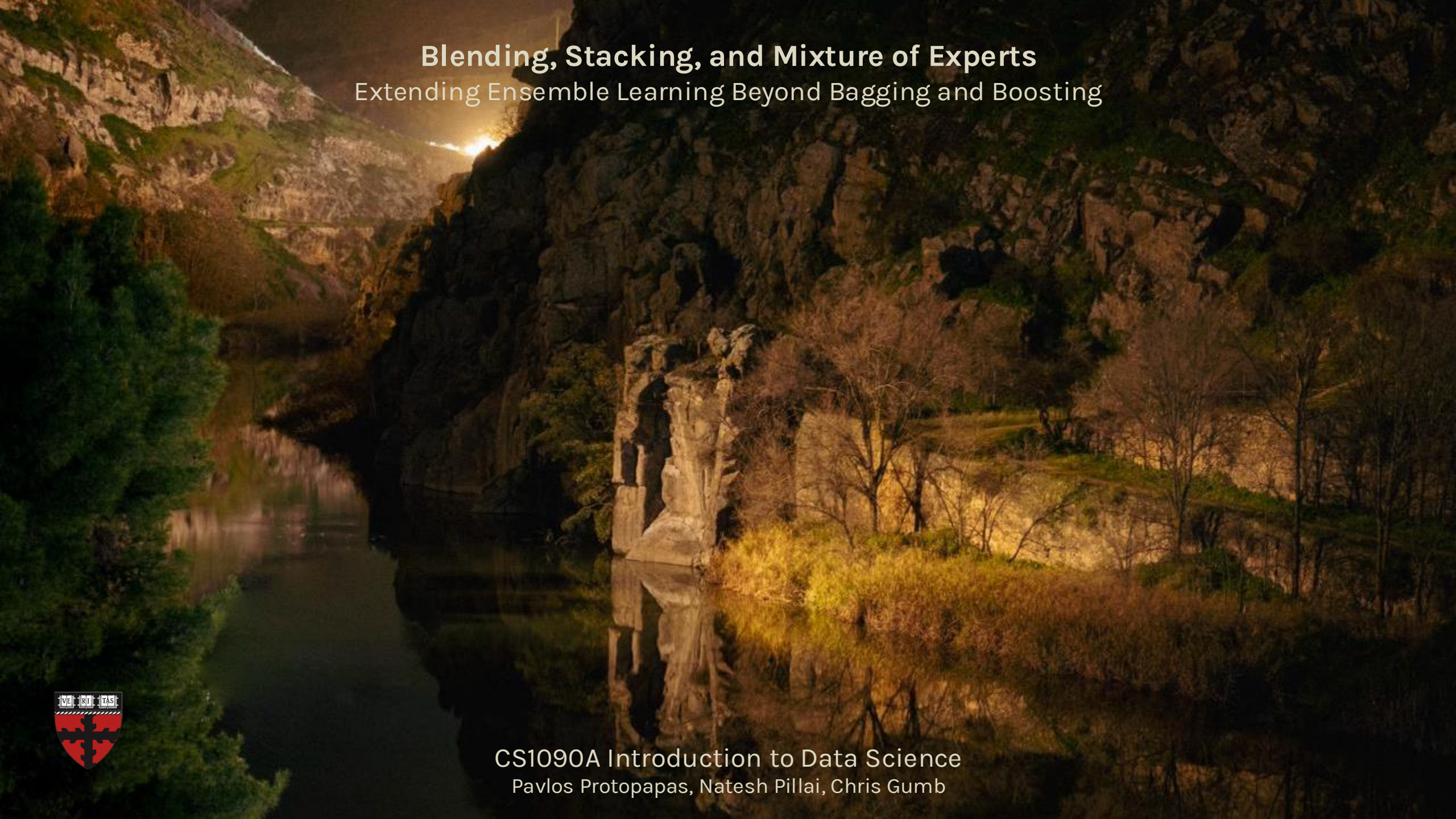


Blending, Stacking, and Mixture of Experts

Extending Ensemble Learning Beyond Bagging and Boosting

CS1090A Introduction to Data Science

Pavlos Protopapas, Natesh Pillai, Chris Gumb

Outline

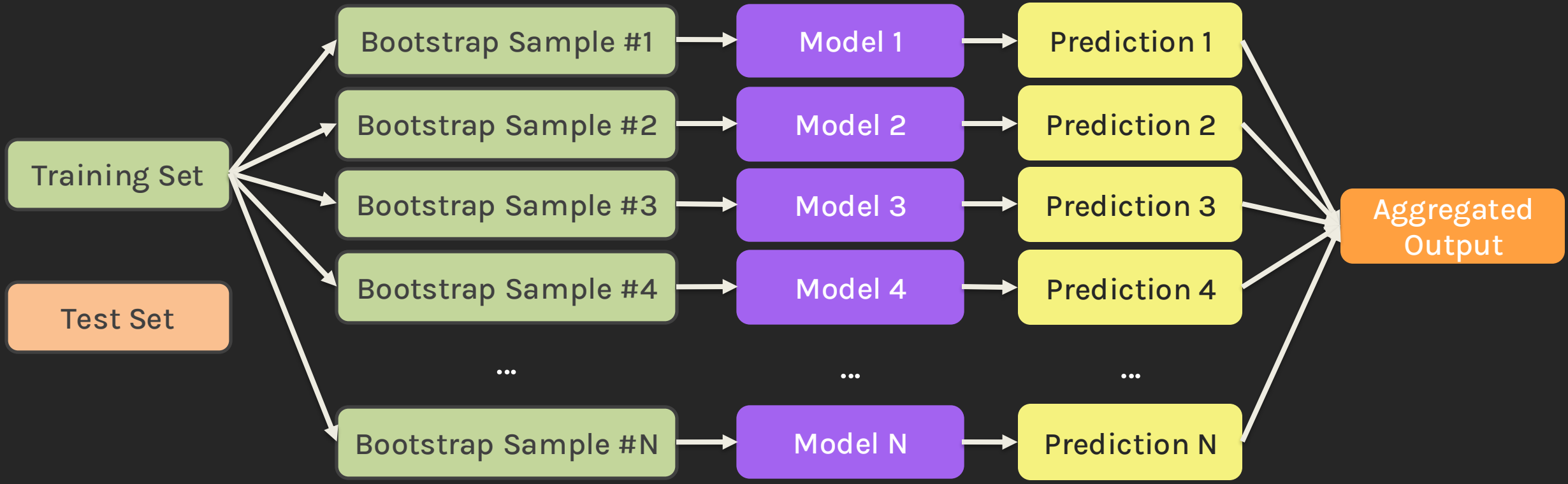
- Ensemble Methods
- Blending
- Stacking
- Mixture of Experts

Outline

- **Ensemble Methods**
- Blending
- Stacking
- Mixture of Experts

Recap: Bagging

Bagging (Bootstrap aggregating) trains models on each bootstrapped sample of the dataset and combines their predictions as the final output.



Recap: Bagging

Bagging (Bootstrap aggregating) trains models on each bootstrapped sample of the dataset and combines their predictions as the final output.

The base models are usually homogeneous 'strong' learners, i.e.,

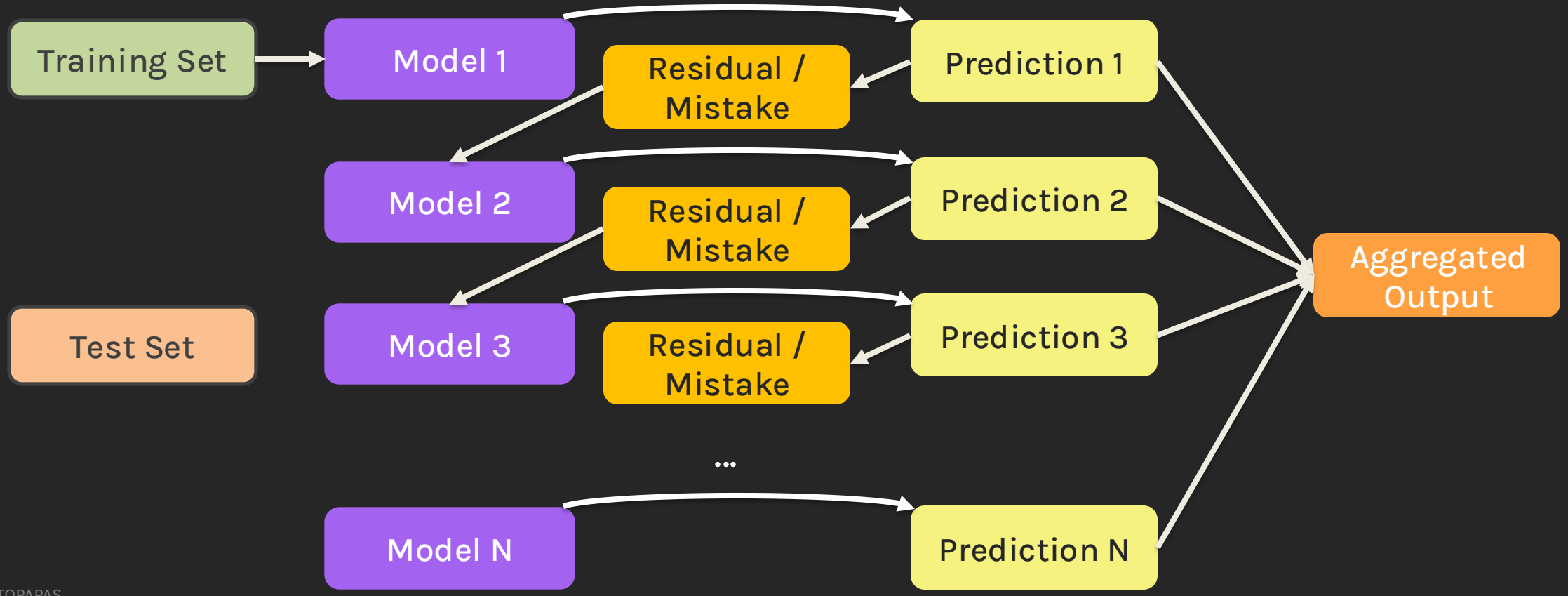
- They are the same type of models defined by same hyperparameters.
- They have low bias (and high variance) and tend to overfit.

Base models are trained in parallel, and their predictions are aggregated through:

- averaging for regression tasks
- majority voting for classification tasks

Recap: Boosting

In Boosting, base models are fit sequentially on the emphasis of residuals/mistakes from the previous ensemble, and final output is aggregated with different weights on base models' predictions.



Recap: Ensemble Learning

In Boosting, base models are fit sequentially on the emphasis of residuals/mistakes from the previous ensemble, and final output is aggregated with different weights on base models' predictions.

The base models are usually homogeneous 'weak' learners, i.e.,

- They are the same type of models defined by same hyperparameters.
- They have high bias and tend to underfit.

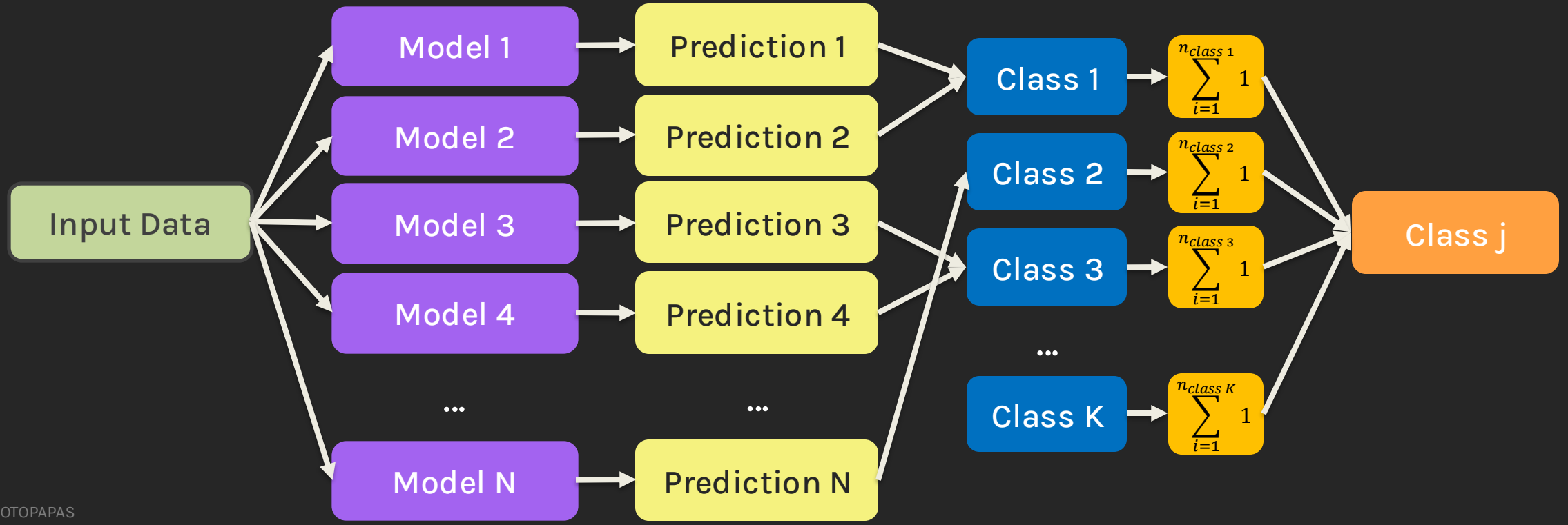
The predictions from base models are aggregated through:

- weighted averaging for regression tasks
- weighted voting for classification tasks

Recap: Ensemble Methods

So far, to aggregate the predictions of base models, we use voting and averaging, including

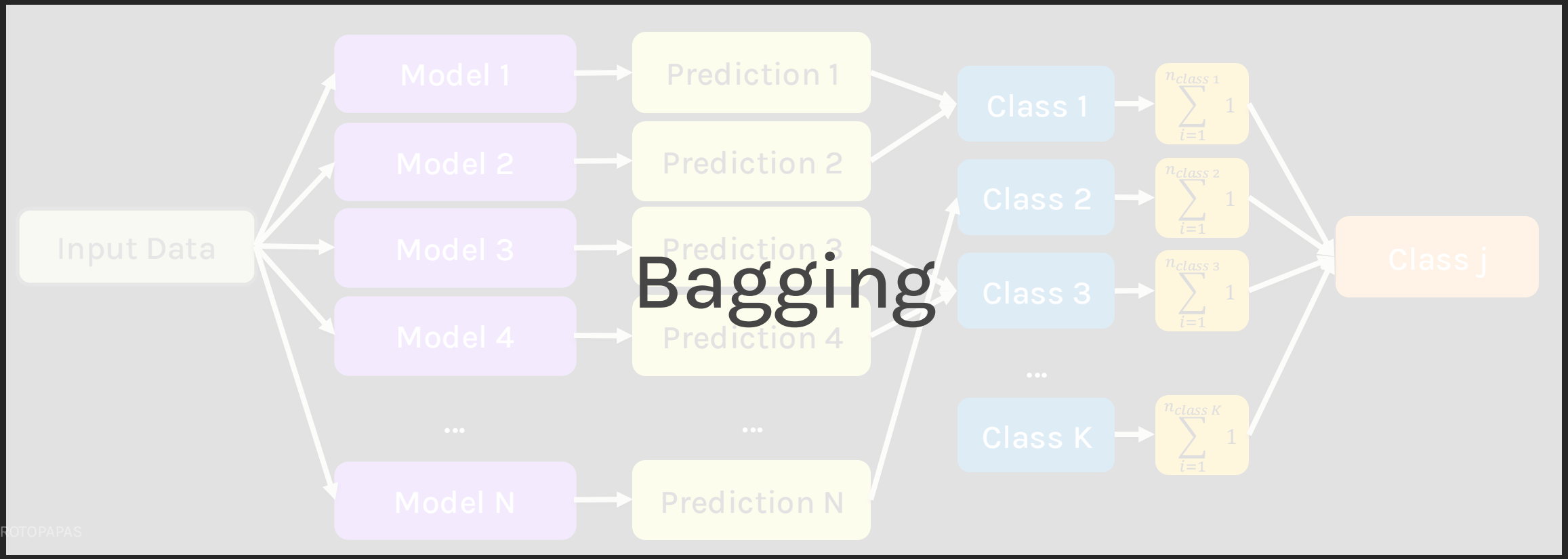
- Majority Voting (classification)



Recap: Ensemble Methods

So far, to aggregate the predictions of base models, we use voting and averaging, including

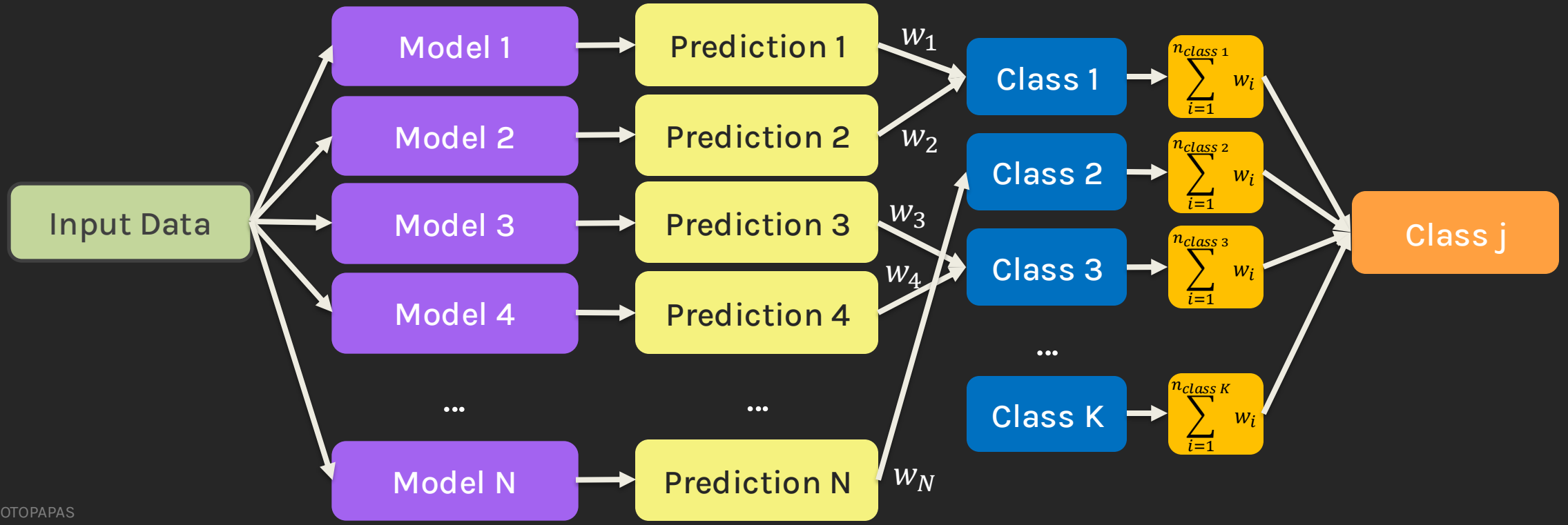
- Majority Voting (classification)



Recap: Ensemble Methods

So far, to aggregate the predictions of base models, we use voting and averaging, including

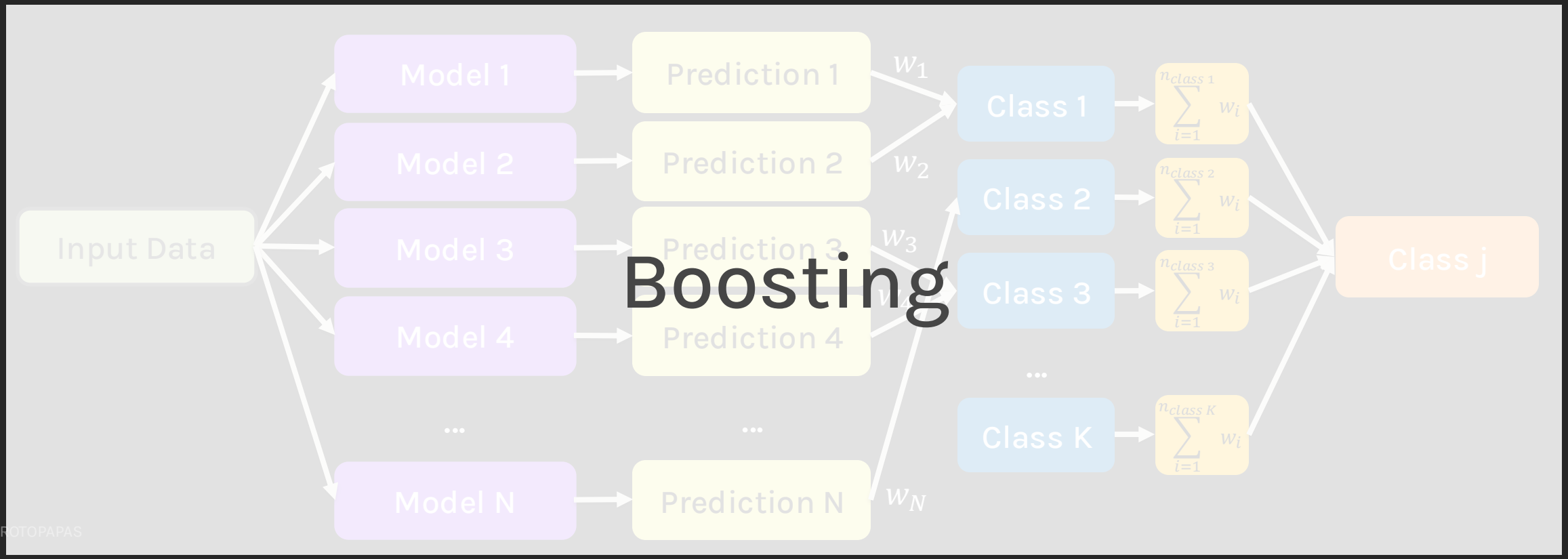
- Weighted Voting (classification)



Recap: Ensemble Methods

So far, to aggregate the predictions of base models, we use voting and averaging, including

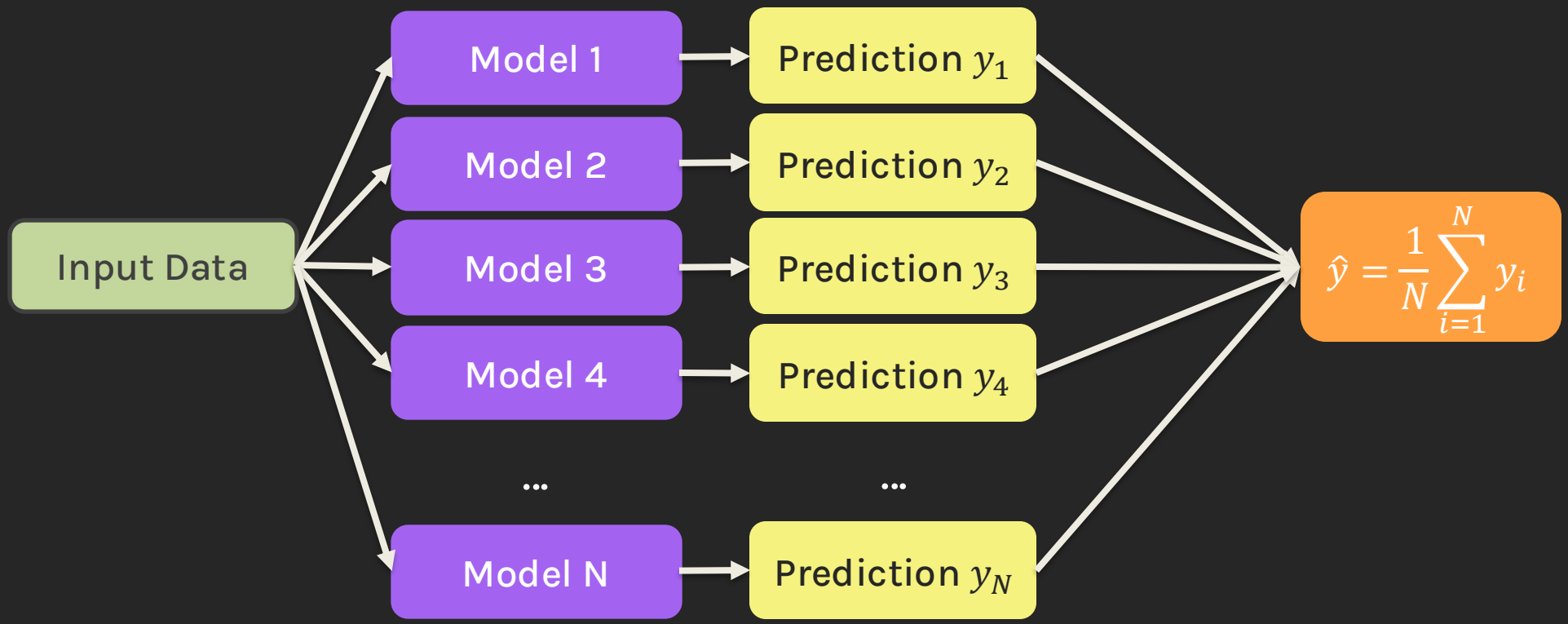
- Weighted Voting (classification)



Recap: Ensemble Methods

So far, to aggregate the predictions of base models, we use voting and averaging, including

