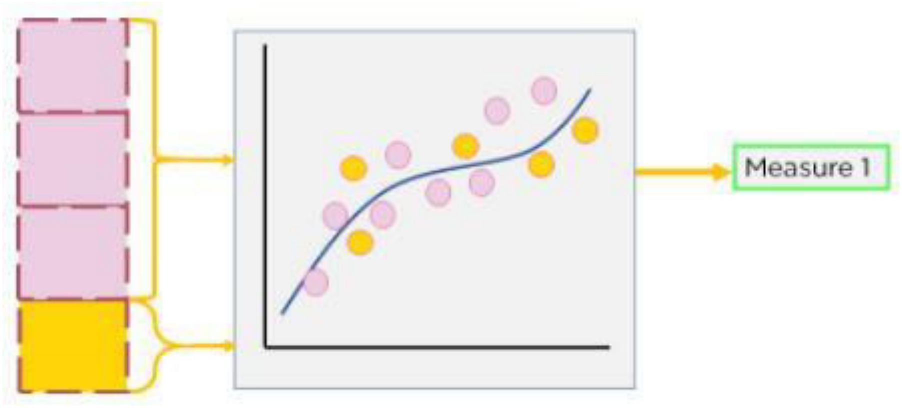


Figure 7: Step 1 of cross-validation partitioning of the dataset

Step 2: Split the data into new train and test sets and re-evaluate the model's performance

After evaluating one portion of the dataset, we choose a different portion to test and train on. The output measure obtained from this new training and testing dataset is again noted.

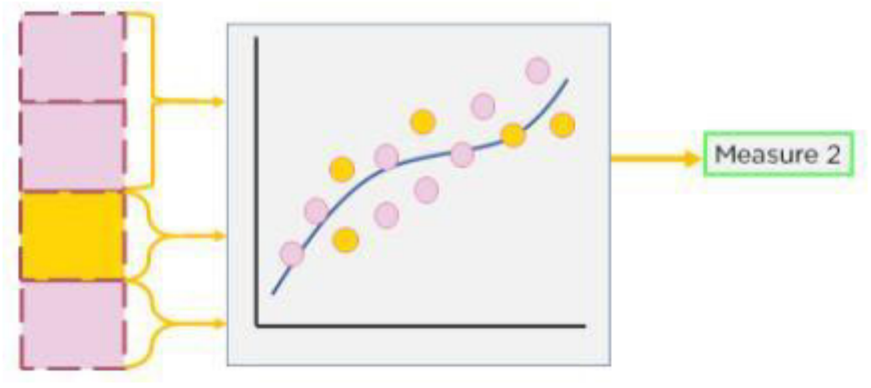


Figure 8: Step 2 of cross-validation revaluation on new portions

This step is repeated multiple times until the model has been trained and evaluated on the entire dataset.

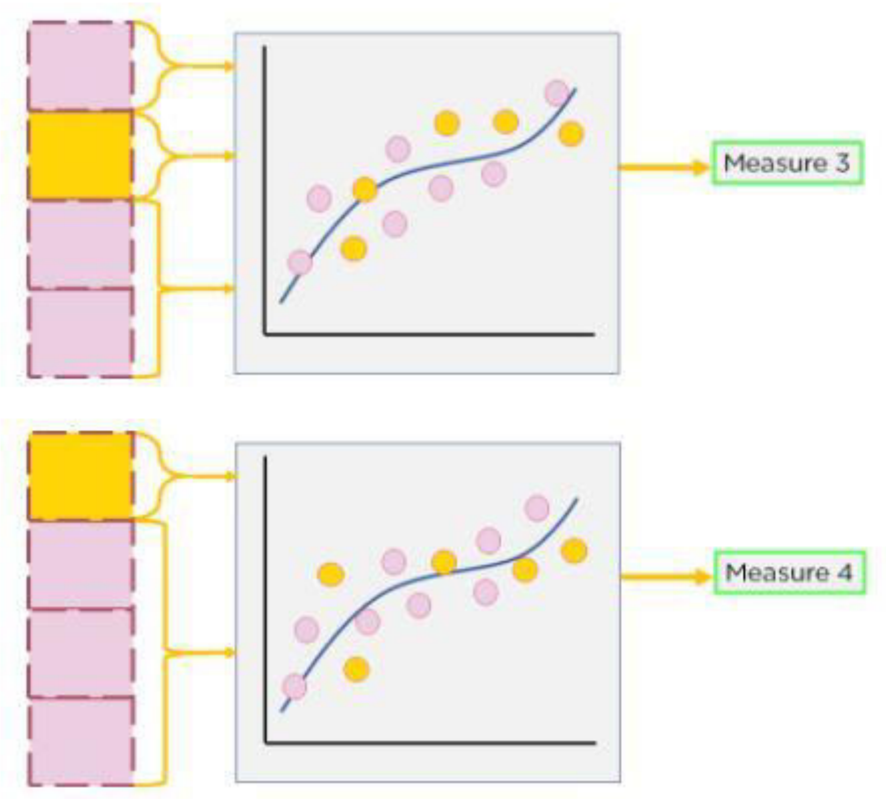


Figure 9: Repeating Step 2 of cross-validation

Step 3: To get the actual performance metric the average of all measures is taken