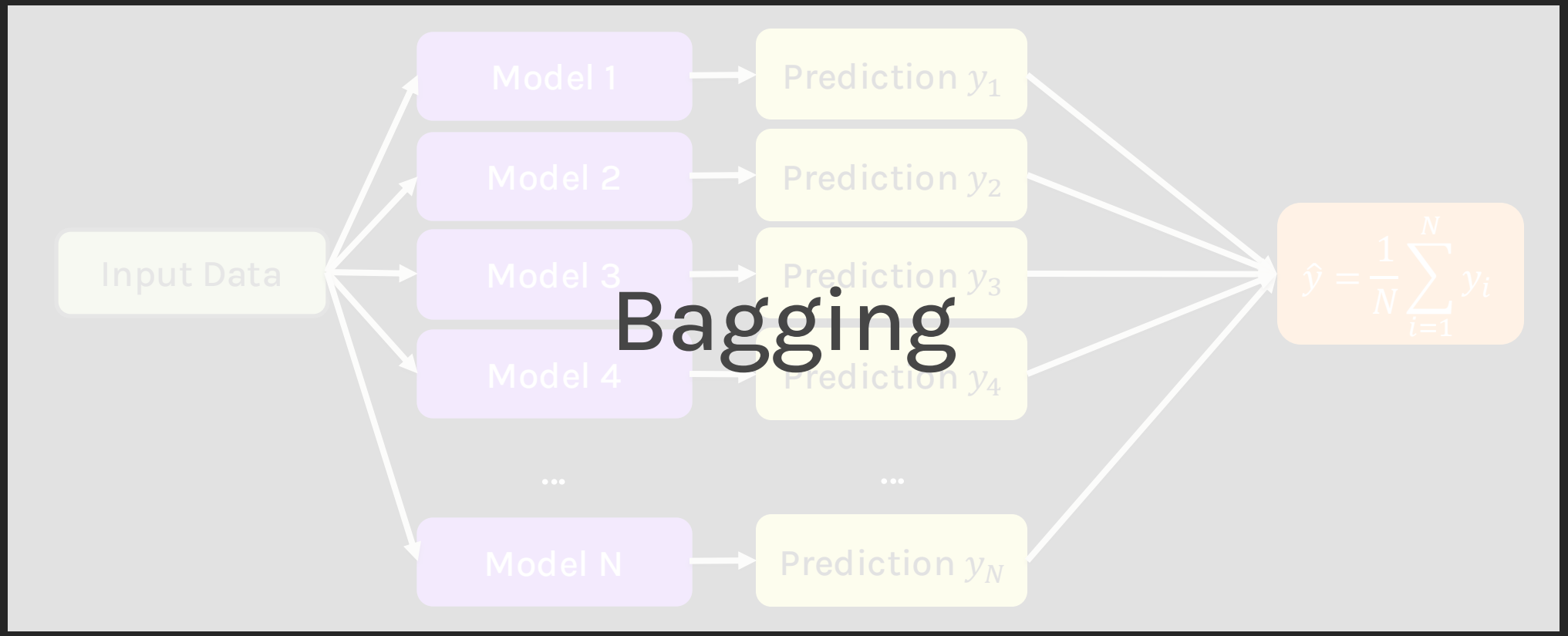
- Simple Averaging (regression)



Recap: Ensemble Methods

So far, to aggregate the predictions of base models, we use voting and averaging, including

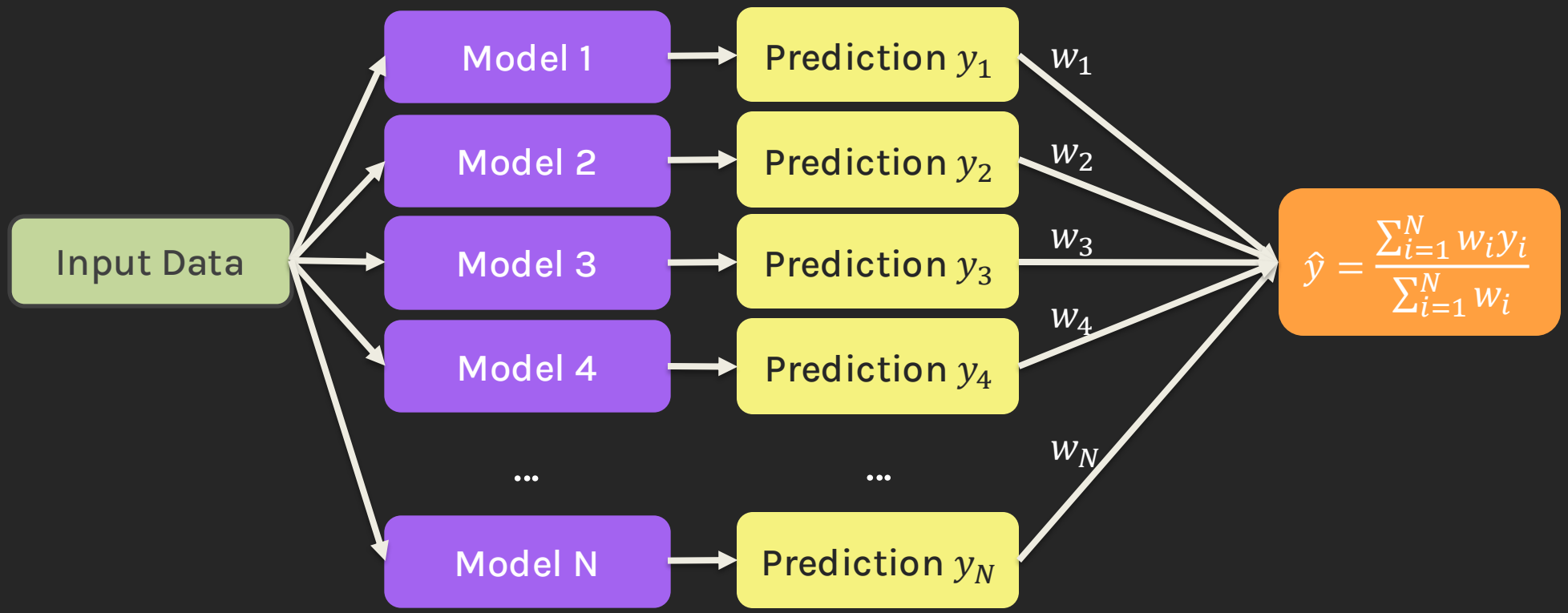
- Simple Averaging (regression)



Recap: Ensemble Methods

So far, to aggregate the predictions of base models, we use voting and averaging, including

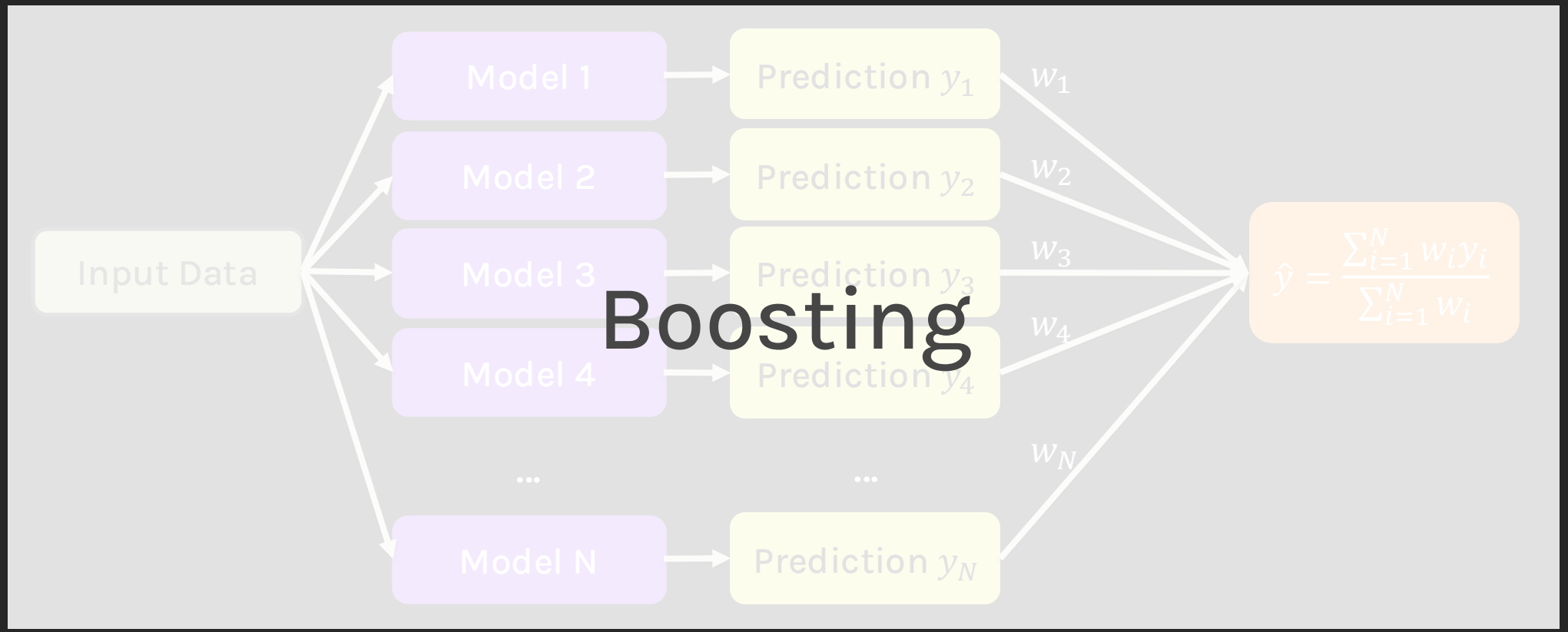
- Weighted Averaging (regression)



Recap: Ensemble Methods

So far, to aggregate the predictions of base models, we use voting and averaging, including

- Weighted Averaging (regression)



Can we aggregate outputs in a different way?

YES!

Can we use different models as base learners?

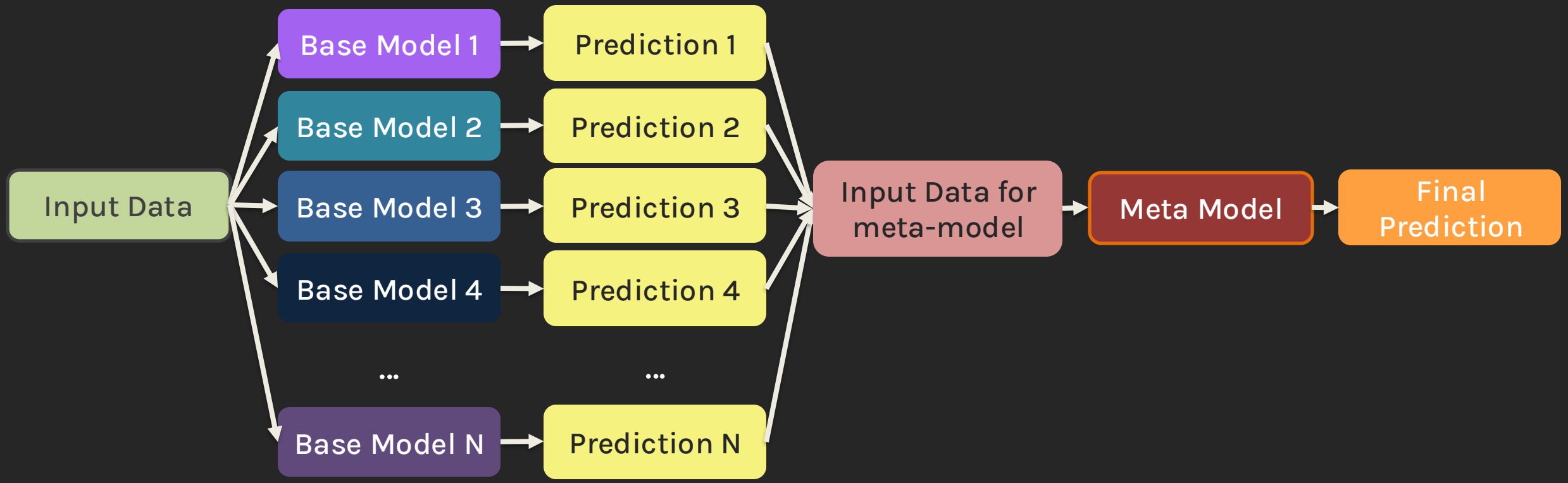
YES!

Outline

- Ensemble Methods
- **Blending**
- Stacking
- Mixture of Experts

Blending

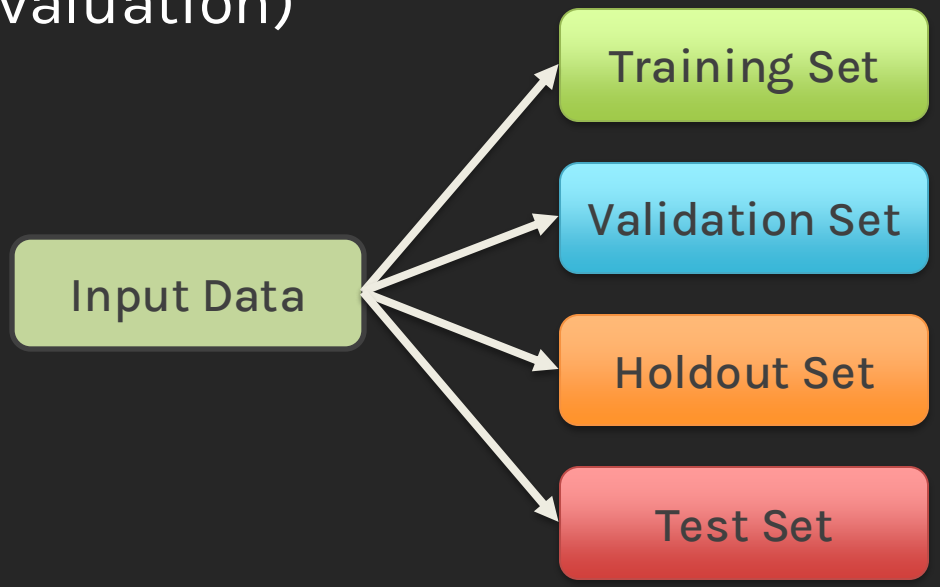
Blending trains heterogeneous learners (i.e., different models) on the dataset in parallel, and it further trains a meta-model on the base models' outputs for making predictions.



Blending

Step 1: Split the dataset into

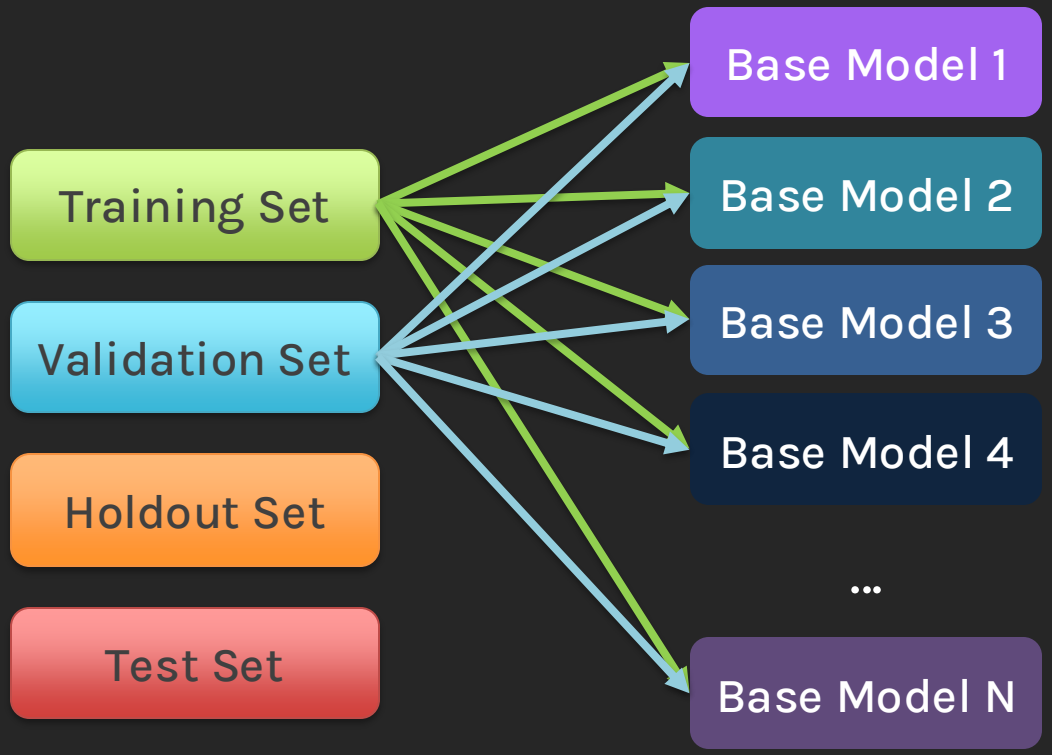
- Training set (to train base models)
- Validation set (to validate base models and generate predictions for training meta model)
- Holdout set (to validate meta model)
- Test set (for evaluation)



Blending

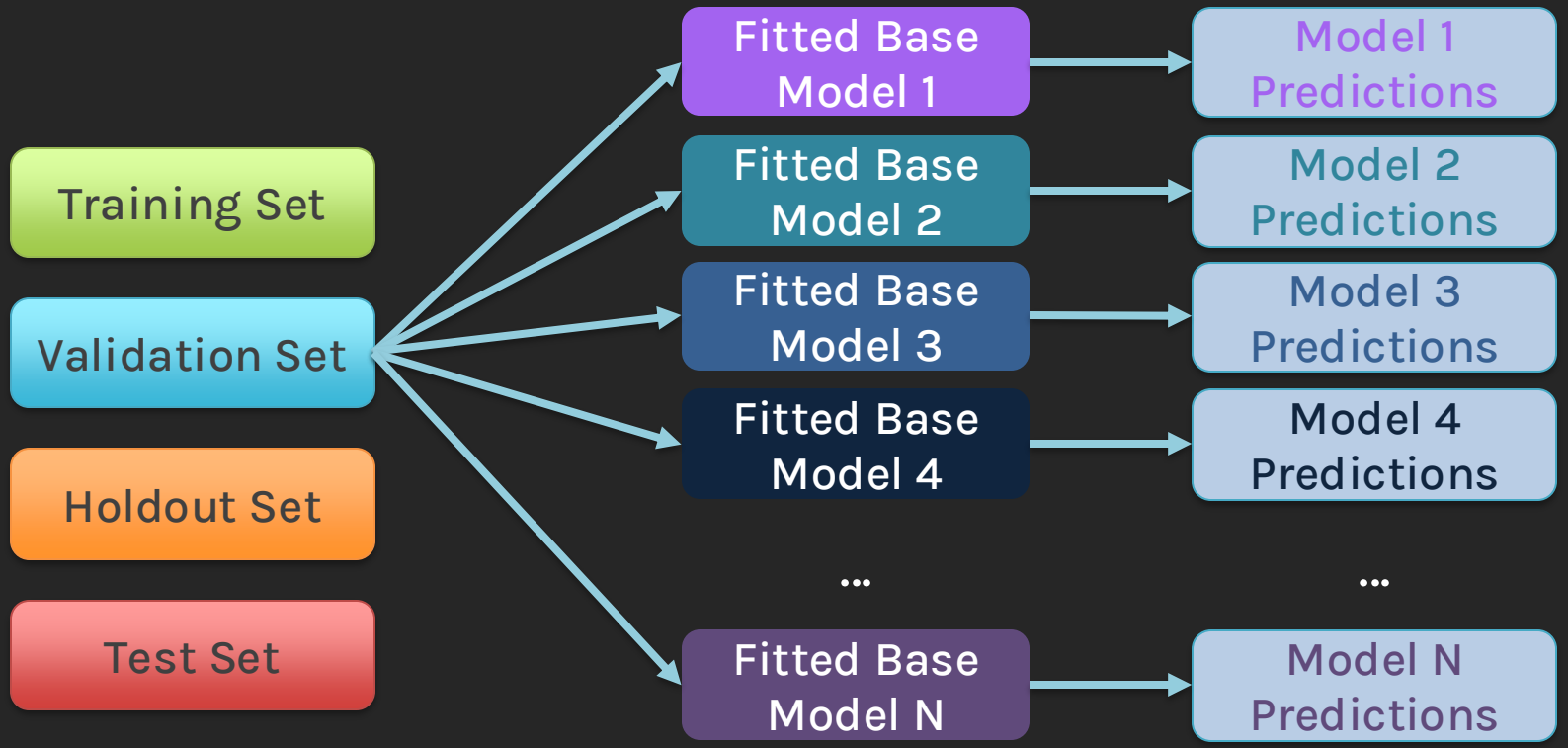
Step 2:

- Fit base models on training set and
- Validate with validation set



Blending

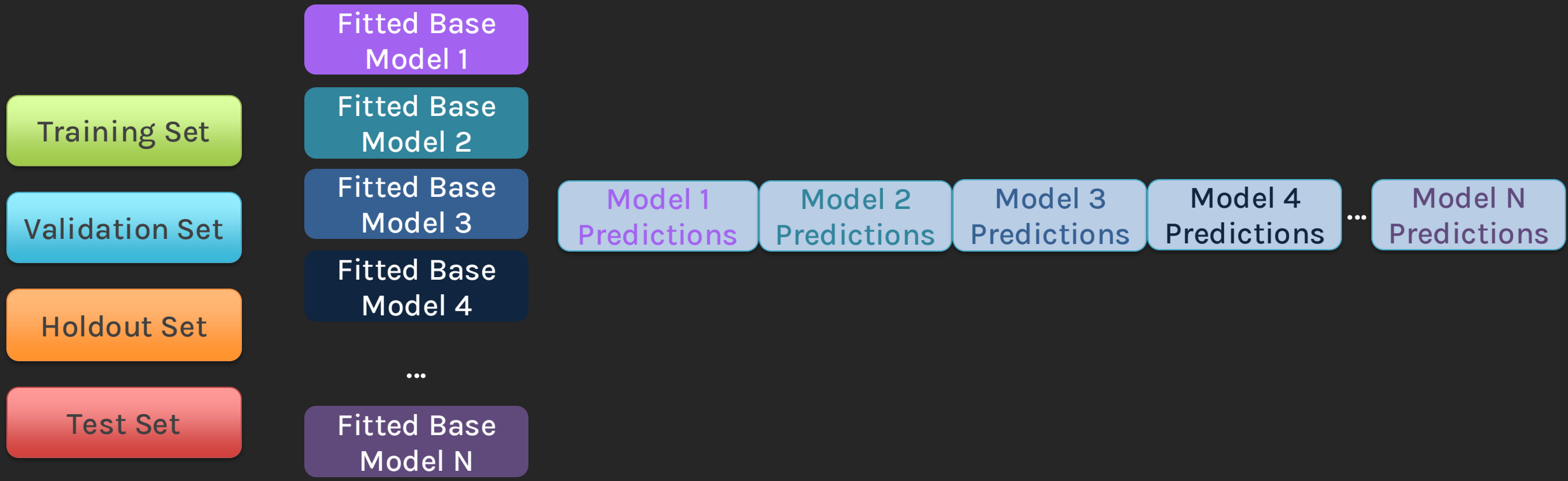
Step 3: Generate base model predictions on validation set and holdout set.



Blending

Step 3: Generate base model predictions on validation set and holdout set.

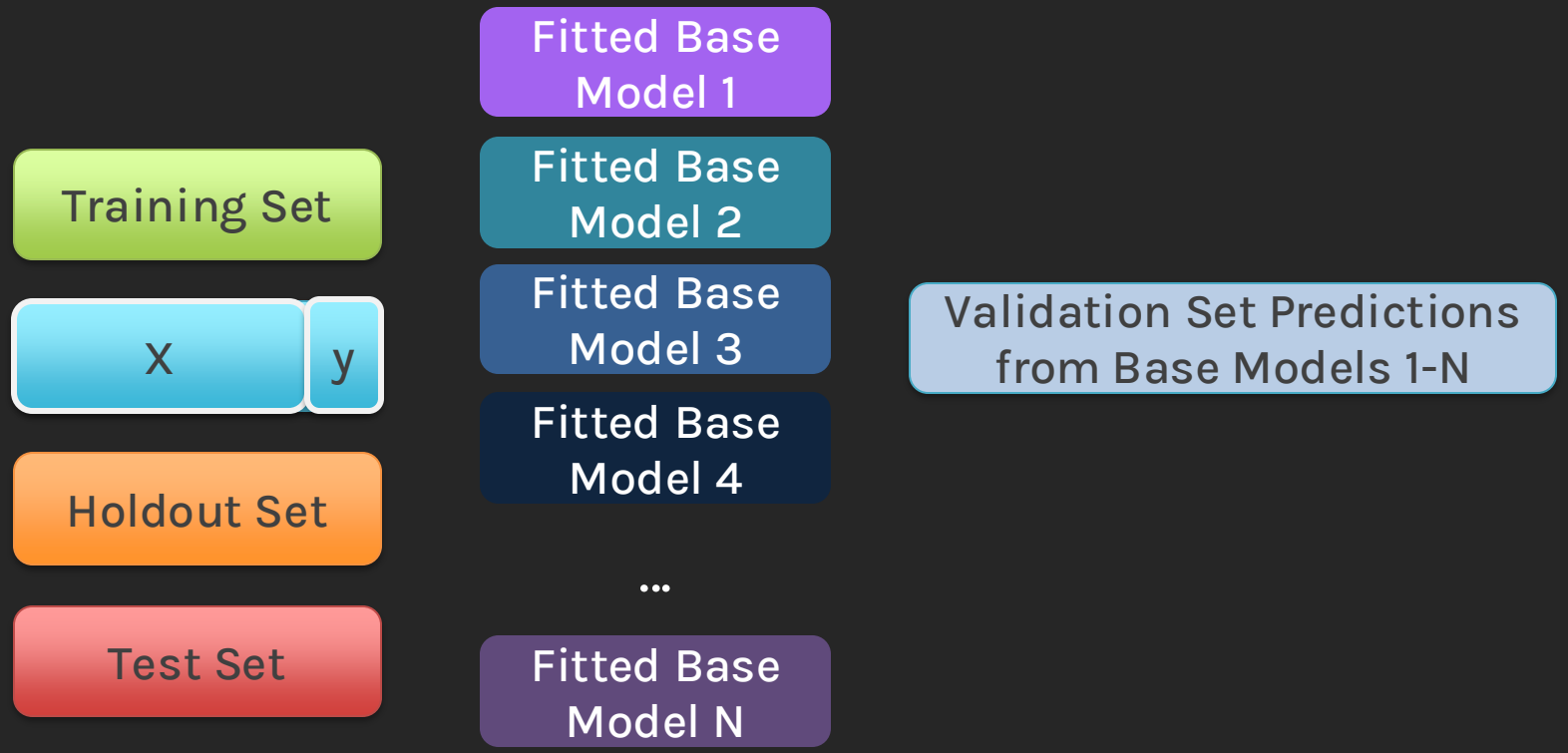
Step 4: Concatenate predictions and the original data into a new dataset.



Blending

Step 3: Generate base model predictions on validation set and holdout set.

Step 4: Concatenate predictions and the original data into a new dataset.