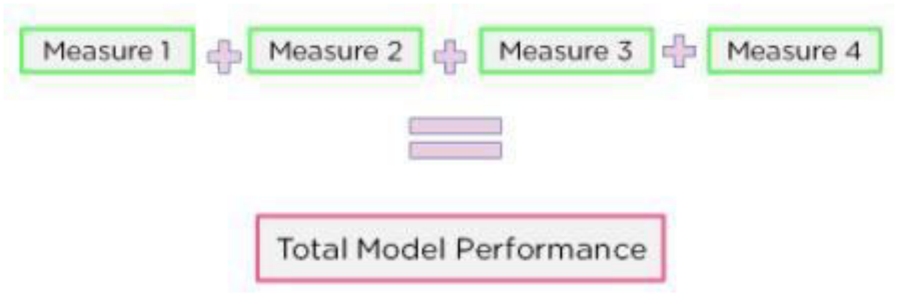


Figure 10: Step 3 of cross-validation getting model performance

Cross validation is a model evaluation method that is better than residuals. The problem with residual evaluations is that they do not give an indication of how well the learner will do when it is asked to make new predictions for data it has not already seen. One way to overcome this problem is to not use the entire data set when training a learner. Some of the data is removed before training begins. Then when training is done, the data that was removed can be used to test the performance of the learned model on ``new'' data. This is the basic idea for a whole class of model evaluation methods called *cross validation*.

Cross-validation is a technique for validating the model efficiency by training it on the subset of input data and testing on previously unseen subset of the input data. We can also say that it is a technique to check how a statistical model generalizes to an independent dataset.

In machine learning, there is always the need to test the stability of the model. It means based only on the training dataset; we can't fit our model on the training dataset. For this purpose, we reserve a particular sample of the dataset, which was not part of the training dataset. After that, we test our model on that sample before deployment, and this complete process comes under cross-validation. This is something different from the general train-test split.

Hence the basic steps of cross-validations are:

- Reserve a subset of the dataset as a validation set.
- Provide training to the model using the training dataset.
- Now, evaluate model performance using the validation set. If the model performs well with the validation set, perform the further step, else check for the issues.

Why cross-validation?

CV provides the ability to estimate model performance on unseen data not used while training.

Data scientists rely on several reasons for using cross-validation during their building process of Machine Learning (ML) models. For instance, tuning the model hyperparameters, testing different properties of the overall datasets, and iterate the training process. Also, in cases where your training dataset is small, and the ability to split them into training, validation, and testing will significantly affect training accuracy. The following main points can summarize the reason we use a CV, but they overlap. Hence, the list is presented here in a simplified way:

(1) Testing on unseen data

One of the critical pillars of validating a learning model before putting them in production is making accurate predictions on unseen data. The unseen data is all types of data that a model has never learned before. Ideally, the testing data is supposed to flow directly to the model in many testing iterations. However, in reality, access to such data is limited or not yet available in a new environment.

The typical 80–20 rule of splitting data into training and testing can still be vulnerable to accidentally ending up in a perfect split that boosts the model accuracy while limiting it from performing the same in a real environment. Sometimes, the accuracy calculated this way is mostly a matter of luck! The 80–20 is not an actual rule per se, and you will find alternative ratios that range between 25~30% for testing and 70~75% for training.

(2) Tuning model hyperparameter

Finding the best combination of model parameters is a common step to tune an algorithm toward learning the dataset's hidden patterns. But, doing this step on a simple training-testing split is typically not recommended. The model performance is usually very sensitive to such parameters, and adjusting those based on a predefined dataset split should be avoided. It can cause the model to overfit and reduce its ability to generalize.

(3) The third split is not achievable

For parameter tuning and avoid model overfitting, you might find some recommendations around splitting the dataset into three partitions: training, testing, and validation. For instance, 70% of the data is used for training, 20% for validation, and the remaining 10% is used for testing. In cases where the actual dataset is small, we might need to invest in using the maximum amount of data to train the model. In other instances, splitting into such three partitions could create bias in the training process where some significant examples are kept in training or validation splits. Hence, CV can become handy to resolve this issue.