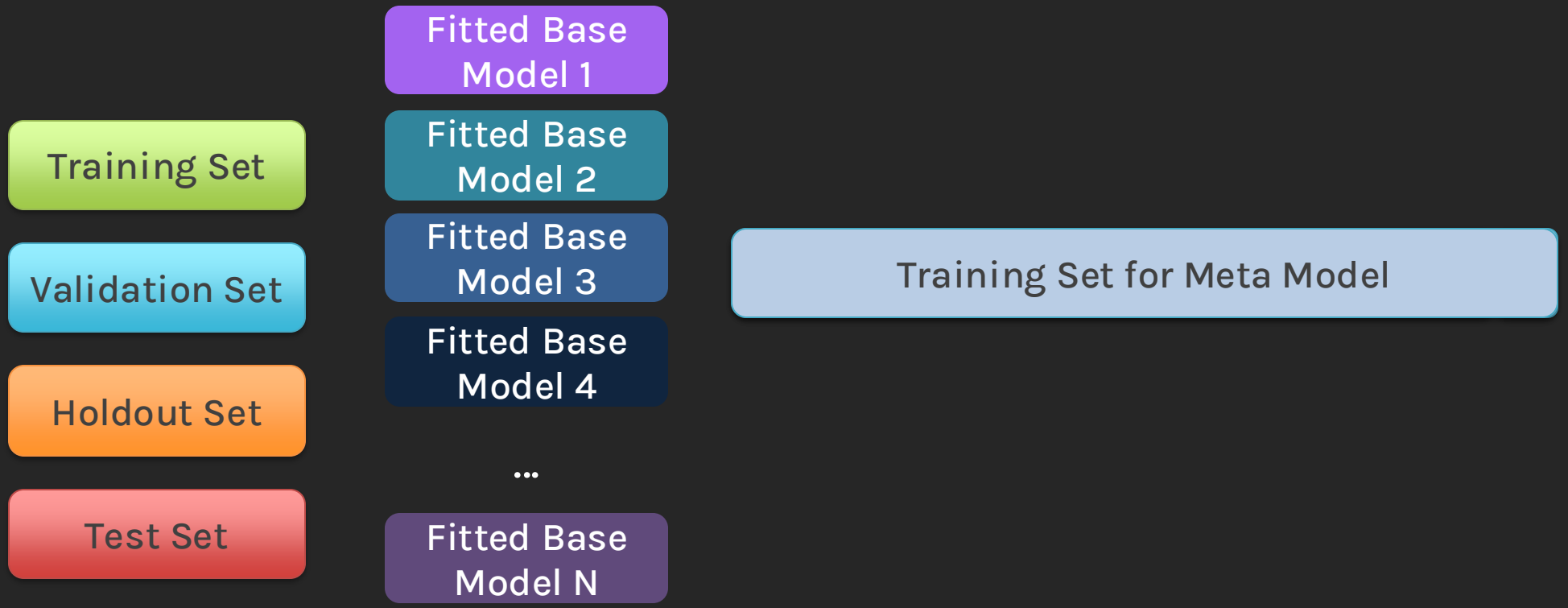


Blending

Step 3: Generate base model predictions on validation set and holdout set.

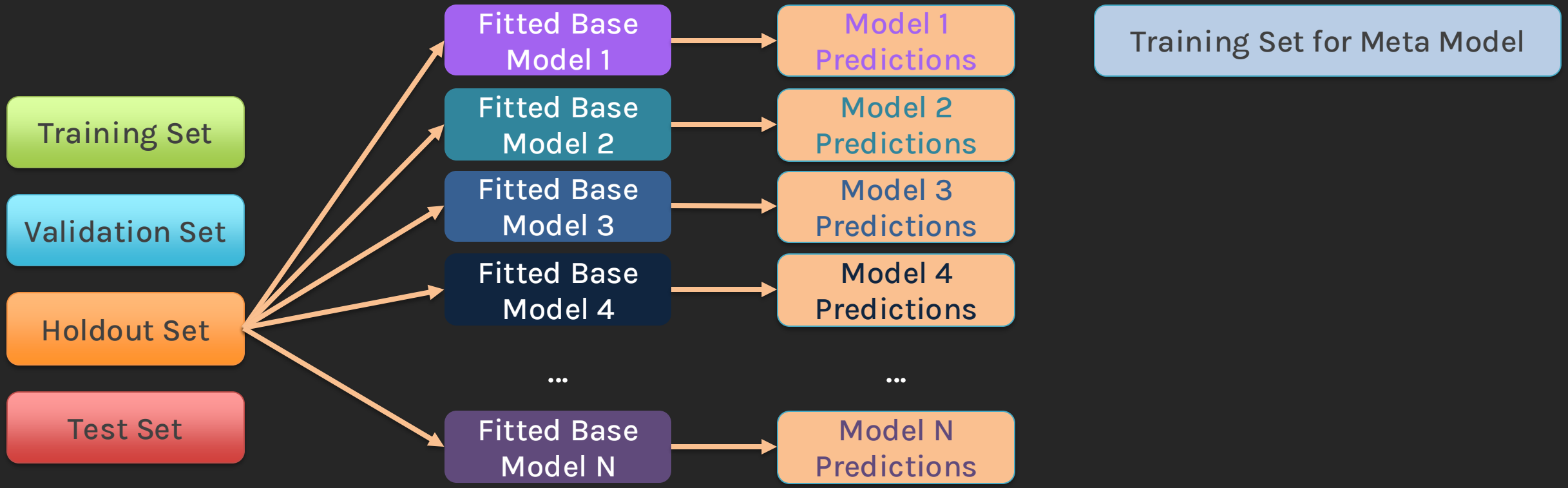
Step 4: Concatenate predictions and the original data into a new dataset.

Step 5: Combined features from validation set will be used for training meta model.



Blending

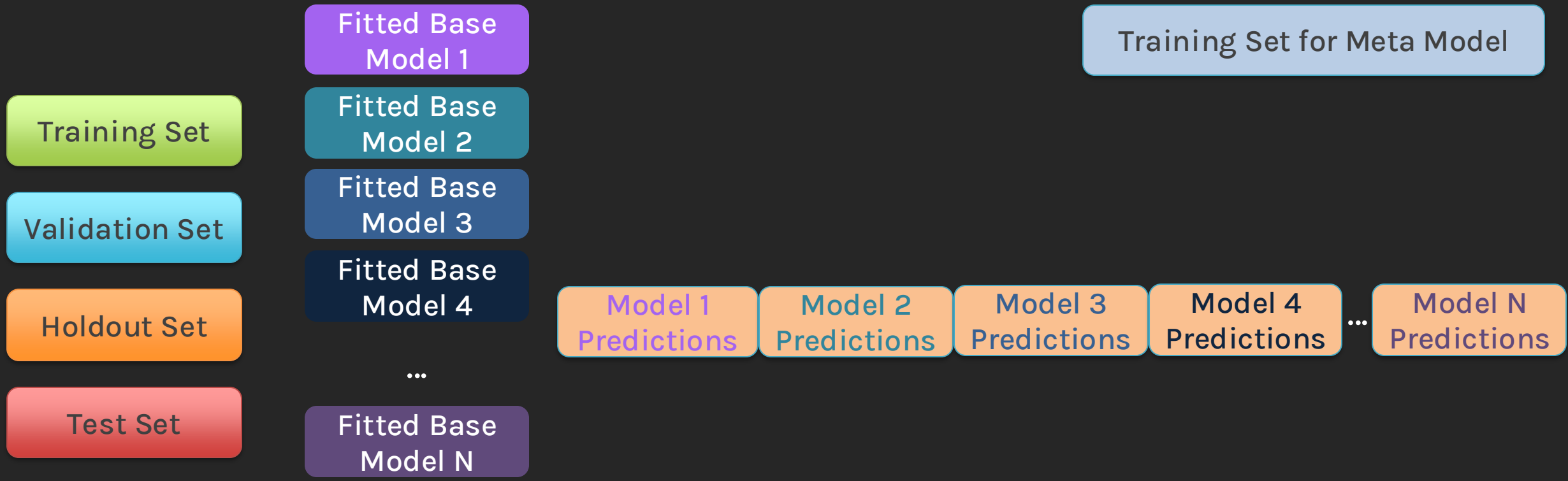
Step 3: Generate base model predictions on validation set and holdout set.



Blending

Step 3: Generate base model predictions on validation set and holdout set.

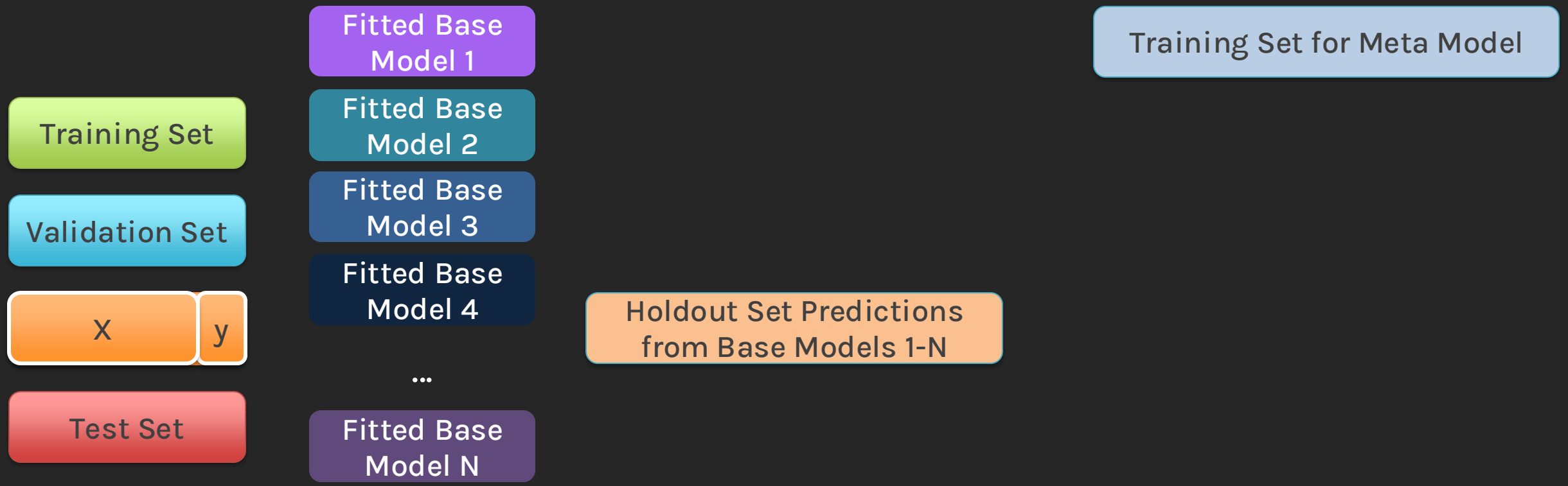
Step 4: Concatenate predictions into a new dataset.



Blending

Step 3: Generate base model predictions on validation set and holdout set.

Step 4: Concatenate predictions into a new dataset.

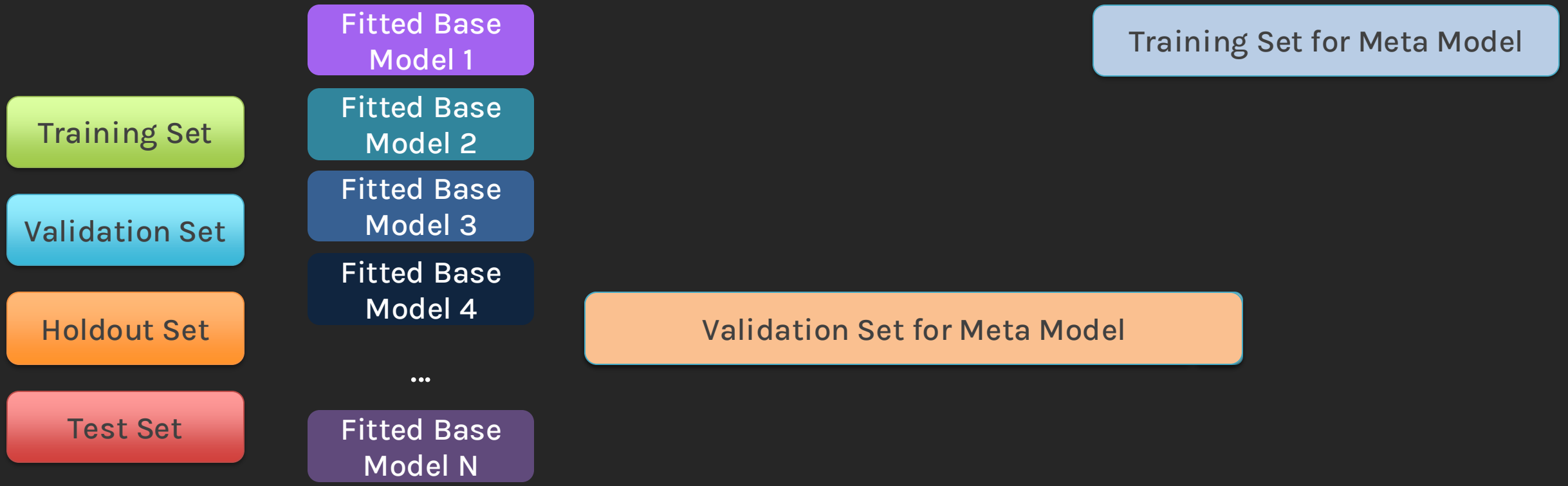


Blending

Step 3: Generate base model predictions on validation set and holdout set.

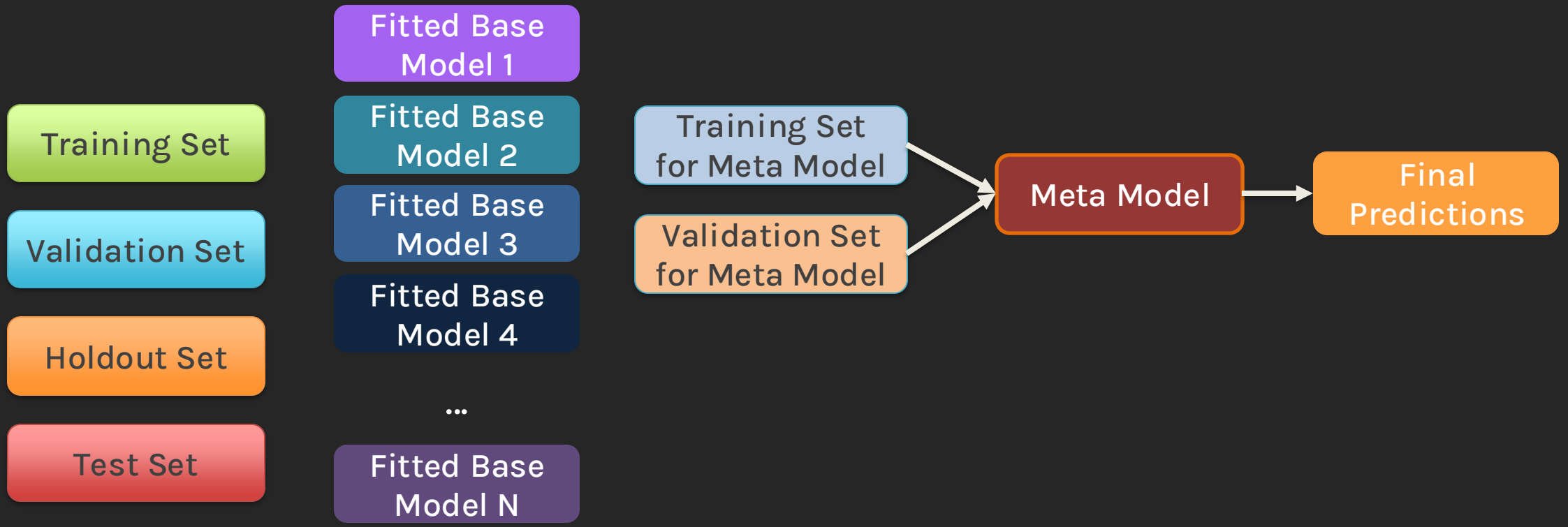
Step 4: Concatenate predictions into a new dataset.

Step 5: Combined features from holdout set will be used for validating meta model.



Blending

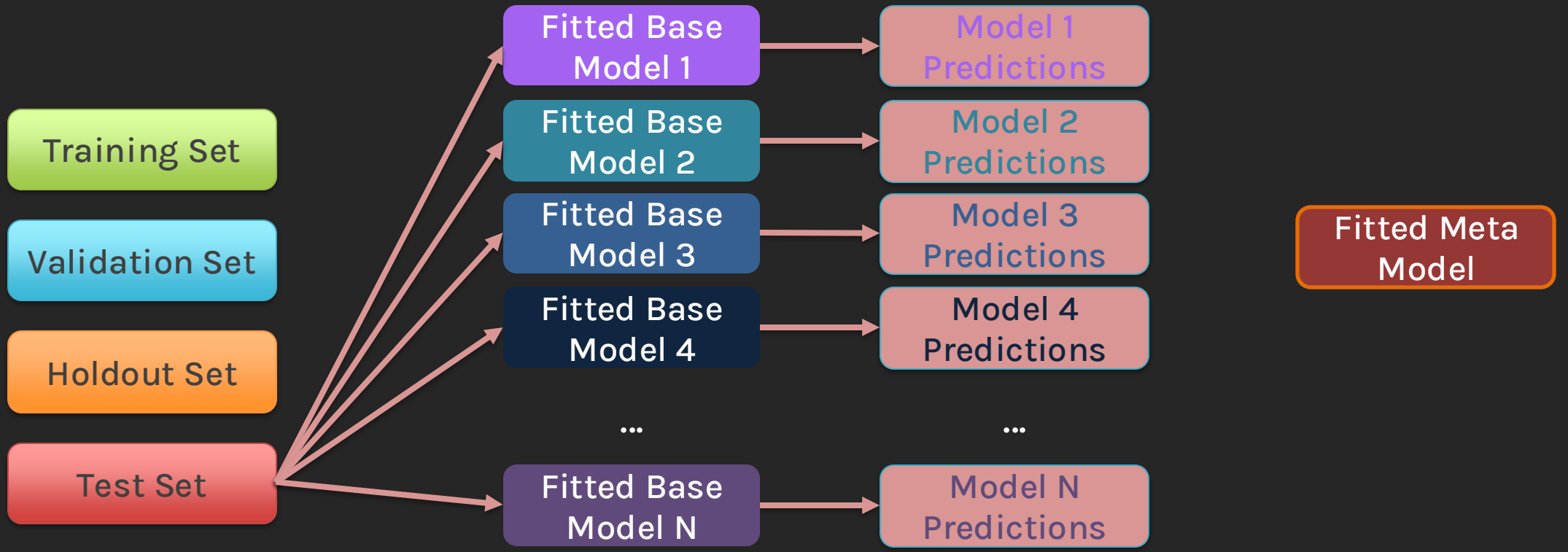
Step 6: Fit meta model with training and validation sets built with original data and base model predictions.



Blending

During inference (e.g., evaluation on test set):

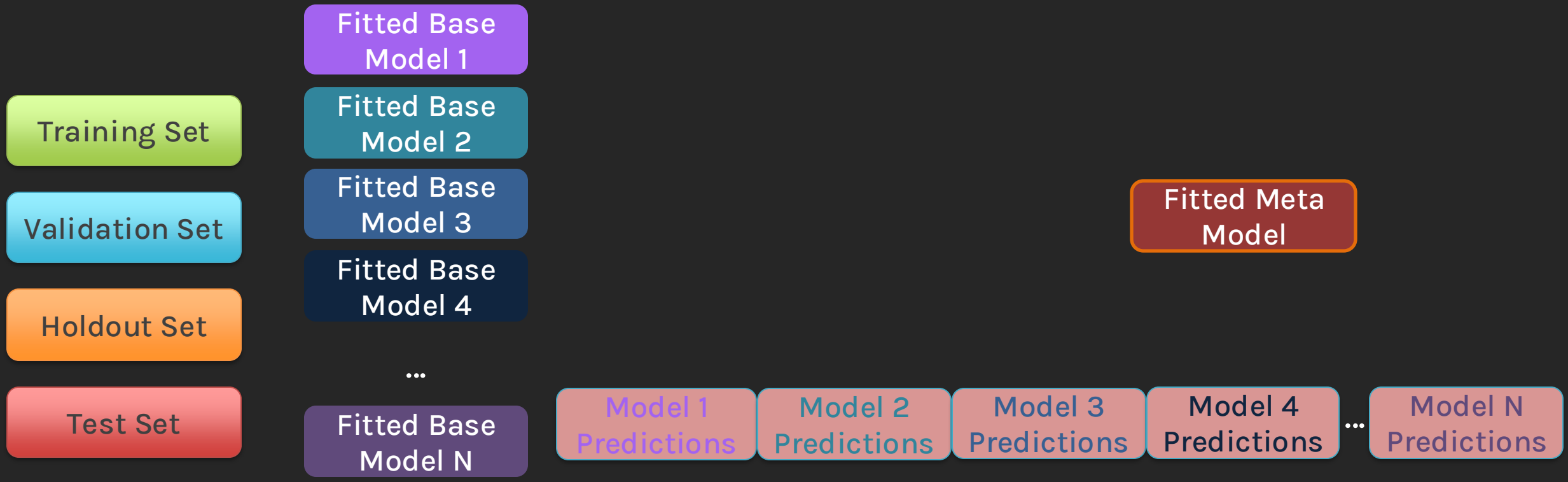
- Generate predictions from base models



Blending

During inference (e.g., evaluation on test set):

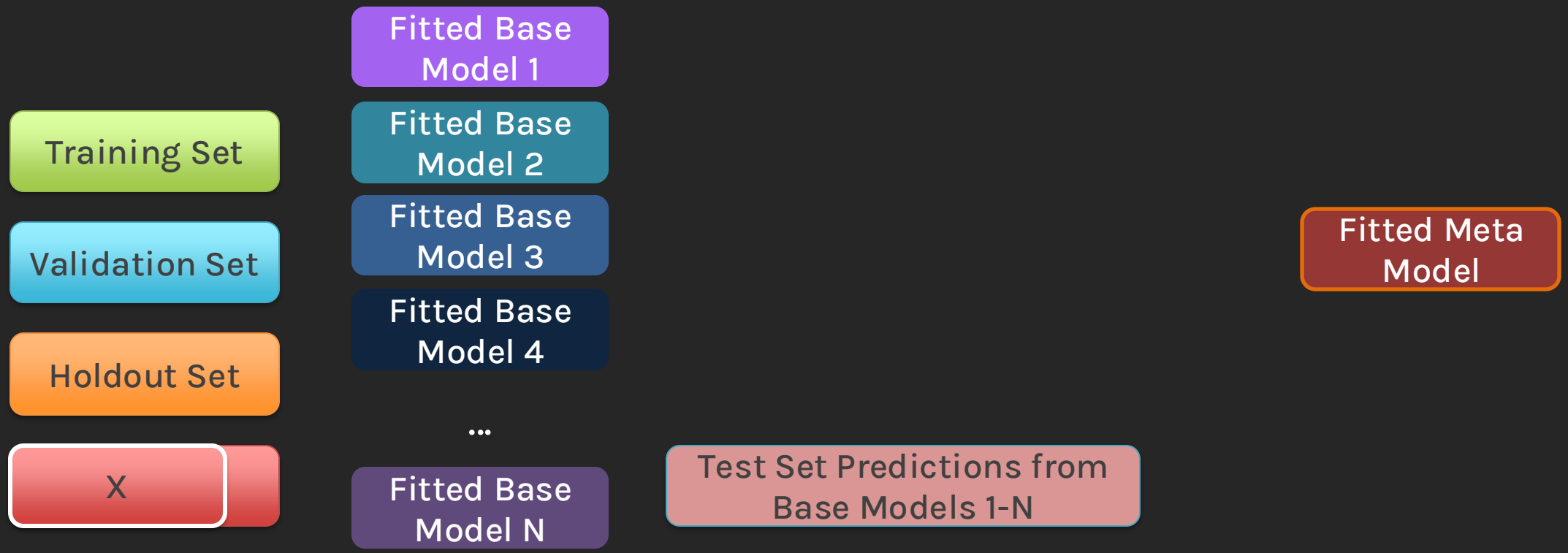
- Generate predictions from base models



Blending

During inference (e.g., evaluation on test set):

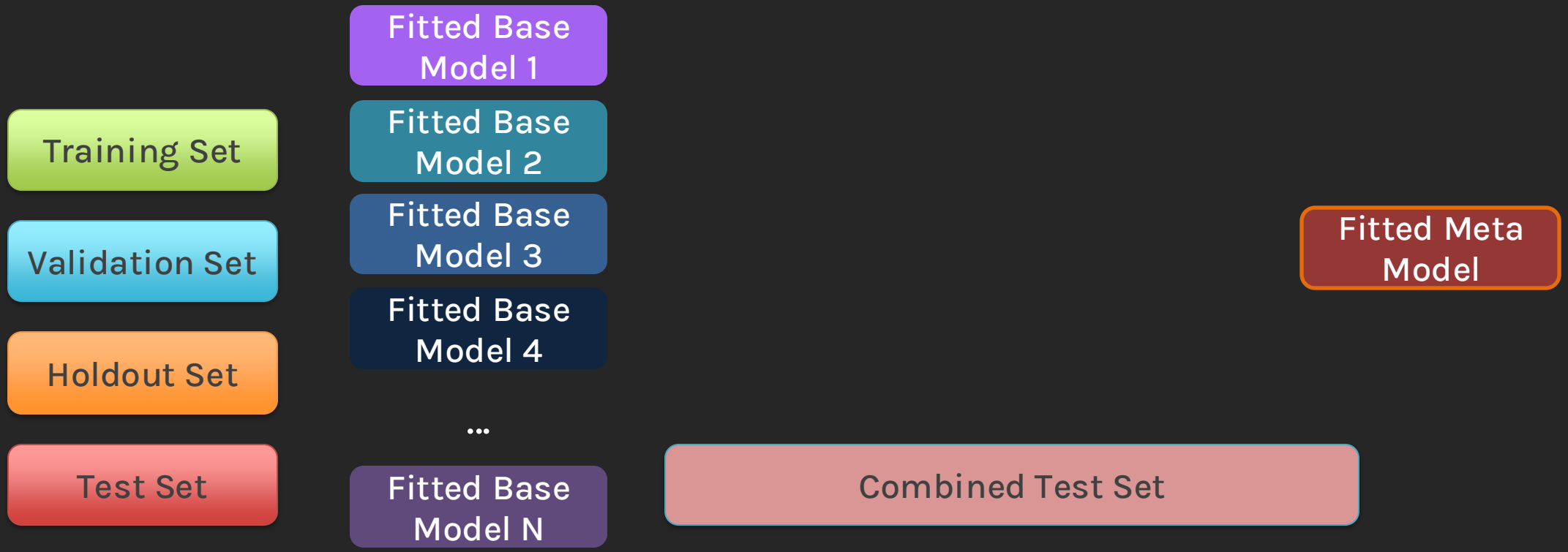
- Generate predictions from base models
- Combine model predictions and original data



Blending

During inference (e.g., evaluation on test set):

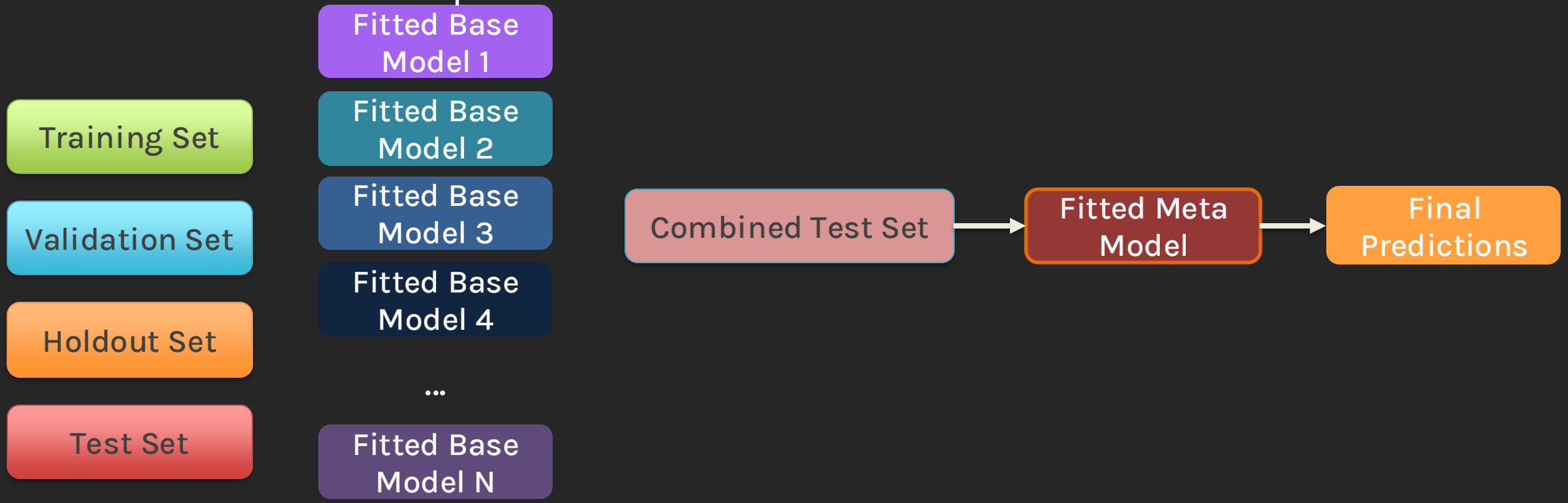
- Generate predictions from base models
- Combine model predictions and original data



Blending

During inference (e.g., evaluation on test set):

- Generate predictions from base models
- Combine model predictions and original data
- Generate model predictions



Blending

Compared with Bagging and Boosting, Blending provides flexibility

- Allows combining models of different types
- Meta model learns how to best combine diverse model outputs

Blending models can be more interpretable if meta-model is simple (e.g. linear regression), where contribution of each base model can be explicitly visible.

But we are dividing the dataset into more pieces for training blending models, and the amount of data for training meta model is only the size of validation set (usually 10%-25% of original data).

Question: Can we get more data for training meta model?

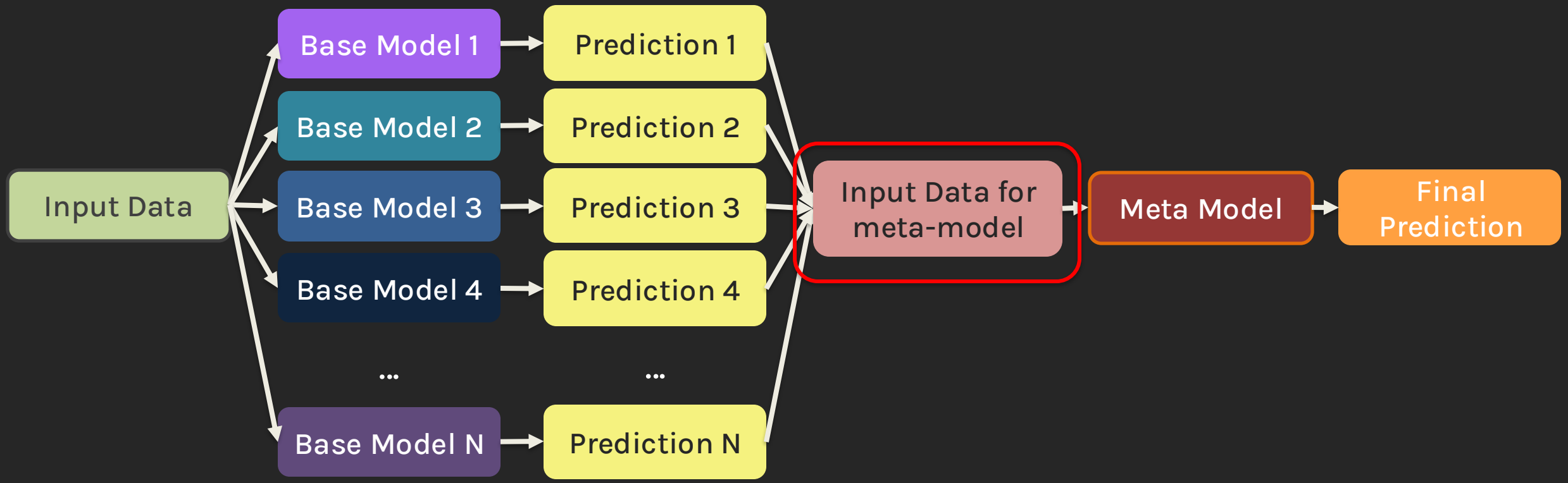
Outline

- Ensemble Methods
- Blending
- **Stacking**
- Mixture of Experts

Stacking

Similar to Blending, Stacking also trains heterogeneous learners on the dataset, and a meta model is fit with base models' predictions.

The difference is how we construct data for meta model.



Stacking

Like Blending, Stacking also trains heterogeneous learners on the dataset, and a meta model is fit with base models' predictions.

The difference is how we construct data for meta model.

"Stacked generalization works by deducing the biases of the generalizer(s) with respect to a provided learning set. This deduction proceeds by generalizing in a second space whose inputs are (for example) the guesses of the original generalizers when taught with part of the learning set and trying to guess the rest of it, and whose output is (for example) the correct guess." Stack Generalization 1992 - D. Wolpert

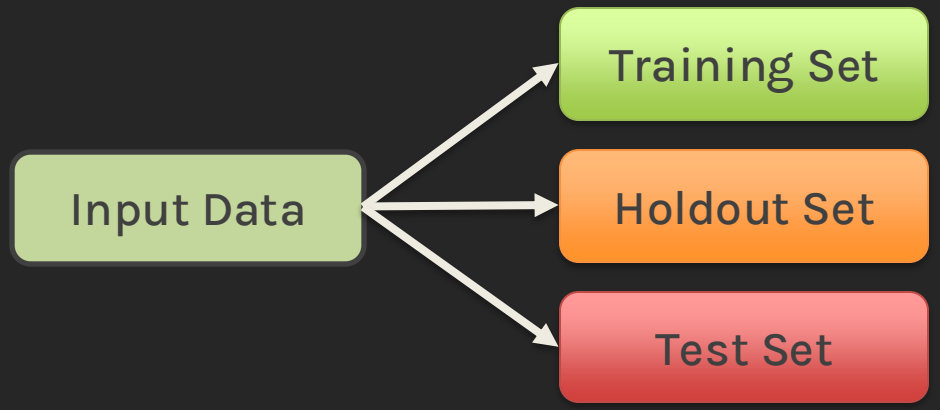
Stacking

Step 1: Similar to blending, we first split the dataset into

- Training set (to train base models)
- Holdout set (to validate meta model)
- Test set (for evaluation)

Note that we don't have validation set here anymore, because we will be doing

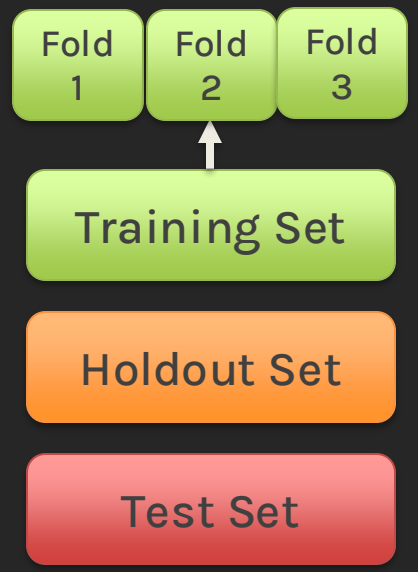
CROSS VALIDATION!



Stacking

Step 1: In cross validation, we need to split the training data into folds.

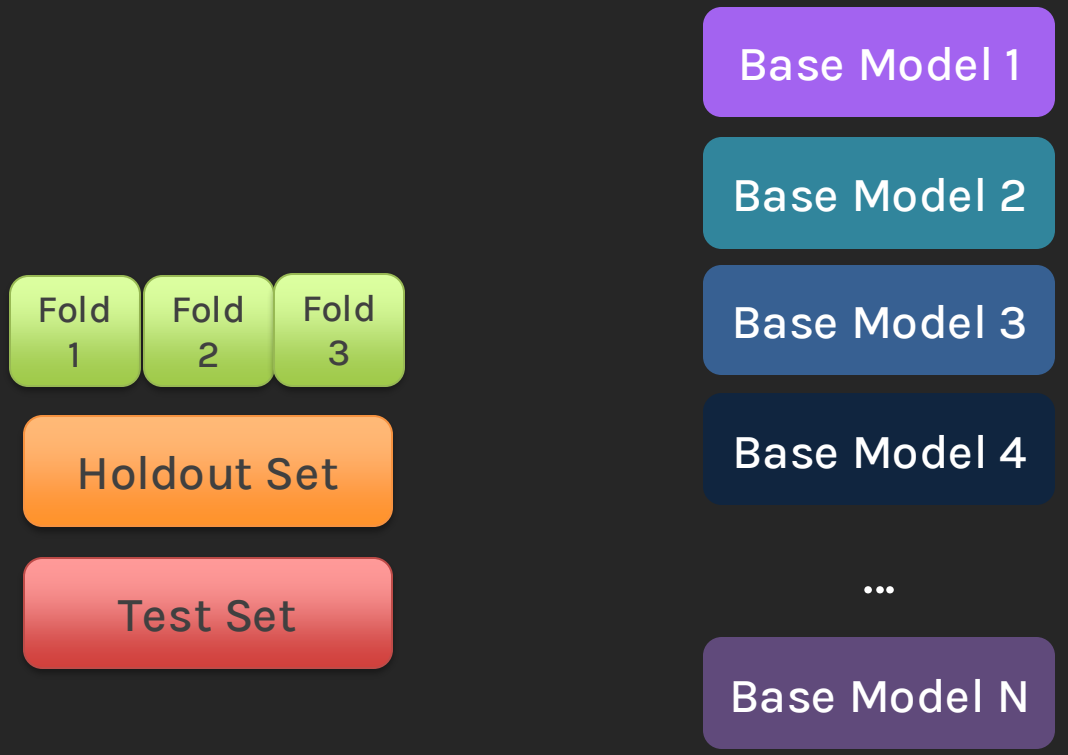
For simplicity here, let's assume we want $k = 3$ folds.



Stacking

Step 2: Fit base models with cross validation. While we do validation with each fold, collect these predictions to build our training set for meta model.

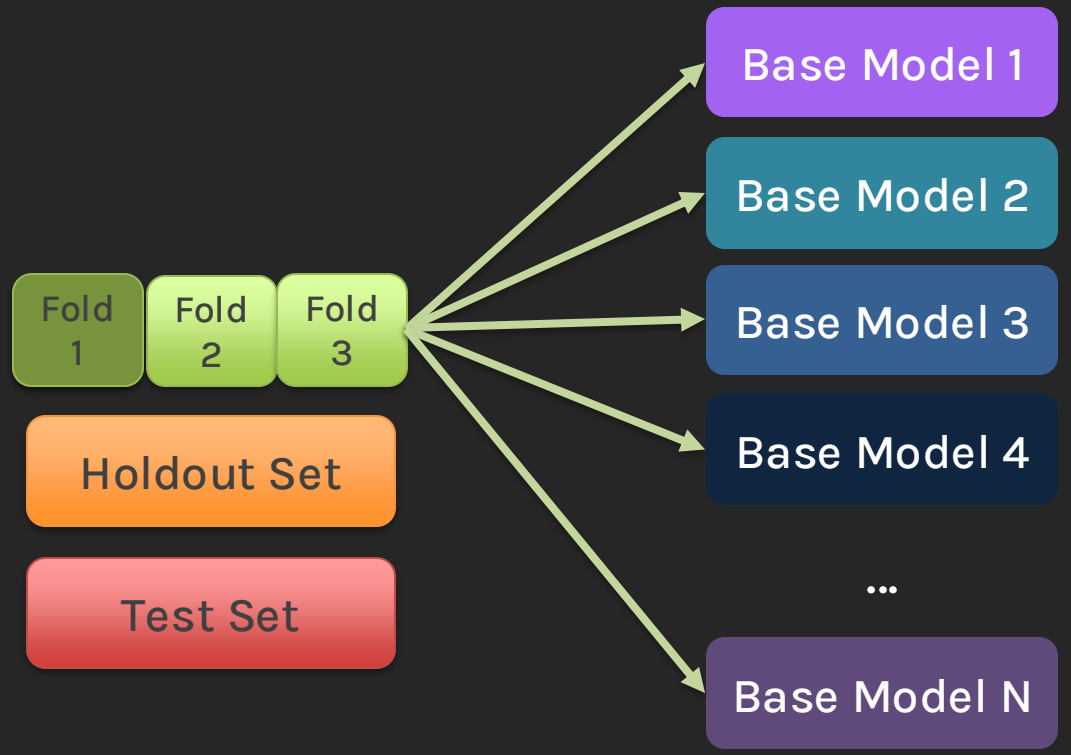
In other words, we concatenate the output of `cross_val_predict()` from each base model into a new dataset.



Stacking

Step 2:

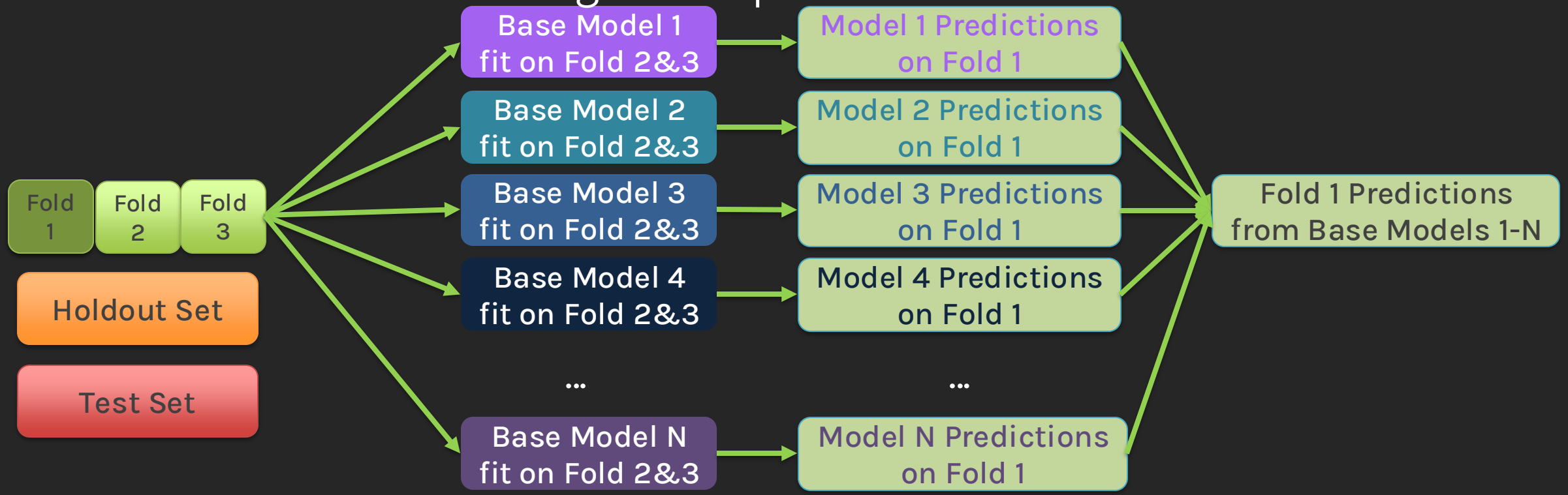
- We firstly take Fold 1 as our validation fold.
- Train on Fold 2 and 3.



Stacking

Step 2:

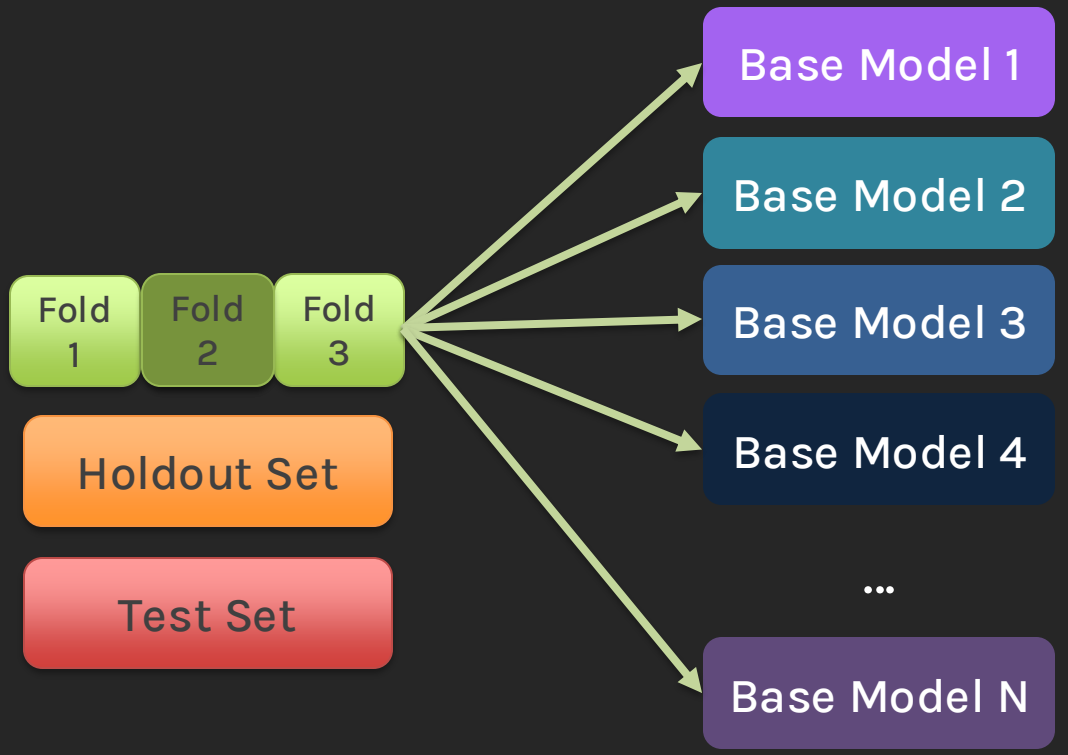
- We firstly take Fold 1 as our validation fold.
- Train on Fold 2 and 3.
- Validate on Fold 1 and generate predictions.



Stacking

Step 2:

- Next, we take Fold 2 as our validation fold.
- Train on Fold 1 and 3.

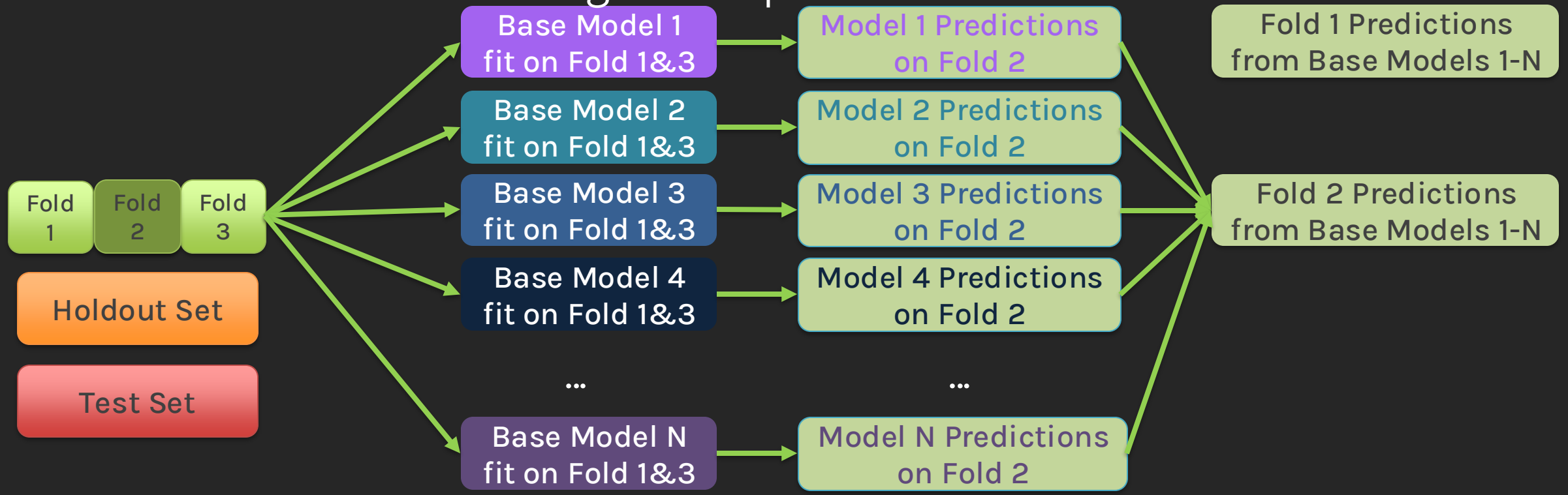


Fold 1 Predictions from Base Models 1-N

Stacking

Step 2:

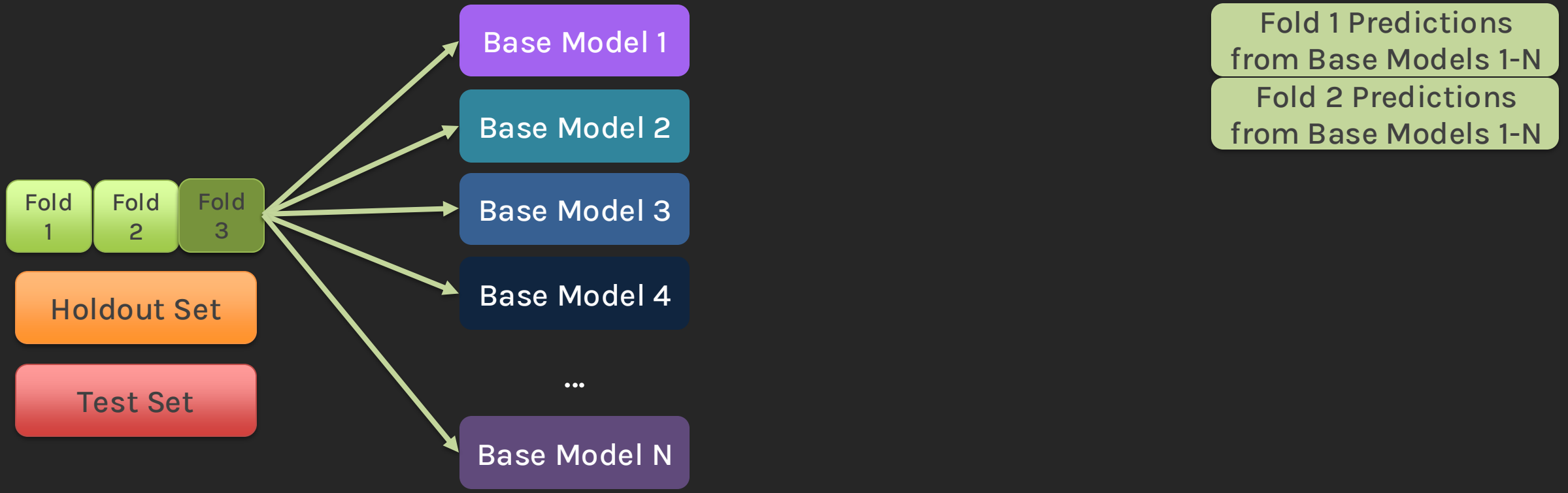
- Next, we take Fold 2 as our validation fold.
- Train on Fold 1 and 3.
- Validate on Fold 2 and generate predictions.



Stacking

Step 2:

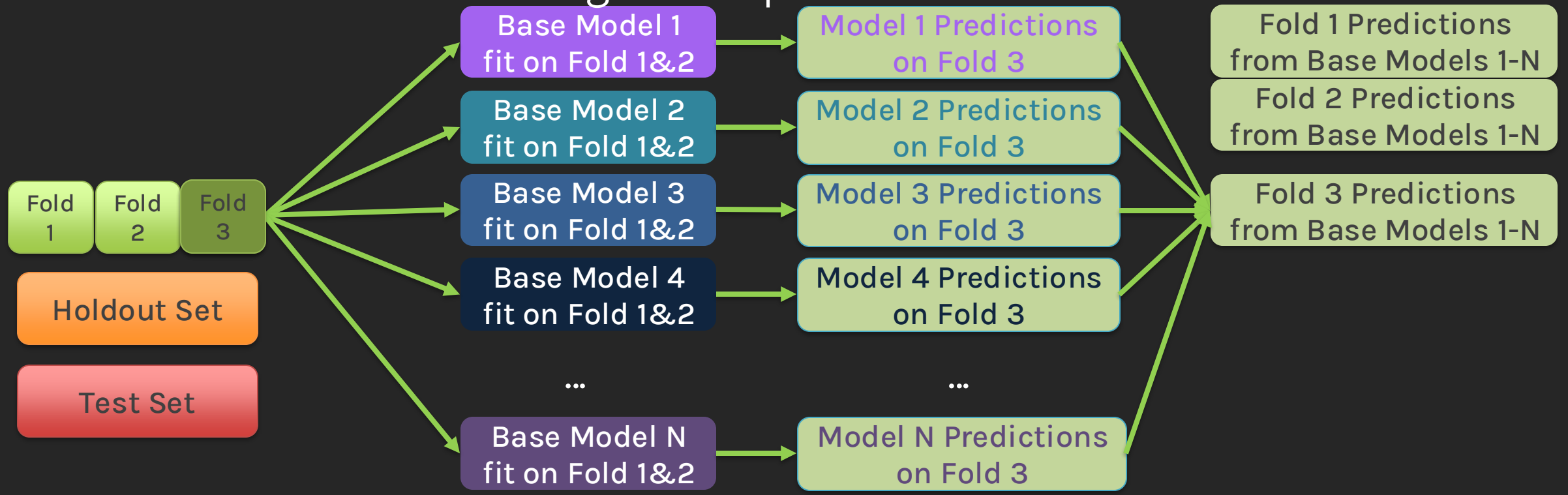
- Now we take Fold 3 as our validation fold.
- Train on Fold 1 and 2.