(4) Avoid instability of sampling

Sometimes the splits of training-testing data can be very tricky. The properties of the testing data are not similar to the properties of the training. Although randomness ensures that each sample can have the same chance to be selected in the testing set, the process of a single split can still bring instability when the experiment is repeated with a new division.

Types of Cross Validation

There are different types of cross validation methods, and they could be classified into two broad categories –

- Non-exhaustive
- Exhaustive Methods.

Non-exhaustive Methods

Non-exhaustive cross validation methods, as the name suggests do not compute all ways of splitting the original data.

Holdout method

This is a quite basic and simple approach in which we divide our entire dataset into two parts viz- training data and testing data. As the name, we train the model on training data and then evaluate on the testing set. Usually, the size of training data is set more than twice that of testing data, so the data is split in the ratio of 70:30 or 80:20.

In this approach, the data is first shuffled randomly before splitting. As the model is trained on a different combination of data points, the model can give different results every time we train it, and this can be a cause of instability. Also, we can never assure that the train set we picked is representative of the whole dataset.

Also when our dataset is not too large, there is a high possibility that the testing data may contain some important information that we lose as we do not train the model on the testing set.

The hold-out method is good to use when you have a very large dataset, you’re on a time crunch, or you are starting to build an initial model in your data science project.

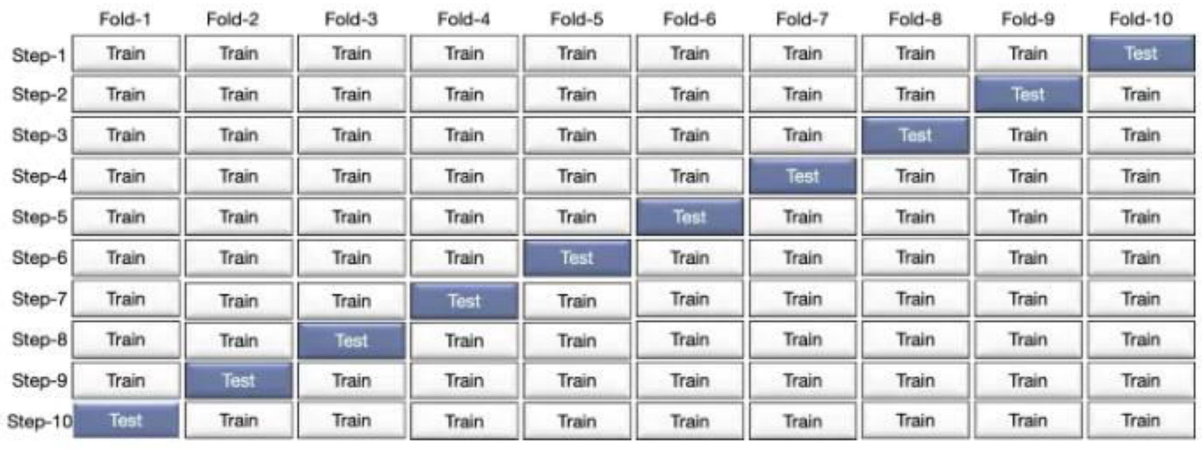
K fold cross validation

K-fold cross validation is one way to improve the holdout method. This method guarantees that the score of our model does not depend on the way we picked the train and test set. The data set is divided into k number of subsets and the holdout method is repeated k number of times. Let us go through this in steps:

1. Randomly split your entire dataset into k number of folds (subsets)
2. For each fold in your dataset, build your model on k – 1 folds of the dataset. Then, test the model to check the effectiveness for kth fold
3. Repeat this until each of the k-folds has served as the test set
4. The average of your k recorded accuracy is called the cross-validation accuracy and will serve as your performance metric for the model.

Because it ensures that every observation from the original dataset has the chance of appearing in training and test set, this method generally results in a less biased model compare to other methods. It is one of the best approaches if we have limited input data.

The disadvantage of this method is that the training algorithm has to be rerun from scratch k times, which means it takes k times as much computation to make an evaluation.