Stacking

Step 2:

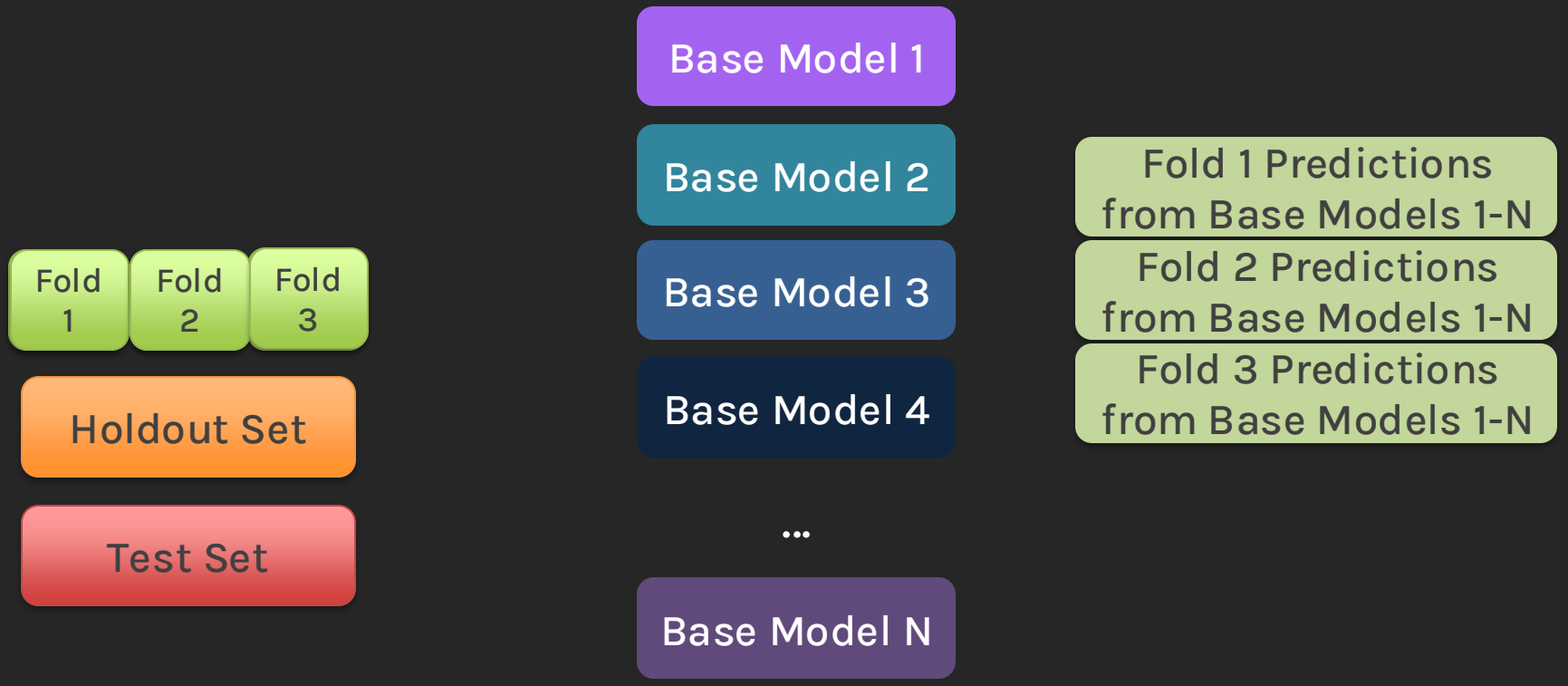
- Now we take Fold 3 as our validation fold.
- Train on Fold 1 and 2.
- Validate on Fold 3 and generate predictions.



Stacking

Step 2:

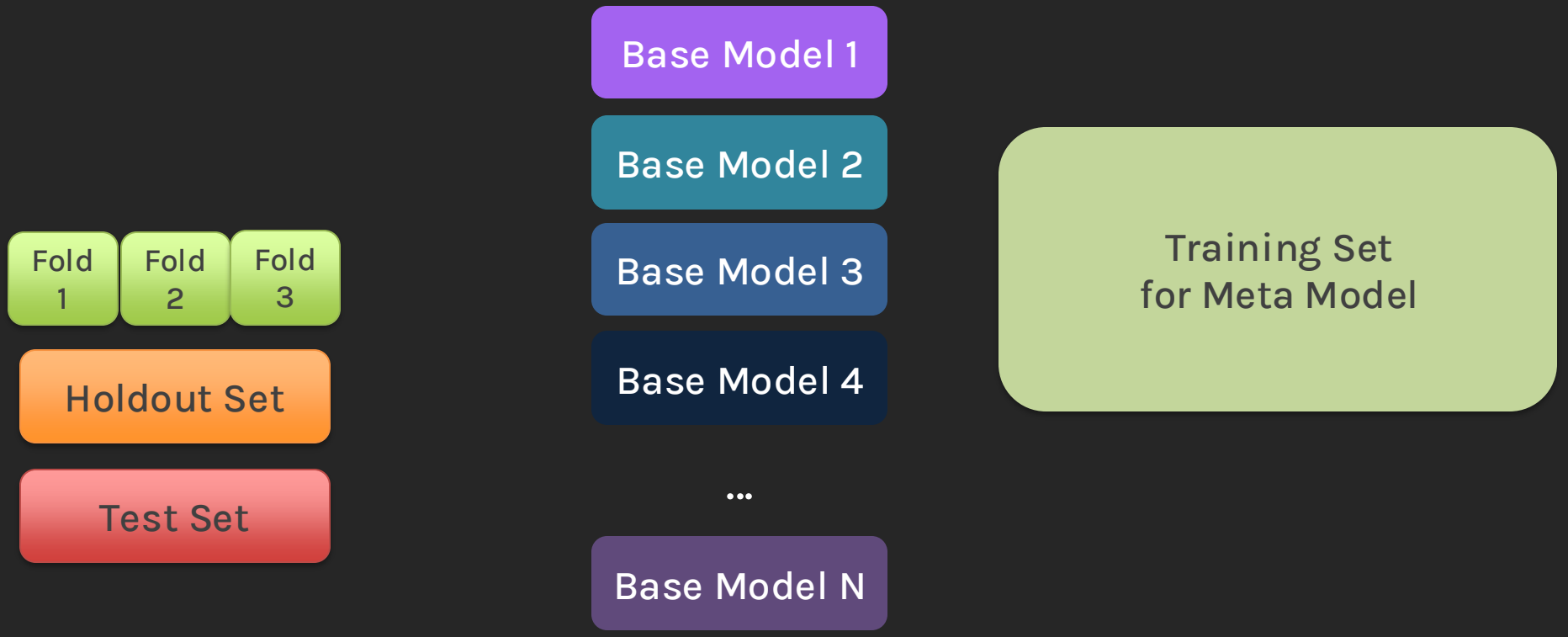
Combine the training folds with the base model predictions, and this is the training set for meta model. Note that different from blending, the size of training set is the same as the original training set!



Stacking

Step 2:

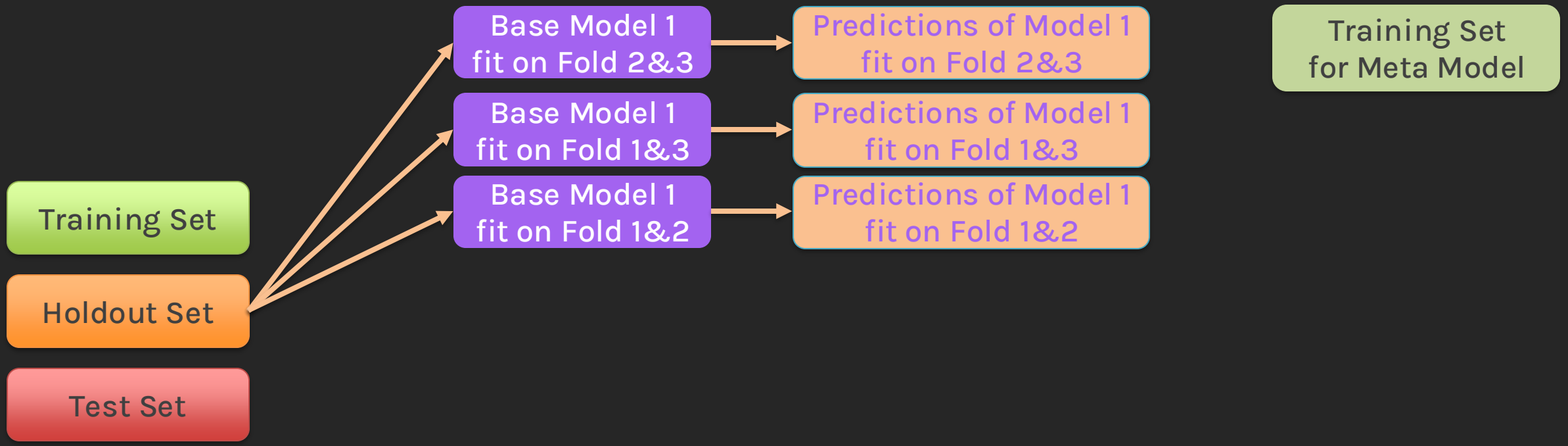
Combine the training folds with the base model predictions, and this is the training set for meta model. Note that different from blending, the size of training set is the same as the original training set!



Stacking

Step 2:

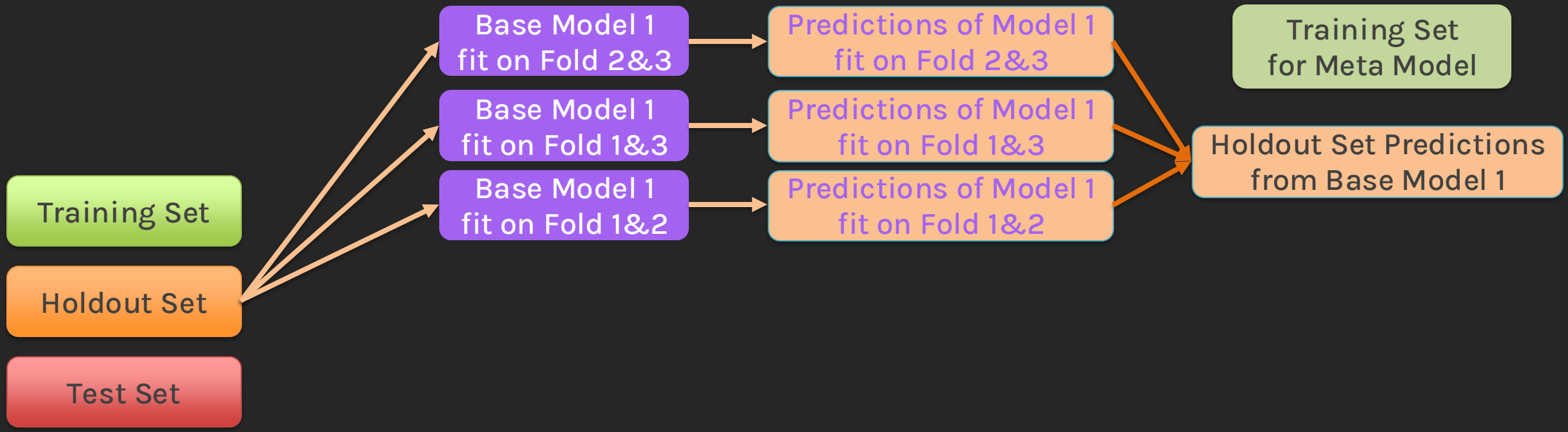
Meanwhile, in each fold round of cross validation, we want to make predictions on holdout set.



Stacking

Step 3:

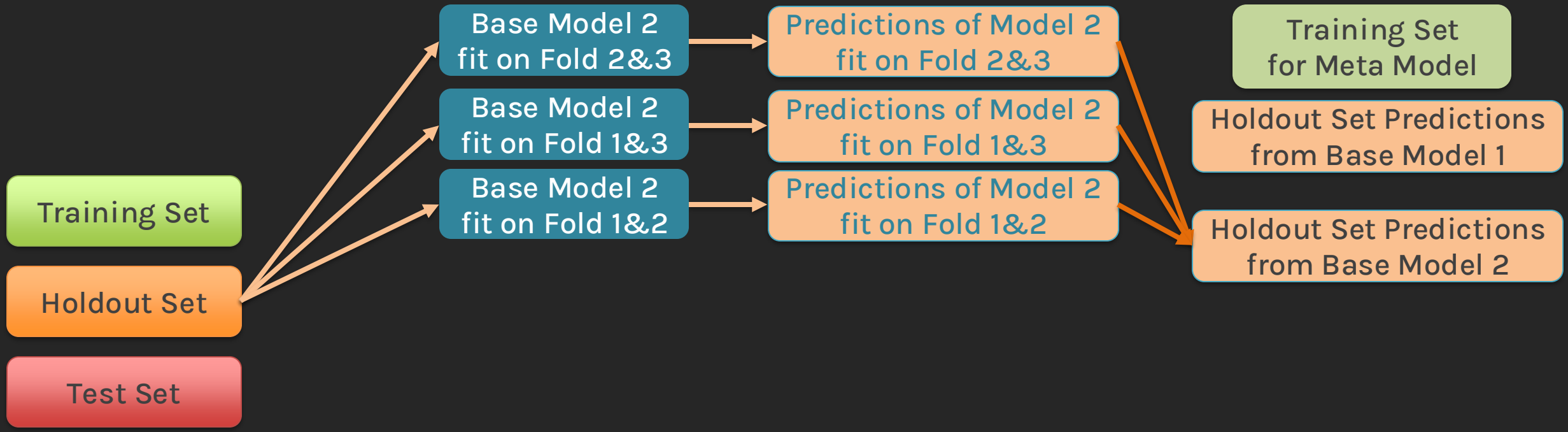
Meanwhile, in each fold round of cross validation, we want to make predictions on holdout set. Then we take the average across the folds (i.e., we have one group of holdout set predictions per base model).



Stacking

Step 3:

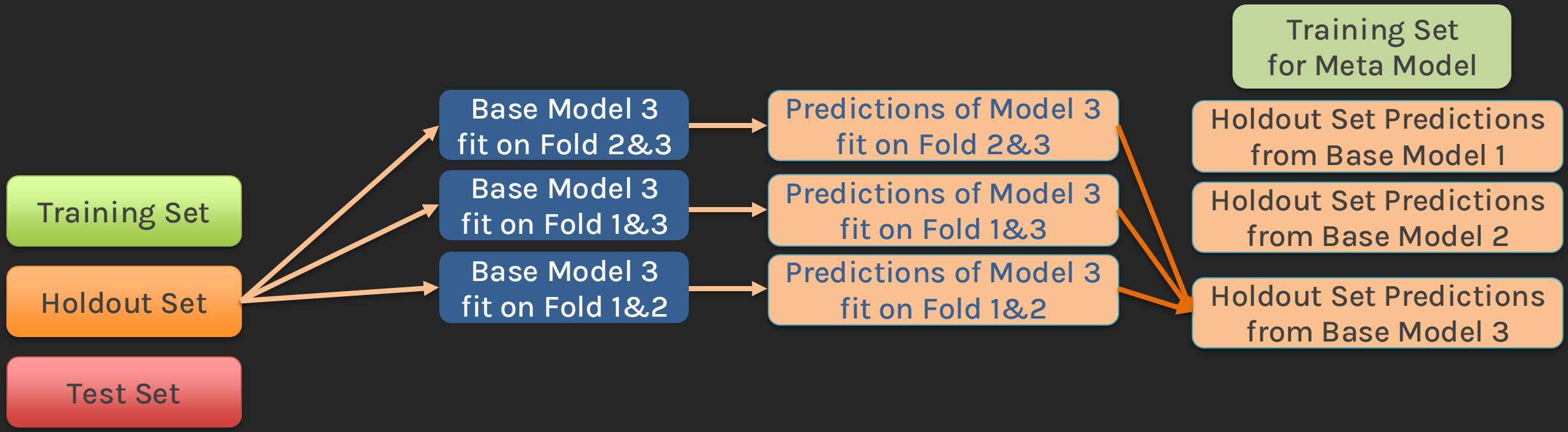
We repeat for all other base models. Generate holdout set predictions and take the average.



Stacking

Step 3:

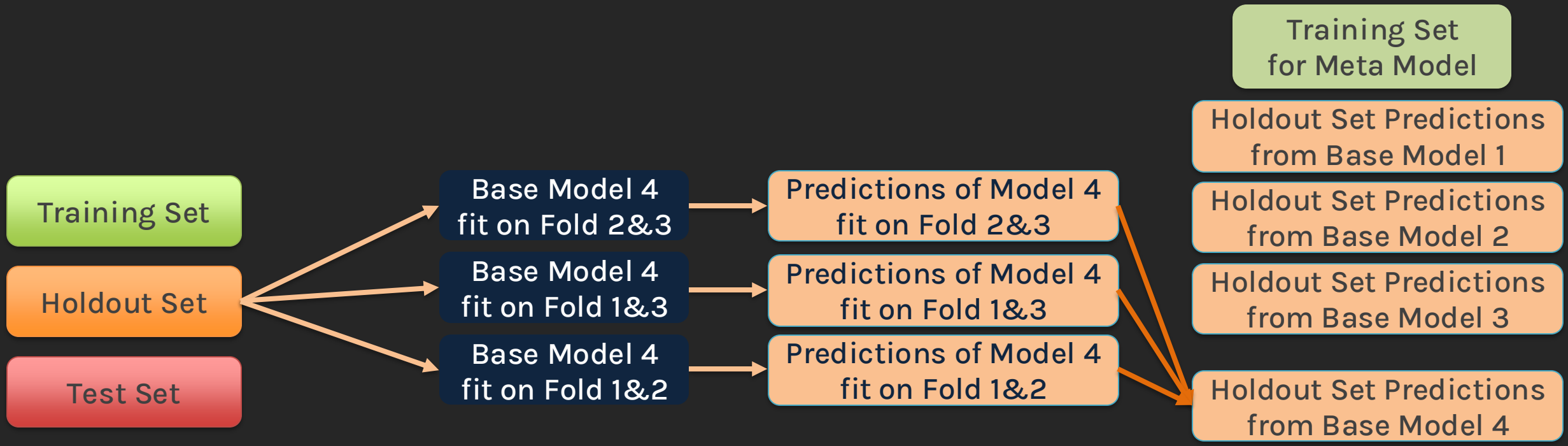
We repeat for all other base models. Generate holdout set predictions and take the average.



Stacking

Step 3:

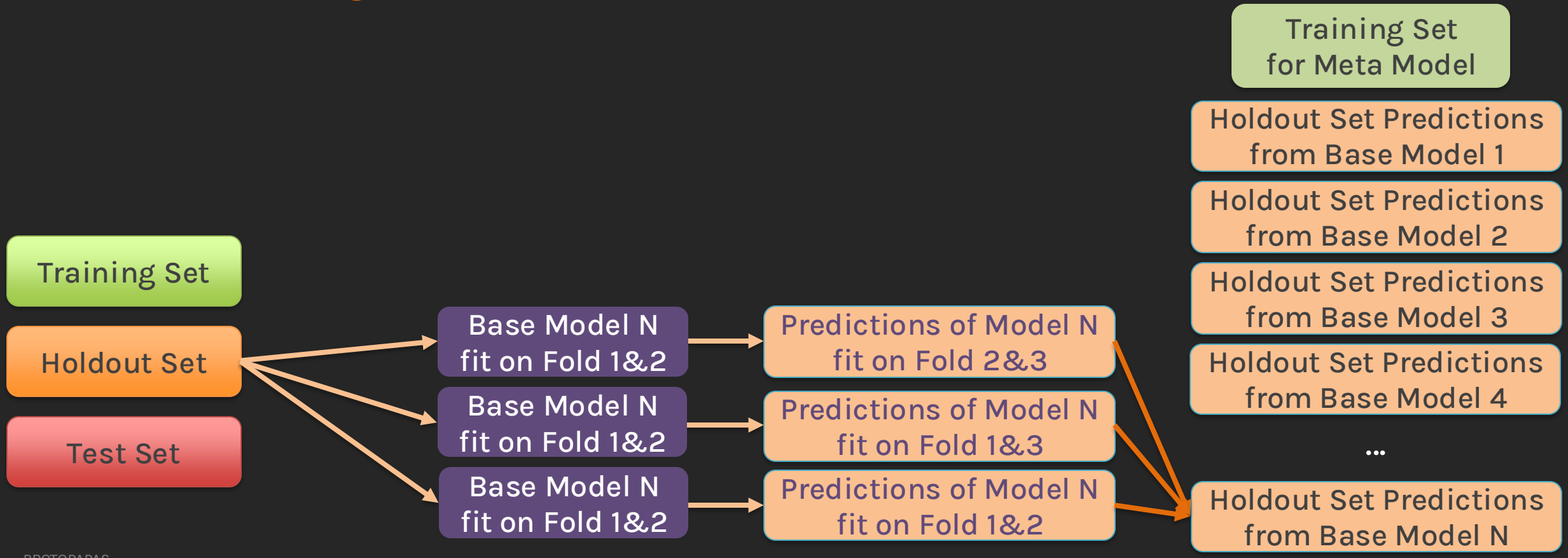
We repeat for all other base models. Generate holdout set predictions and take the average.



Stacking

Step 3:

We repeat for all other base models. Generate holdout set predictions and take the average.



Stacking

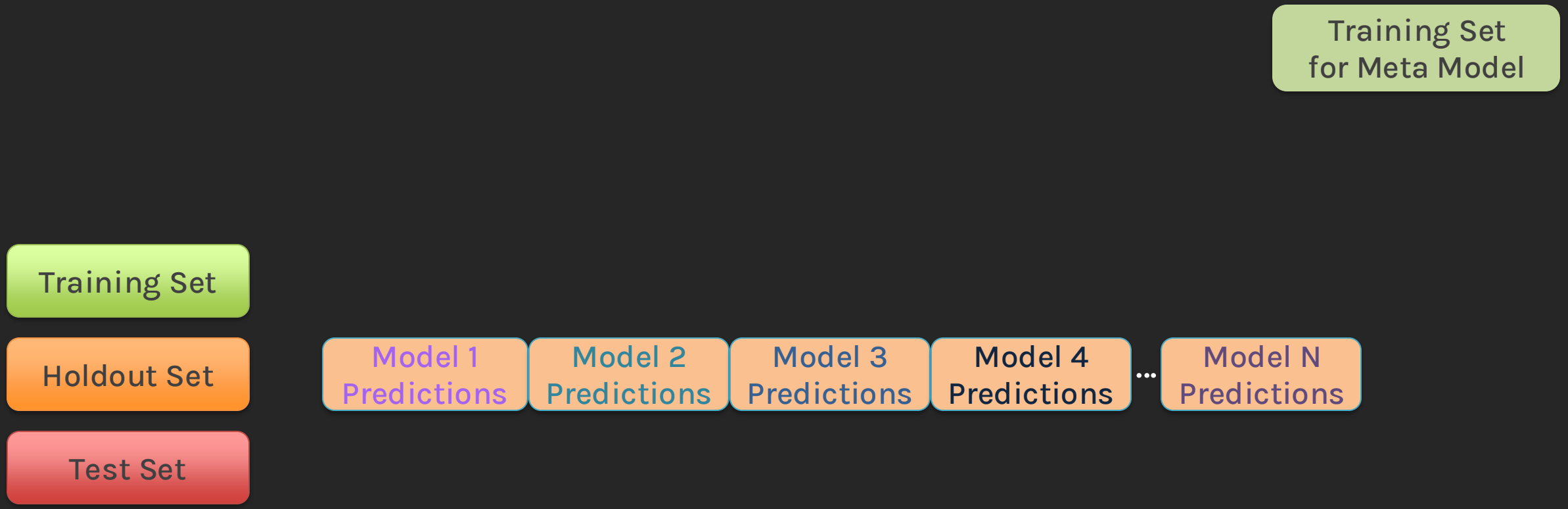
Step 3:

We repeat for all other base models. Generate holdout set predictions and take the average.

Stacking

Step 3:

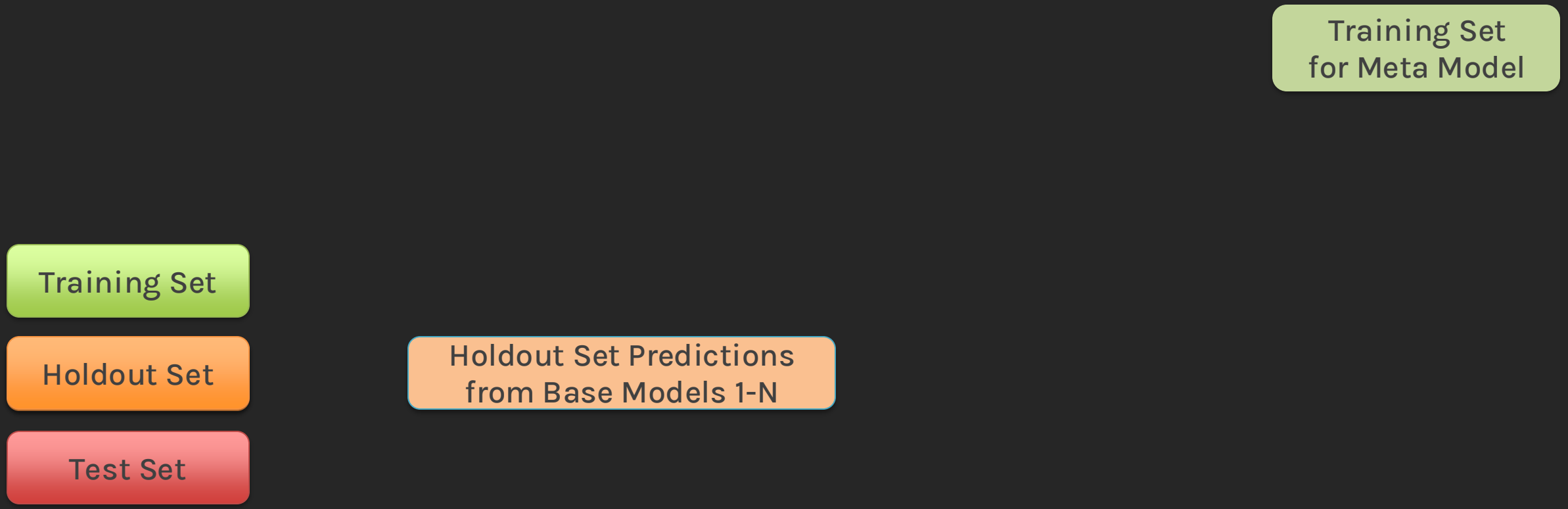
Combined features from holdout set will be ready for validating meta model.



Stacking

Step 3:

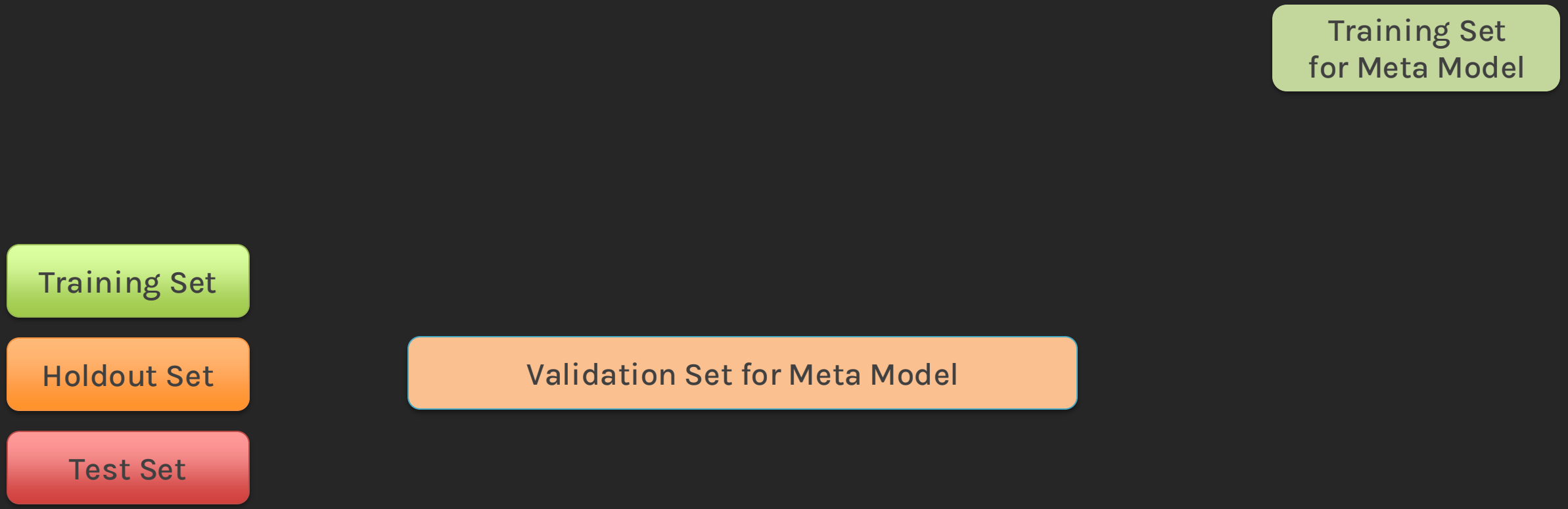
Combined features from holdout set will be ready for validating meta model.



Stacking

Step 3:

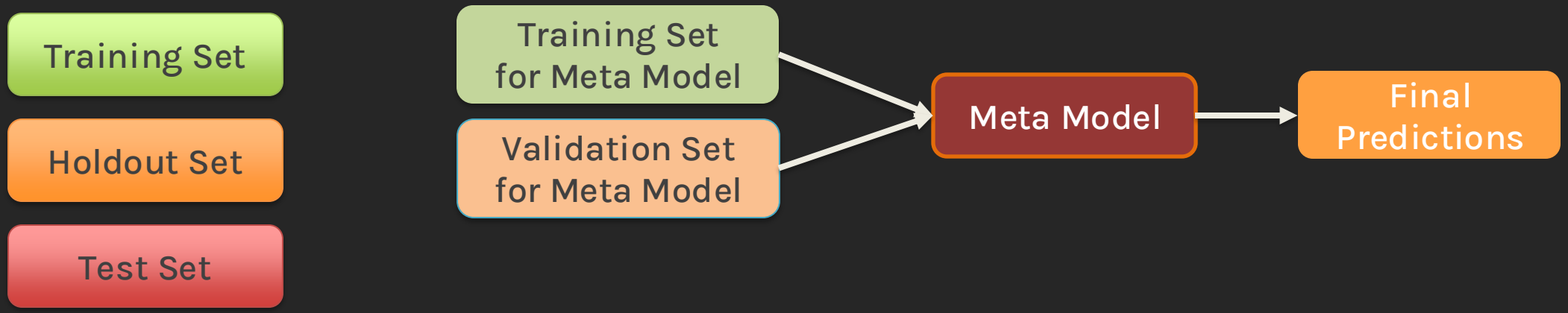
Combined features from holdout set will be ready for validating meta model.



Stacking

Step 4:

The remaining steps of fitting meta model and making predictions are the same as blending.



Important Note with sklearn

Note that we have used both the base model predictions and original data for training meta model when introducing blending and stacking.

In practice, we can also use the combination of only base model predictions and response variables (only y).

Although less features are provided to meta model, this avoids meta model from overfitting to the original data and makes it learn the right balance of predictions from base models.

This is controlled by a Boolean argument `passthrough` in sklearn's `StackingRegressor()` and `StackingClassifier()`. The default value is False (meaning meta model doesn't see the original data).

How to combine outputs from base models?

Model outputs can be different depending on the tasks.

Regression tasks are easier to understand, as the model outputs a single value for each sample.

The outputs of classifiers are usually in the format of probabilities for each class. How do we combine them?

- For binary classification, class 0 and class 1 probabilities are perfectly collinear, so we only keep one of them (e.g. take predicted probability for class 1 as the model output).
- For multi-class classification, all probabilities sum to one for each sample, so we keep 'num_classes-1' classes. Usually the probabilities for class 0 (first column) are dropped.

A Quick Summary

With ensemble methods, we can combine models to obtain a more accurate and/or more robust model, including

- Bagging and Boosting with homogeneous models
- Blending and Stacking with heterogenous models

The base models in the ensemble are combined in a cooperative way, and to achieve this, we need all base models to fit on the entire training set and learn data distributions and patterns over the complete range.

This could be potentially hard for base models if the dataset contains several different regimes, which have different relationships between input and output.

Is it possible to cover different input regions with different learners?

Outline

- Ensemble Methods
- Blending
- Stacking
- **Mixture of Experts**

Mixture of Experts

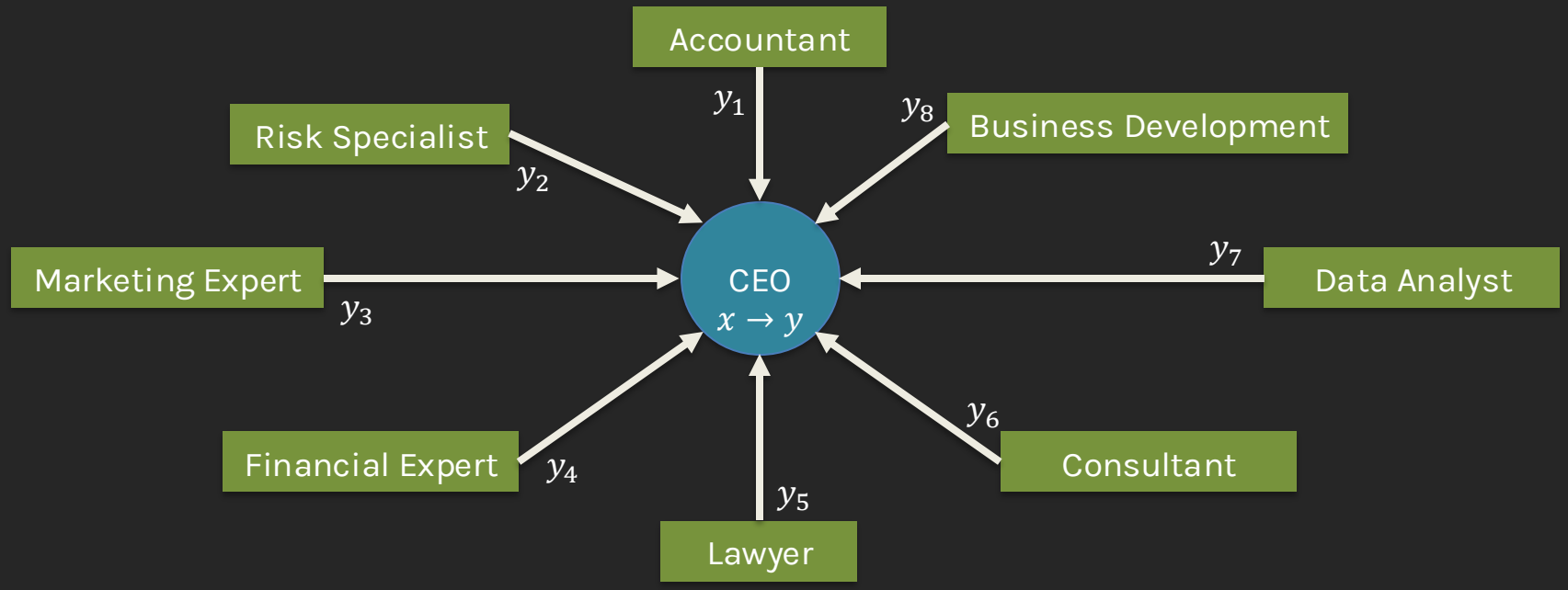
Intuition: Consider the case of a physical examination, and the goal is to have the diagnosis if the patients have any severe diseases.

Patients will go to different consultation rooms, and there will be experts of different medical fields waiting to check different indicators.

You will get a report in the end summarized by some professional doctors or even computer programs, perhaps with an emphasis on some particular sections you want to pay more attention to.

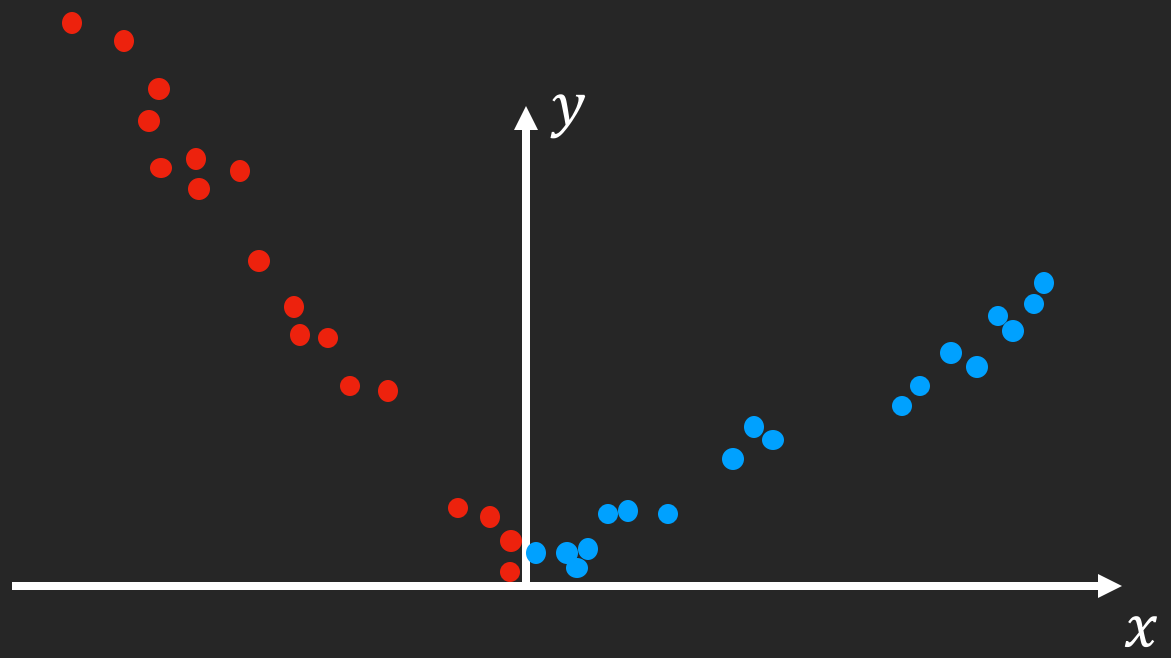
Mixture of Experts

Another Intuition: Consider a regular company with all the different departments. If the CEO wants to make an important decision, he/she may need to consult the professionals from each department to get a comprehensive plan.



Specialization instead of Cooperation

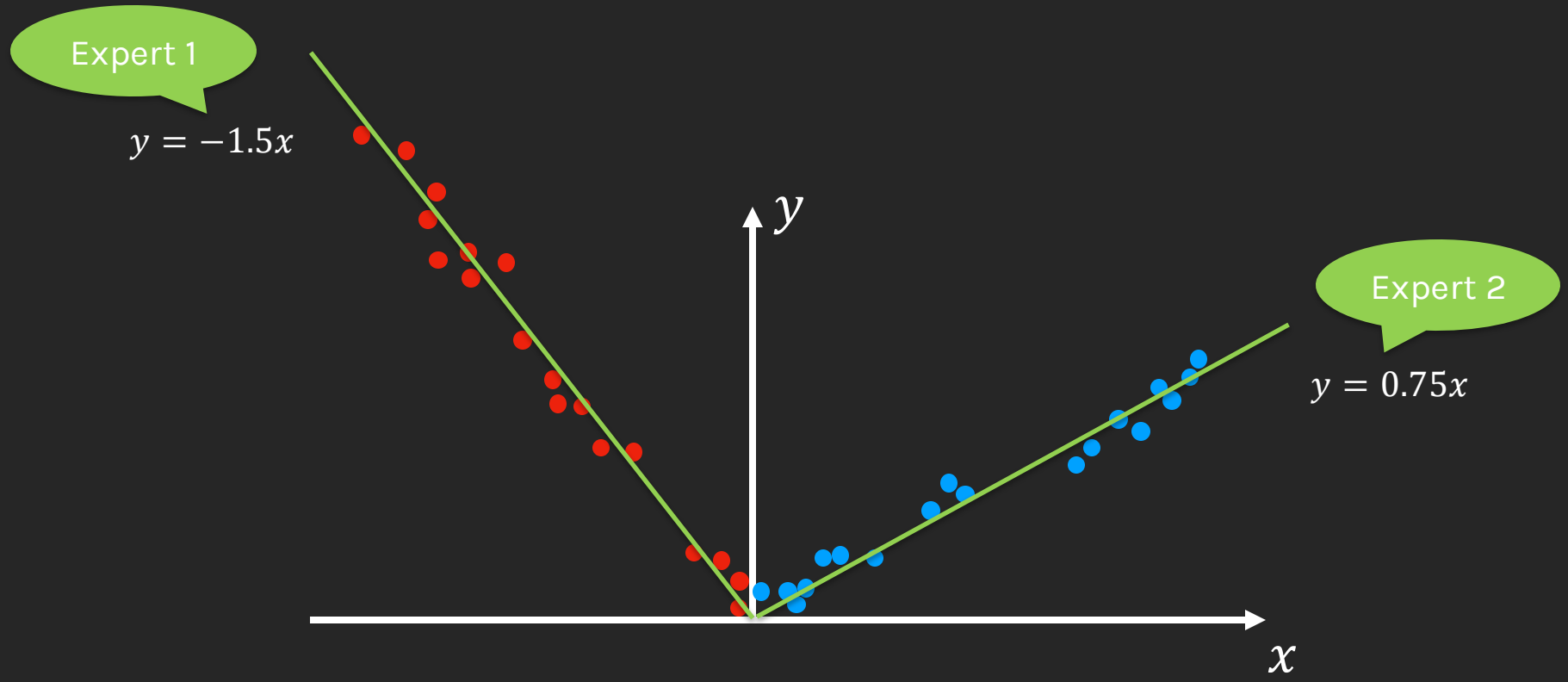
Given the following data example, we have two different data regimes for positive x and negative x values.



Specialization instead of Cooperation

Given the following data example, we have two different data regimes for positive x and negative x values.

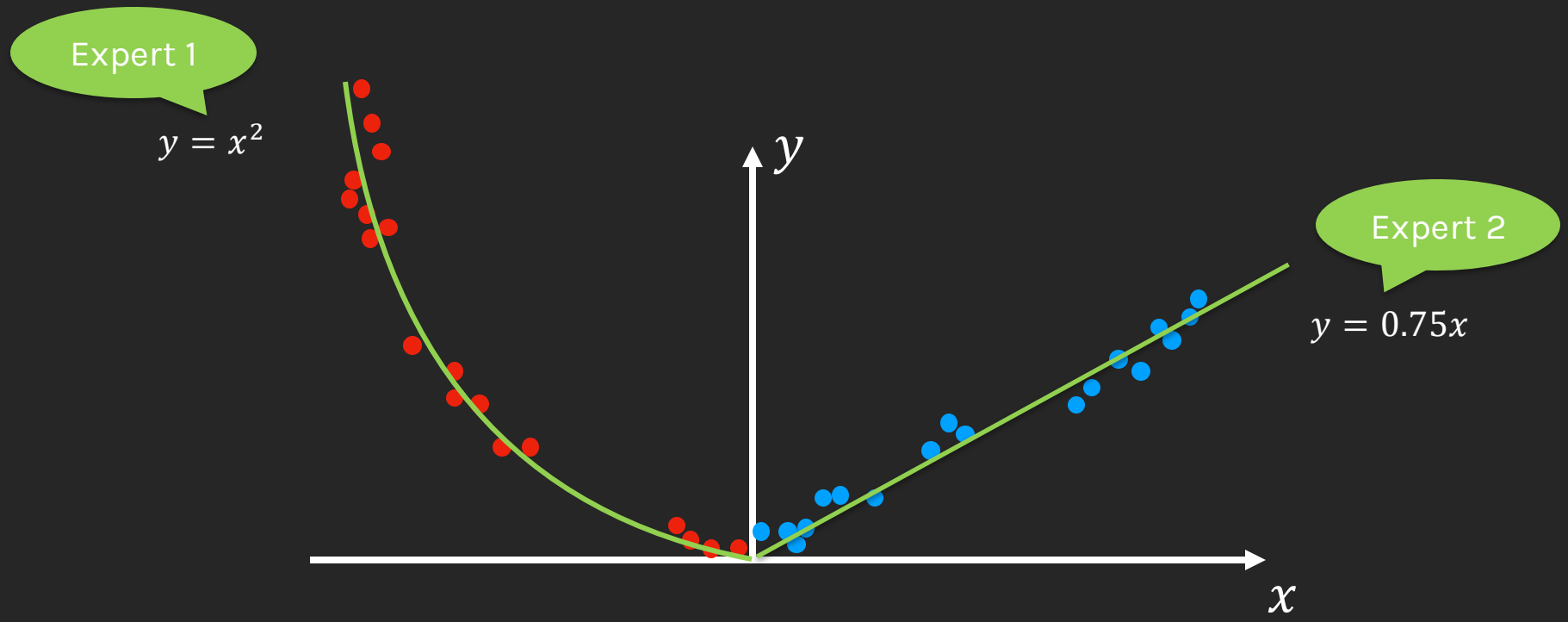
This can be easy if we have two different learners fit on each input region!



Specialization instead of Cooperation

Given the following data example, we have two different data regimes for positive x and negative x values.

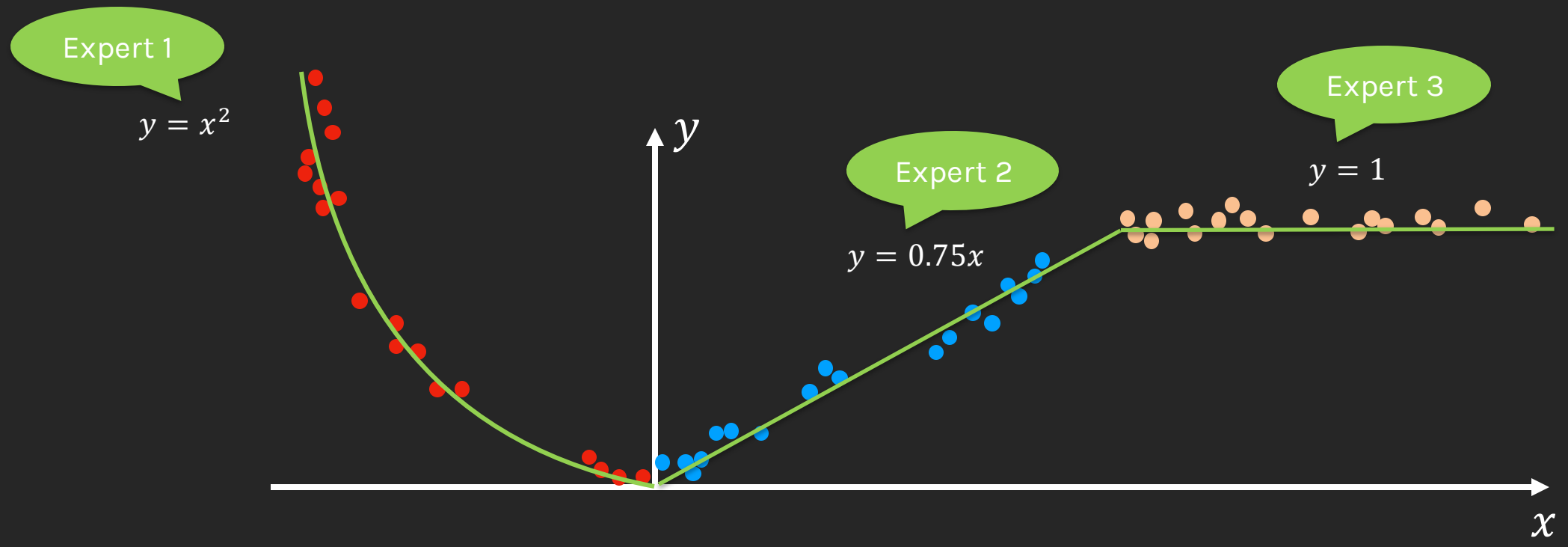
It is still easy even different families of models are involved in two regions!



Specialization instead of Cooperation

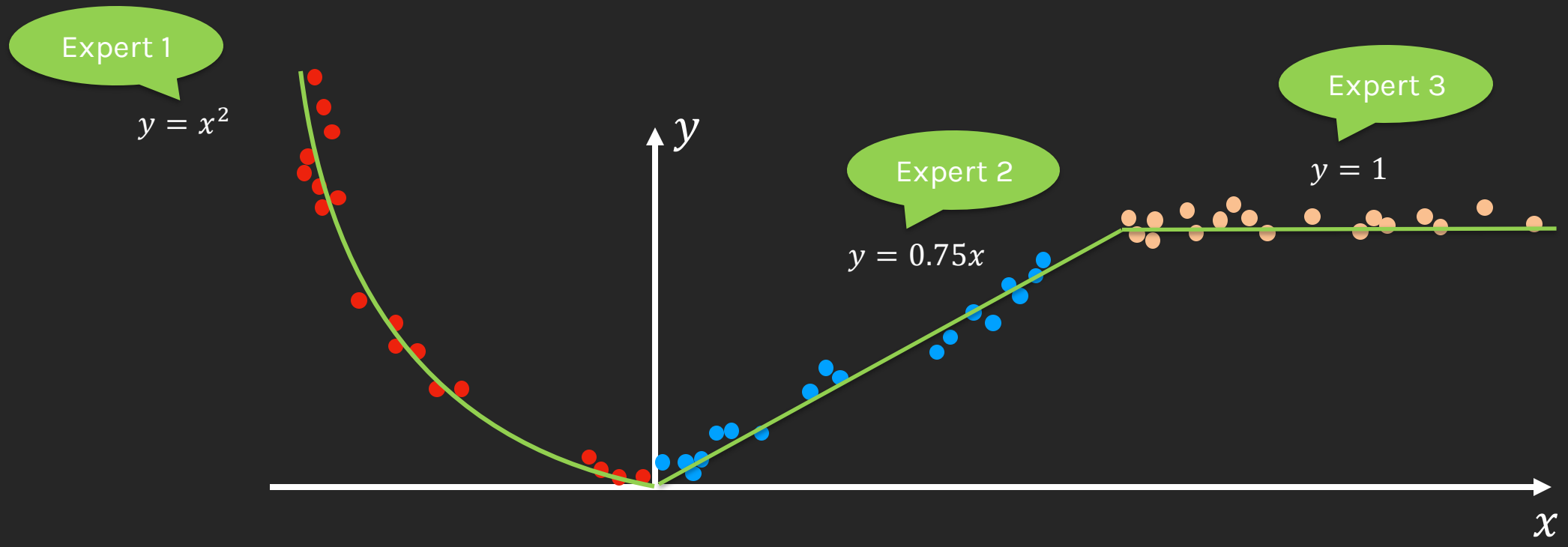
Given the following data example, we have two different data regimes for positive x and negative x values.

More data relationships? Just get another expert!



Specialization instead of Cooperation

It is easy to tell that 3 experts are needed here based on (x, y) distributions.

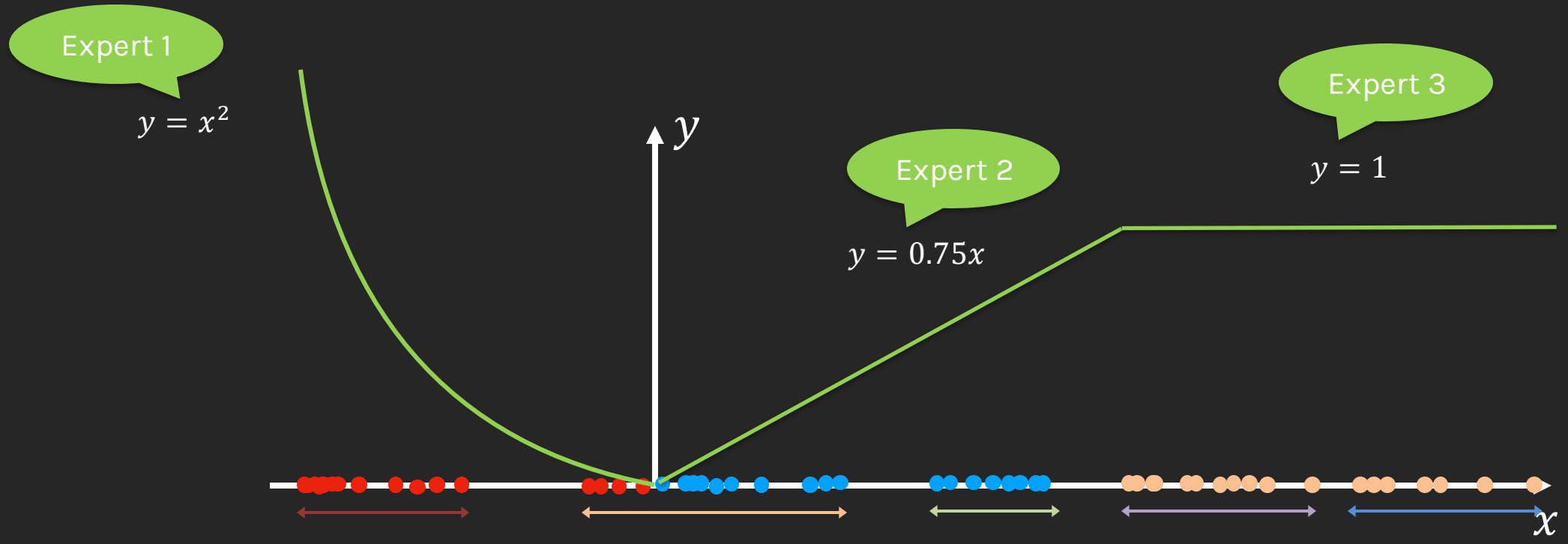


Specialization instead of Cooperation

It is easy to tell that 3 experts are needed here based on (x, y) distributions.