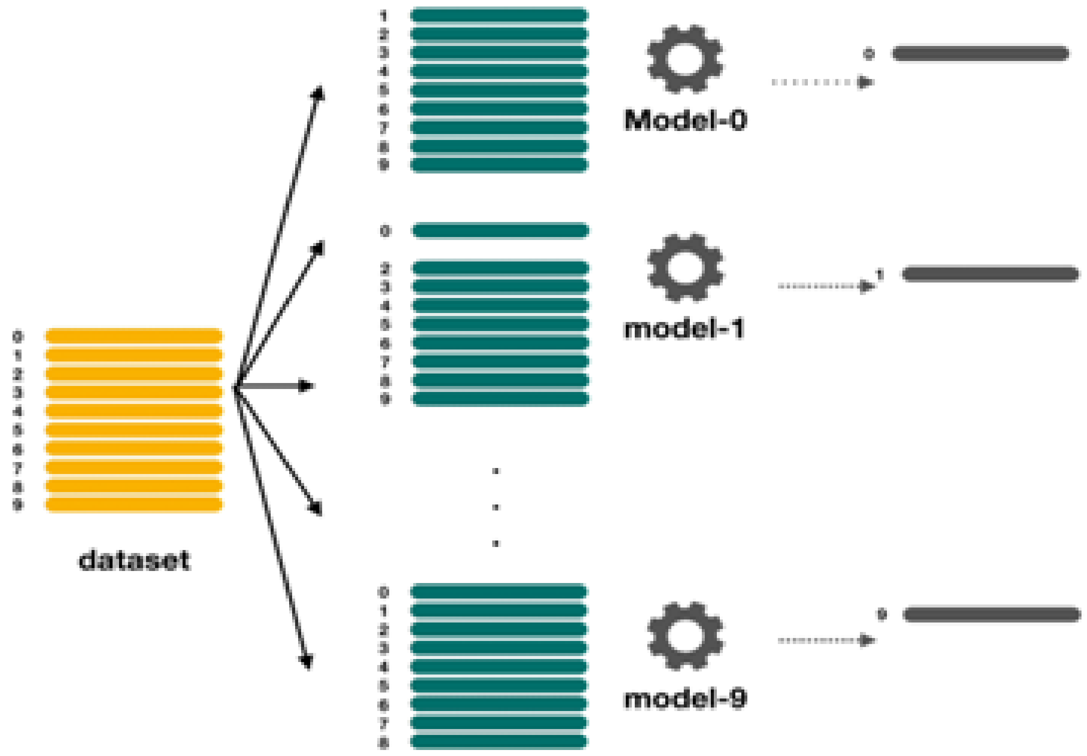


	Fold-1	Fold-2	Fold-3	Fold-4	Fold-5	Fold-6	Fold-7	Fold-8	Fold-9	Fold-10
Step-1	Train	Train	Train	Train	Train	Train	Train	Train	Train	Test
Step-2	Train	Train	Train	Train	Train	Train	Train	Train	Test	Train
Step-3	Train	Train	Train	Train	Train	Train	Train	Test	Train	Train
Step-4	Train	Train	Train	Train	Train	Train	Test	Train	Train	Train
Step-5	Train	Train	Train	Train	Train	Test	Train	Train	Train	Train
Step-6	Train	Train	Train	Train	Test	Train	Train	Train	Train	Train
Step-7	Train	Train	Train	Test	Train	Train	Train	Train	Train	Train
Step-8	Train	Train	Test	Train	Train	Train	Train	Train	Train	Train
Step-9	Train	Test	Train	Train	Train	Train	Train	Train	Train	Train
Step-10	Test	Train	Train	Train	Train	Train	Train	Train	Train	Train

Leave one out cross-validation

This method is similar to the leave-p-out cross-validation, but instead of p, we need to take 1 dataset out of training. It means, in this approach, for each learning set, only one datapoint is reserved, and the remaining dataset is used to train the model. This process repeats for each datapoint. Hence for n samples, we get n different training set and n test set. It has the following features:

- In this approach, the bias is minimum as all the data points are used.
- The process is executed for n times; hence execution time is high.
- This approach leads to high variation in testing the effectiveness of the model as we iteratively check against one data point.



Limitations of Cross-Validation

There are some limitations of the cross-validation technique, which are given below:

For the ideal conditions, it provides the optimum output. But for the inconsistent data, it may produce a drastic result. So, it is one of the big disadvantages of cross-validation, as there is no certainty of the type of data in machine learning.

In predictive modeling, the data evolves over a period, due to which, it may face the differences between the training set and validation sets. Such as if we create a model for the prediction of stock market values, and the data is trained on the previous 5 years stock values, but the realistic future values for the next 5 years may drastically different, so it is difficult to expect the correct output for such situations.

Applications of Cross-Validation

This technique can be used to compare the performance of different predictive modeling methods.

It has great scope in the medical research field.

It can also be used for the meta-analysis, as it is already being used by the data scientists in the field of medical statistics.