But to identify the domains, we usually only have this to start with.

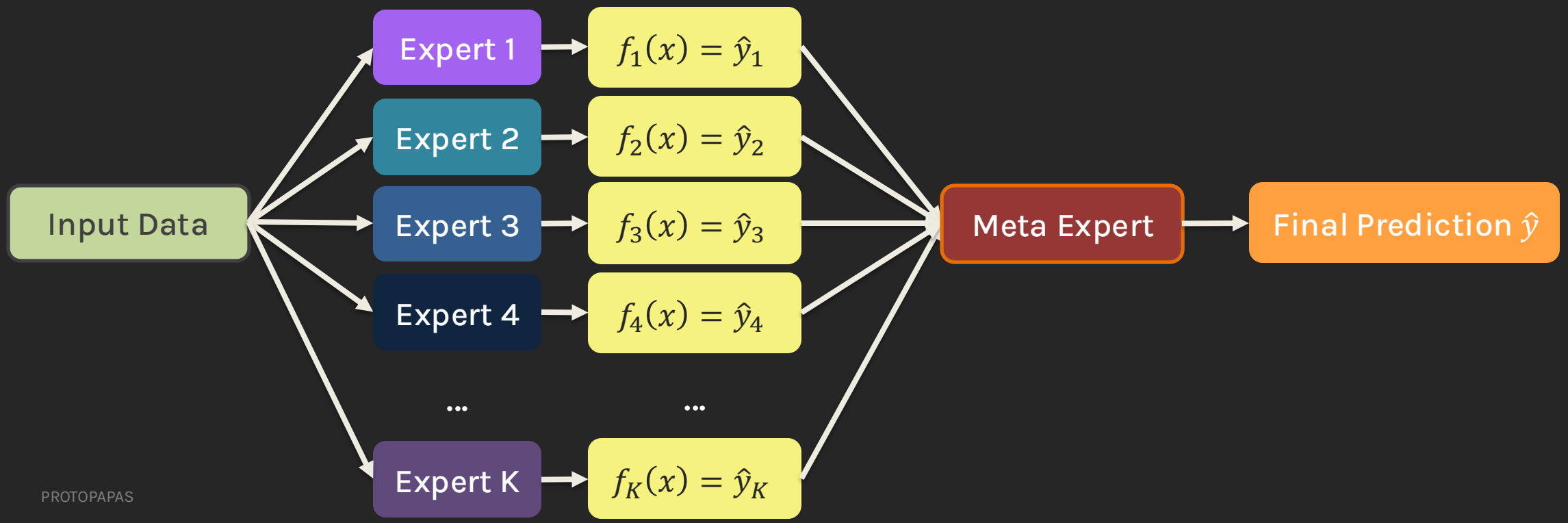
How do we assign these points to the correct expert?



Mixture of Experts

We begin with a framework similar to blending/stacking.

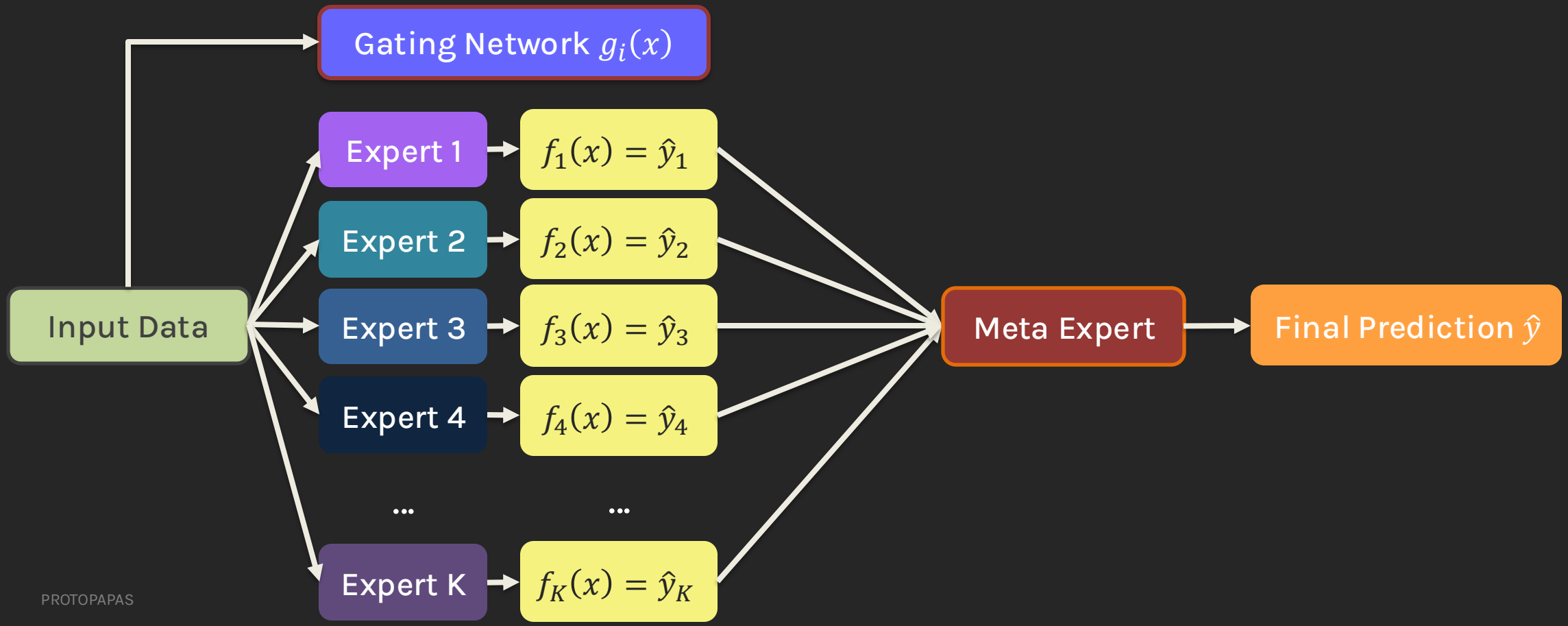
- Each trained expert will make predictions $\hat{y}_i$
- A meta expert will learn from experts to make final prediction $\hat{y}$.



Mixture of Experts

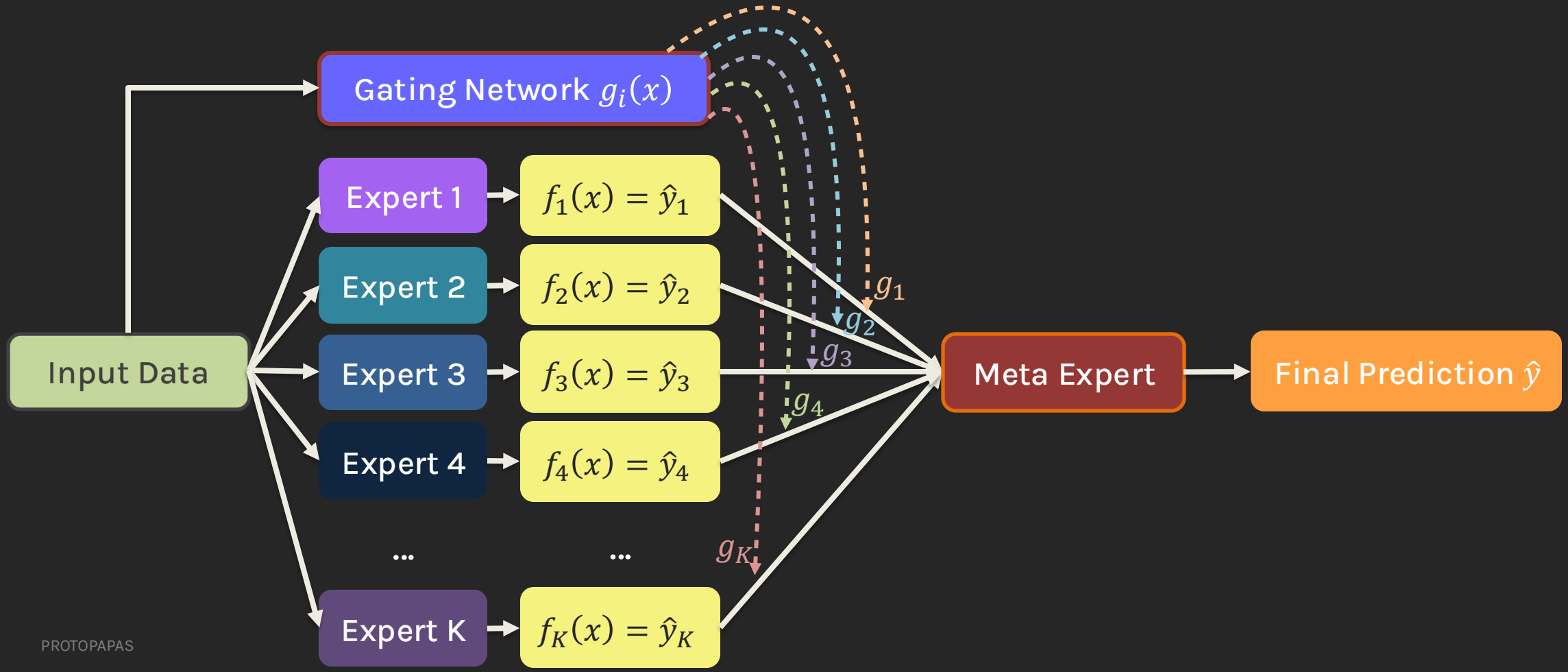
What makes Mixture of Experts special and different?

We have a gating network!



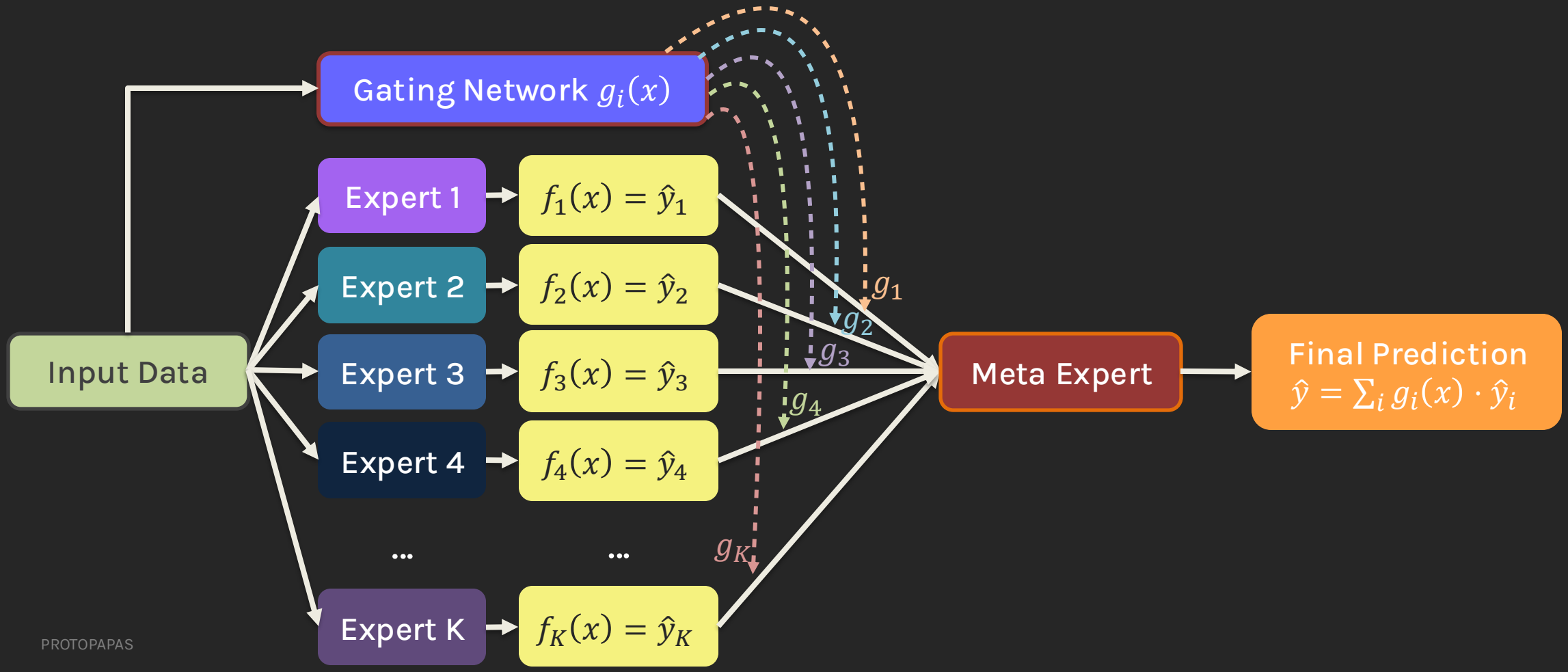
Mixture of Experts

Gating network uses gating functions g_i to decide what combination of experts to use.



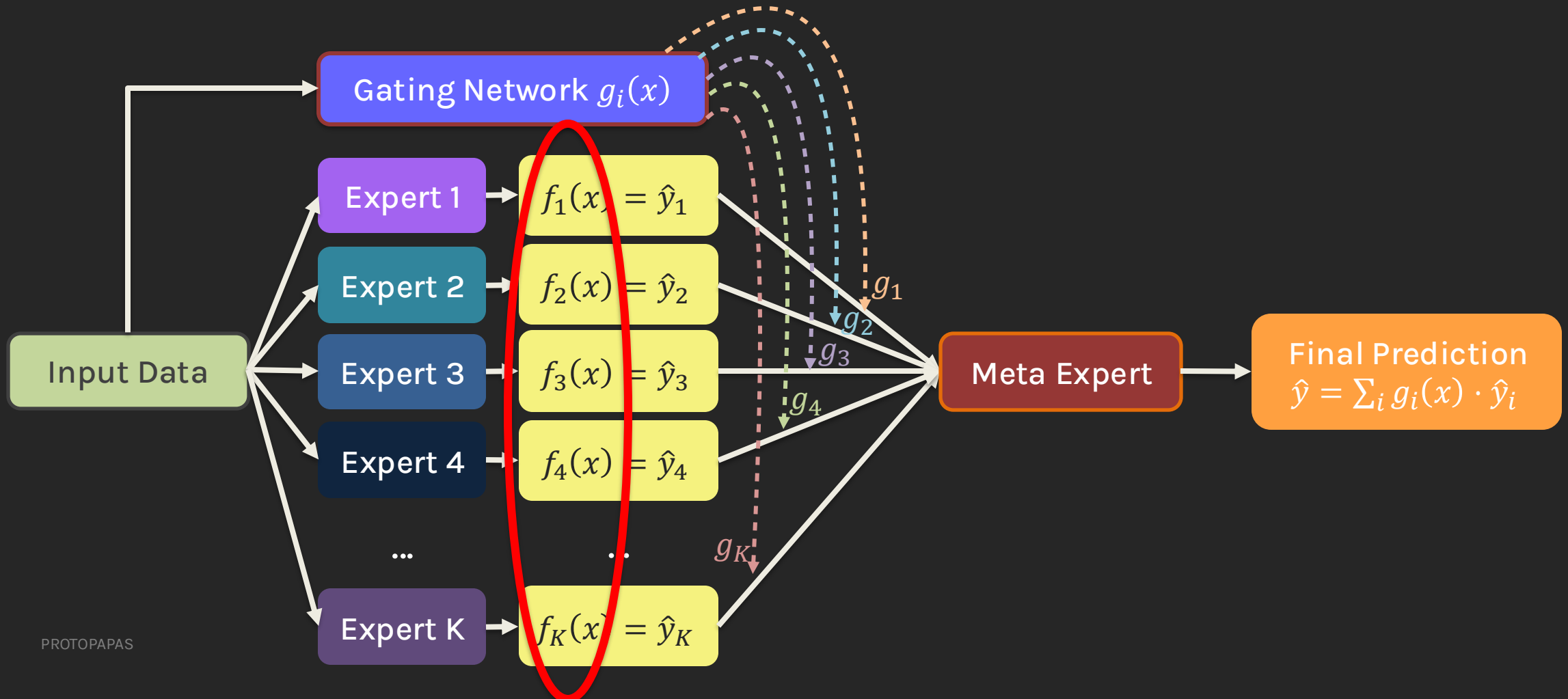
Mixture of Experts

Gating network uses gating functions g_i to decide what combination of experts to use.



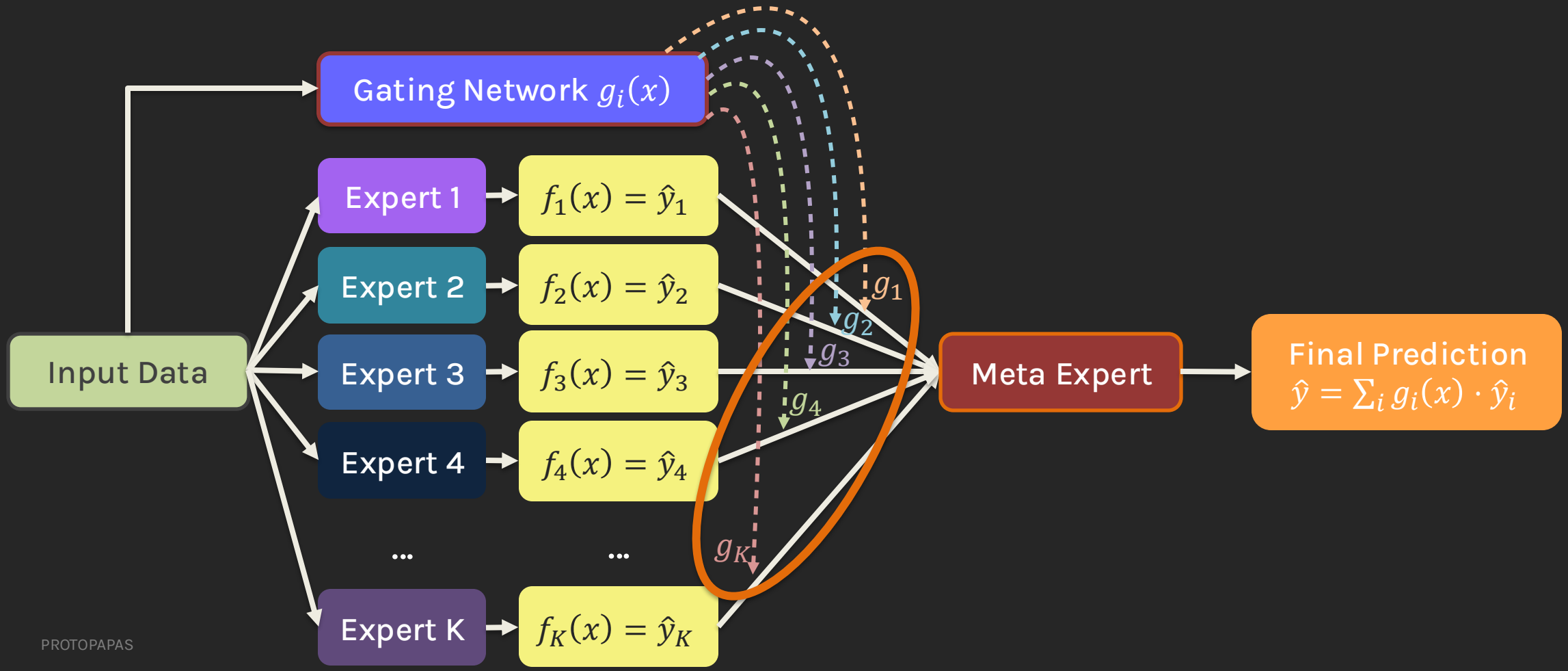
Mixture of Experts

Fitting the model involves 1) learning the parameters of each expert,



Mixture of Experts

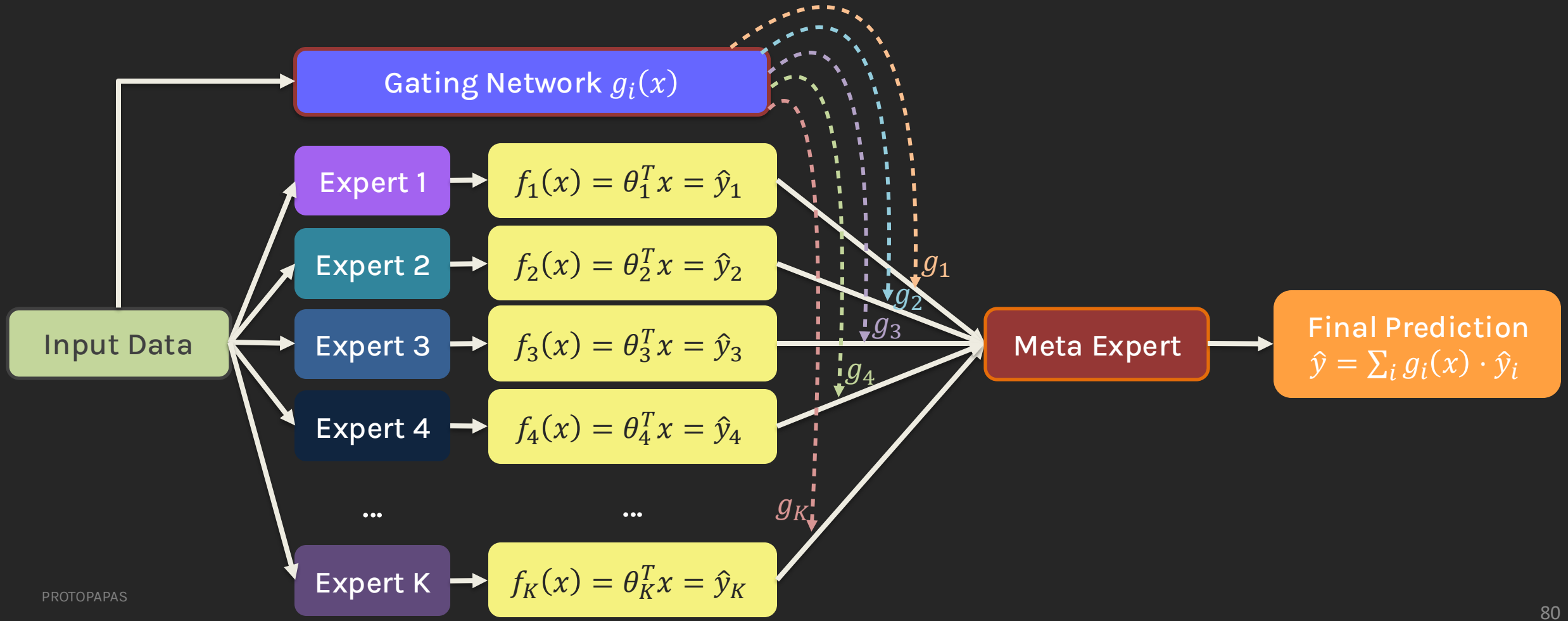
Fitting the model involves 1) learning the parameters of each expert, and 2) learning the parameters of the gating network.



Mixture of Experts

For simplicity here, let's assume the experts are a family of linear models.

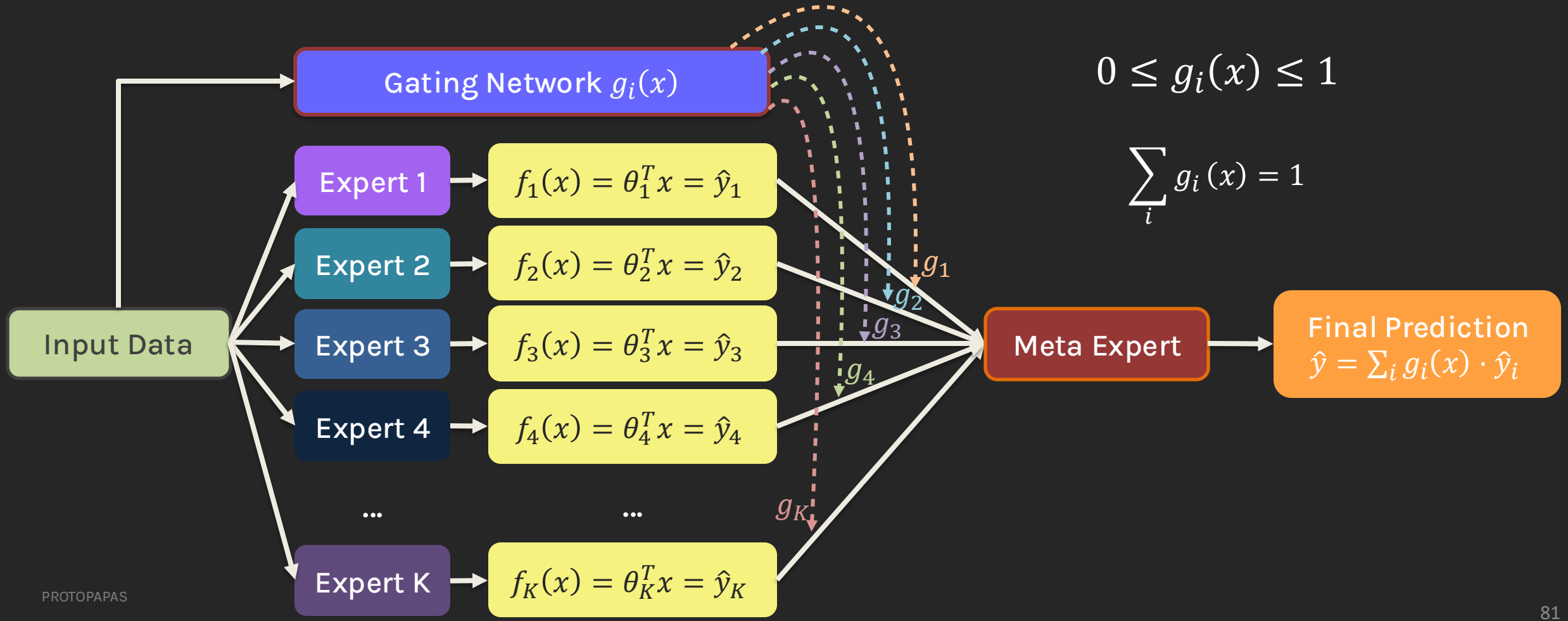
Note that these models don't have to be the same/homogeneous.



Mixture of Experts

What about the gating network?

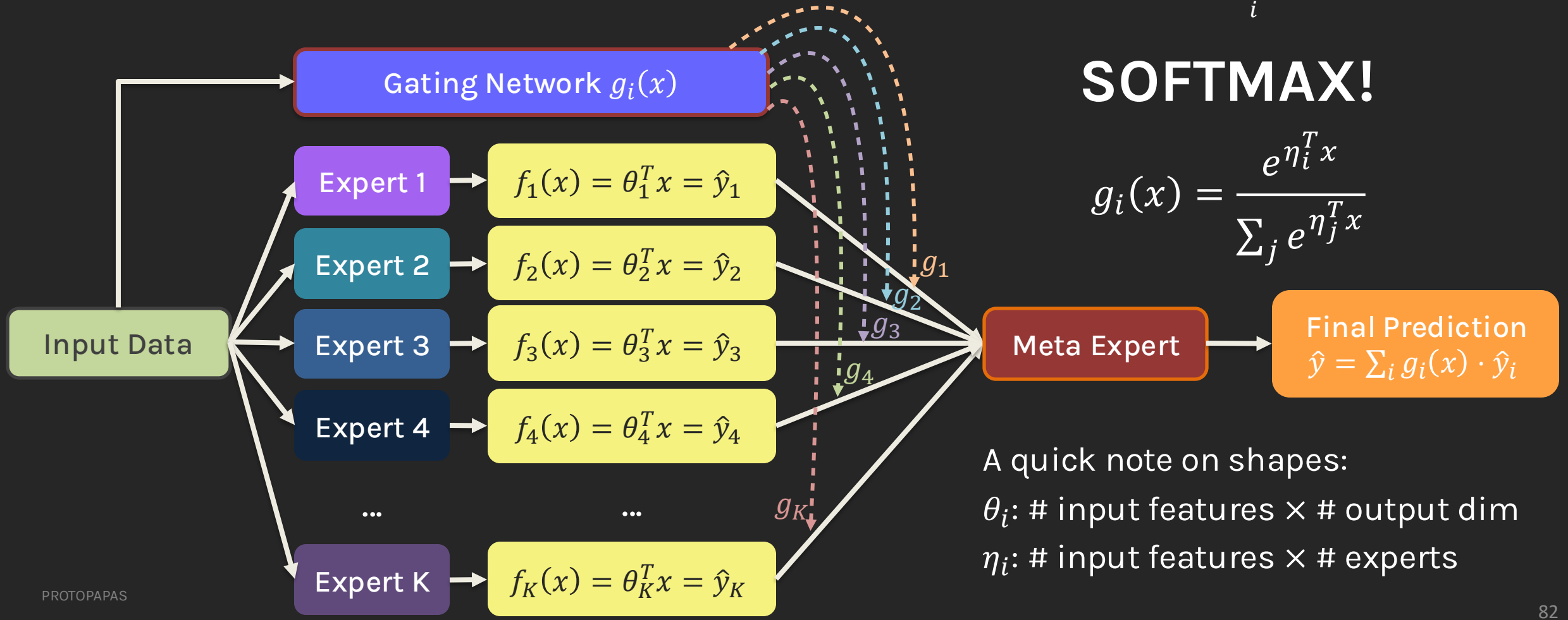
Are there any desired properties of the gating functions?



Mixture of Experts

Are there any functions that satisfy these requirements and help us parameterize the network?

$$0 \leq g_i(x) \leq 1 \qquad \sum_i g_i(x) = 1$$



SOFTMAX!

$$g_i(x) = \frac{e^{\eta_i^T x}}{\sum_j e^{\eta_j^T x}}$$

A quick note on shapes:

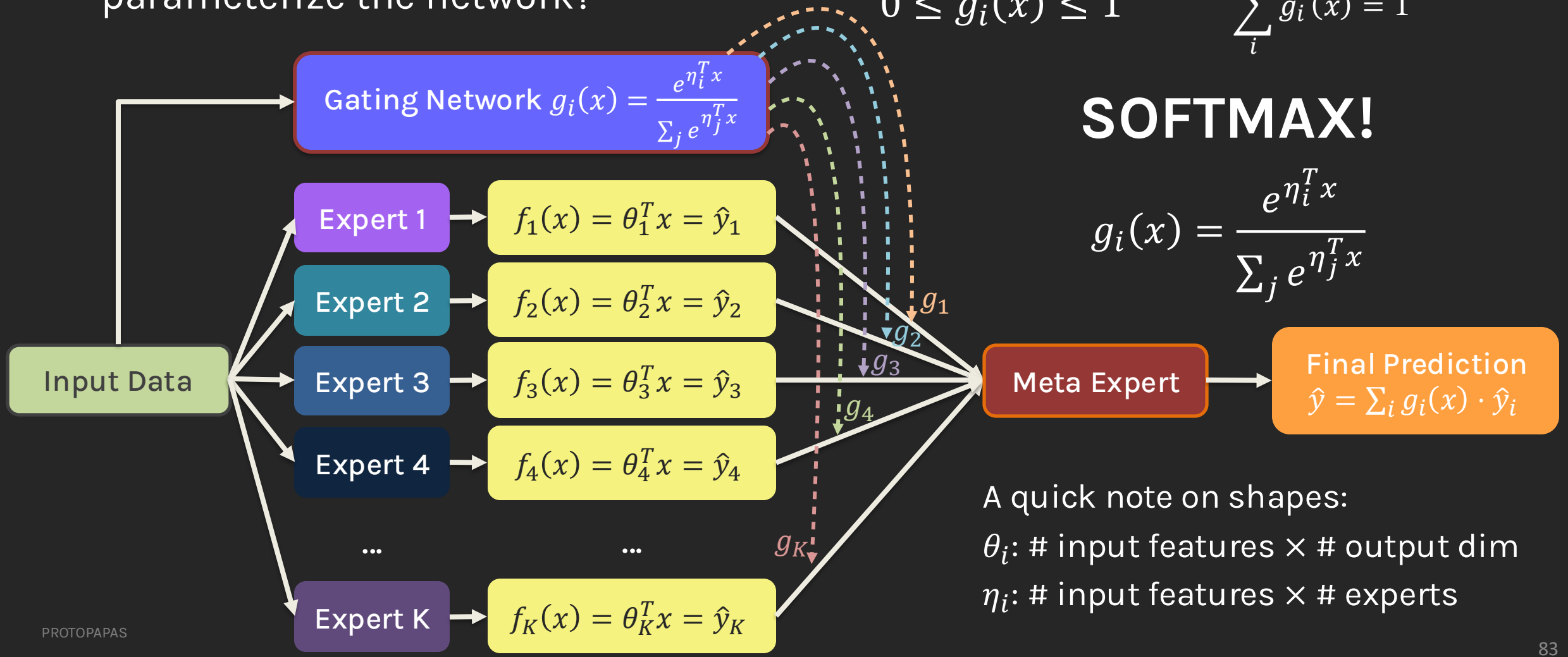
θ_i: # input features × # output dim

η_i: # input features × # experts

Mixture of Experts

Are there any functions that satisfy these requirements and help us parameterize the network?

$$0 \leq g_i(x) \leq 1 \qquad \sum_i g_i(x) = 1$$



SOFTMAX!

$$g_i(x) = \frac{e^{\eta_i^T x}}{\sum_j e^{\eta_j^T x}}$$

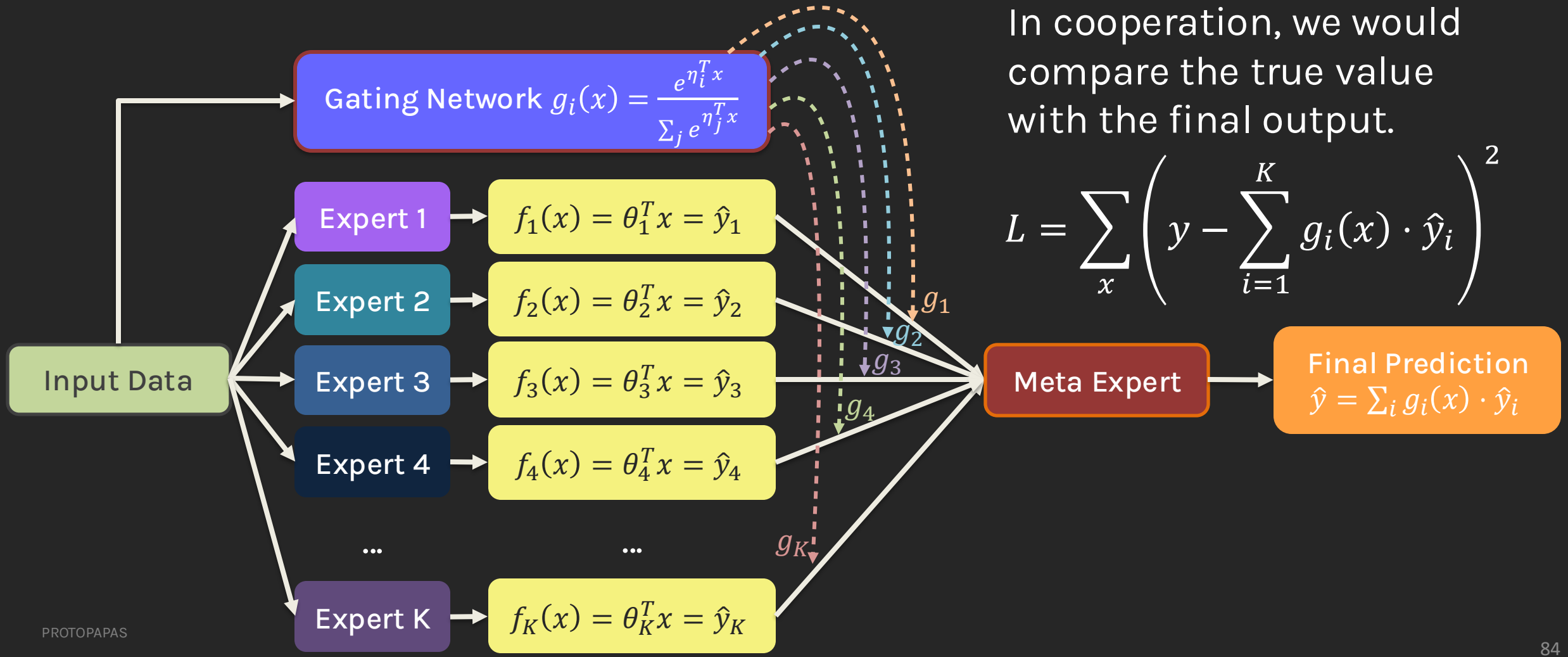
A quick note on shapes:

θ_i: # input features × # output dim

η_i: # input features × # experts

Mixture of Experts

All the parameters are ready. How does the model learn and evaluate?

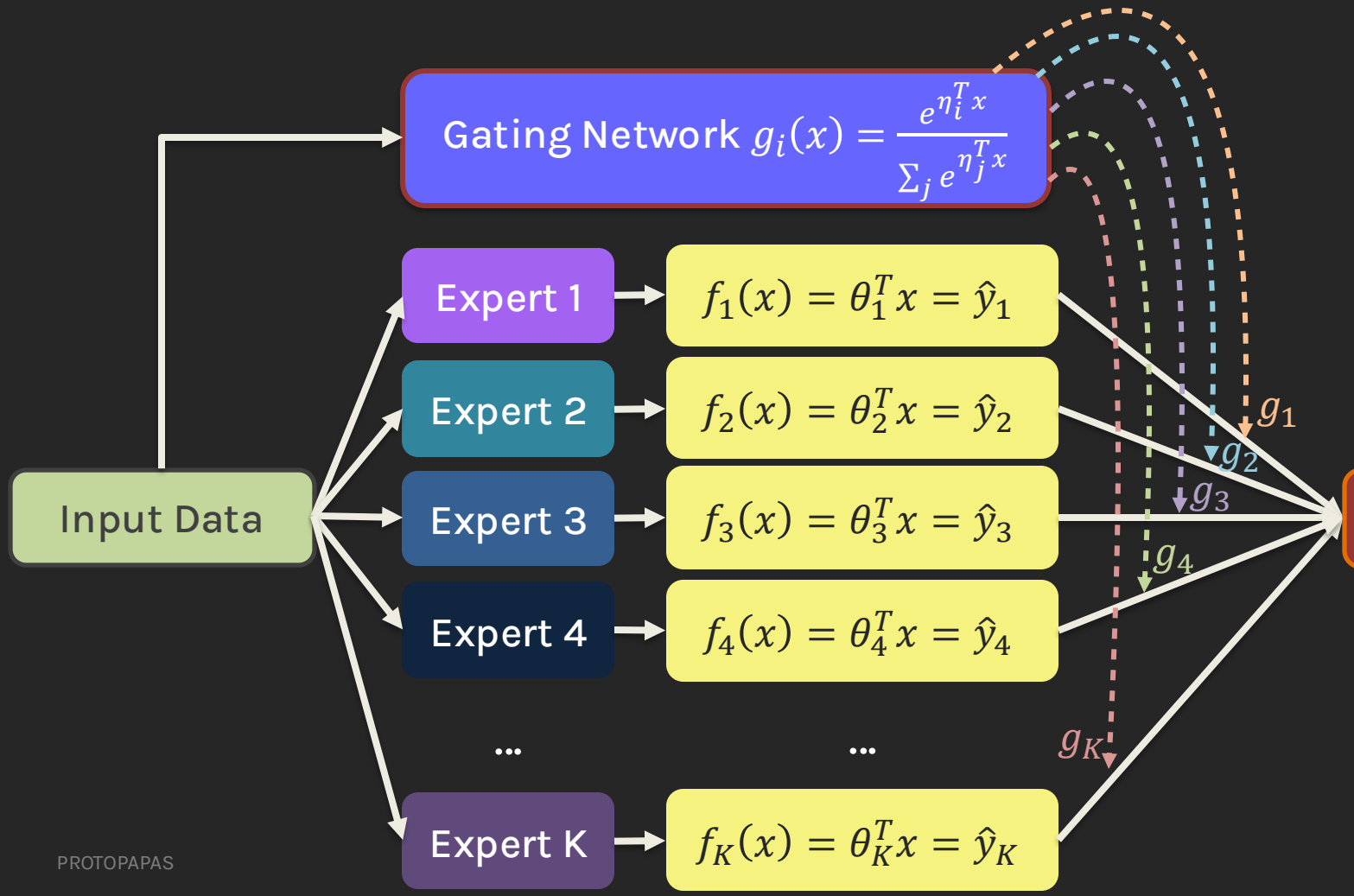


In cooperation, we would compare the true value with the final output.

$$L = \sum_x \left(y - \sum_{i=1}^{K} g_i(x) \cdot \hat{y}_i \right)^2$$

Mixture of Experts

All the parameters are ready. How does the model learn and evaluate?



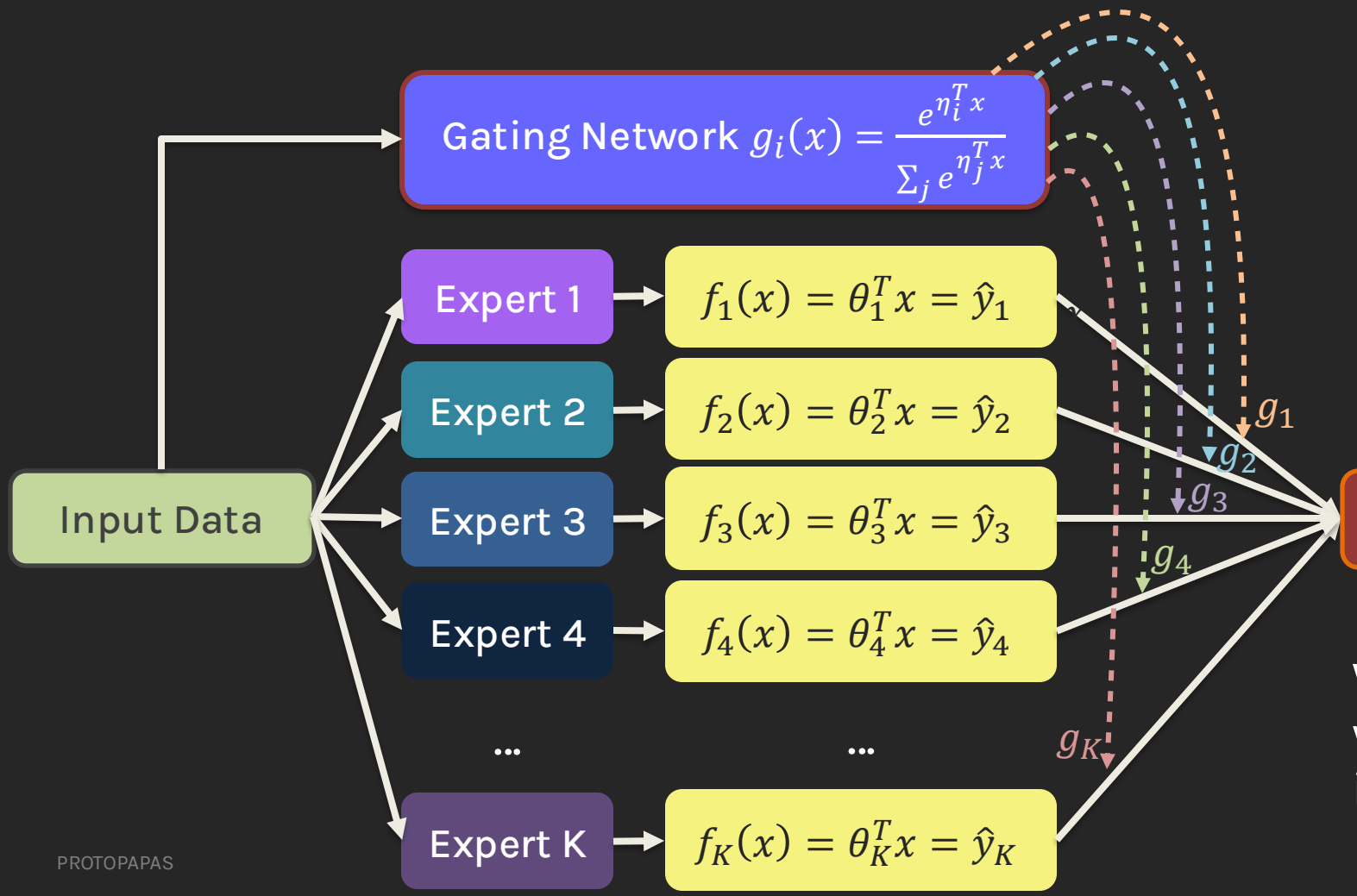
In cooperation, we would compare the true value with the final output.

$$L = \sum_x \left(y - \sum_{i=1}^{K} g_i(x) \cdot \hat{y}_i \right)^2$$

But this goes against our goal of having specialized experts!

Mixture of Experts

All the parameters are ready. How does the model learn and evaluate?



$$L = \sum_x \left(y - \sum_{i=1}^{K} g_i(x) \cdot \hat{y}_i \right)^2$$

$$L = \sum_x \sum_{i=1}^{K} g_i(x) \cdot (y - \hat{y}_i)^2$$

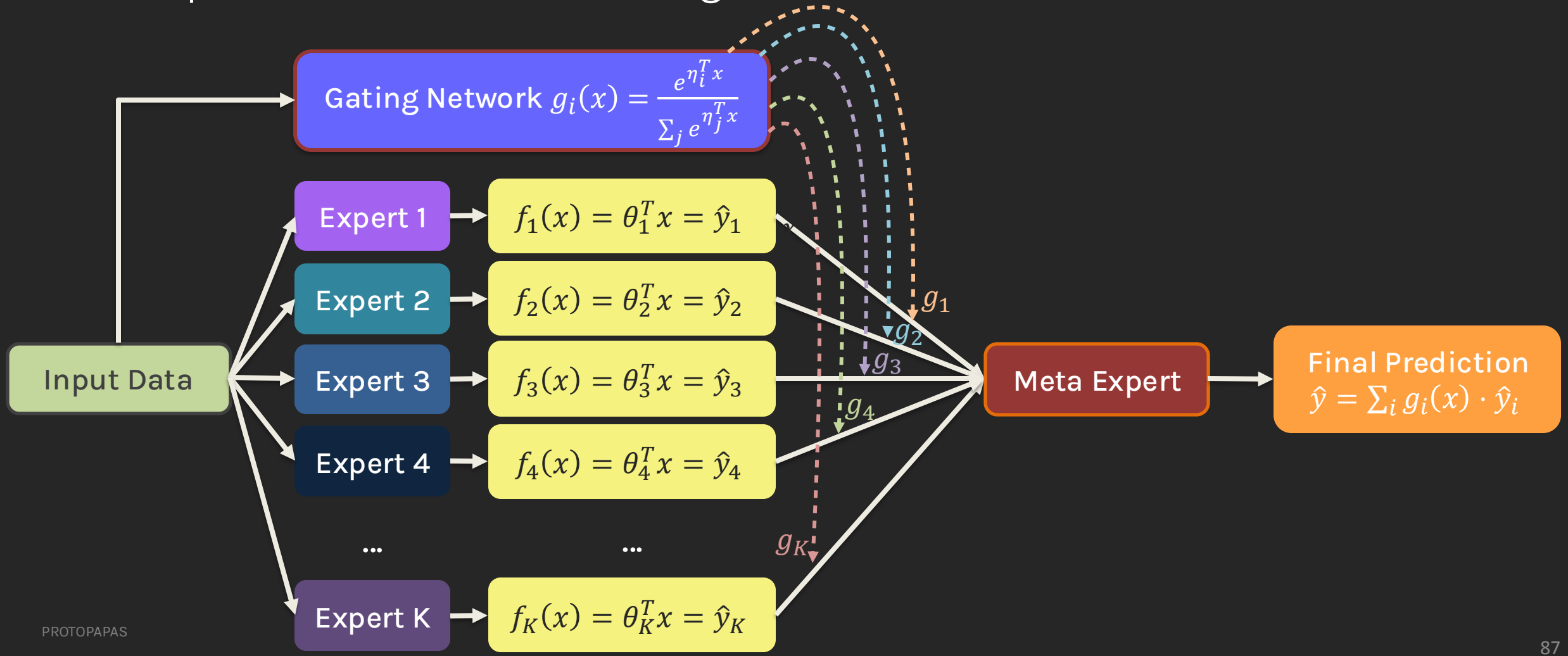
With gating functions, we can give weight to the most important models if we're in the correct domain.

Mixture of Experts

With the loss function, how do we update our model?

Compute the derivatives and do gradient descent!

$$L = \sum_{x} \sum_{i=1}^{K} g_i(x) \cdot (y - \hat{y}_i)^2$$

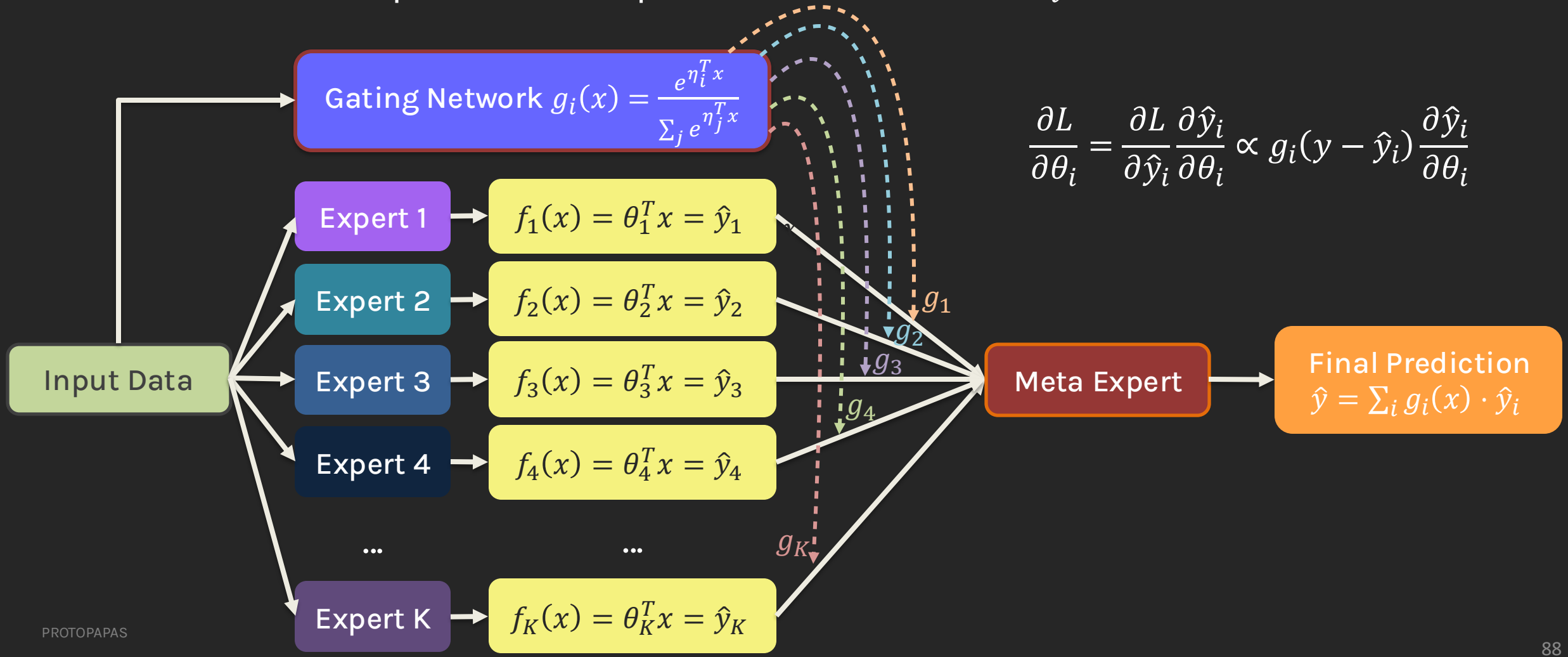


Mixture of Experts

For simplicity, let's focus on only one sample x.

To train each expert, we take partial derivatives to θ_i.

$$L = \sum_{i=1}^{K} g_i(x) \cdot (y - \hat{y}_i)^2$$



$$\frac{\partial L}{\partial \theta_i} = \frac{\partial L}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial \theta_i} \propto g_i(y - \hat{y}_i) \frac{\partial \hat{y}_i}{\partial \theta_i}$$

Mixture of Experts

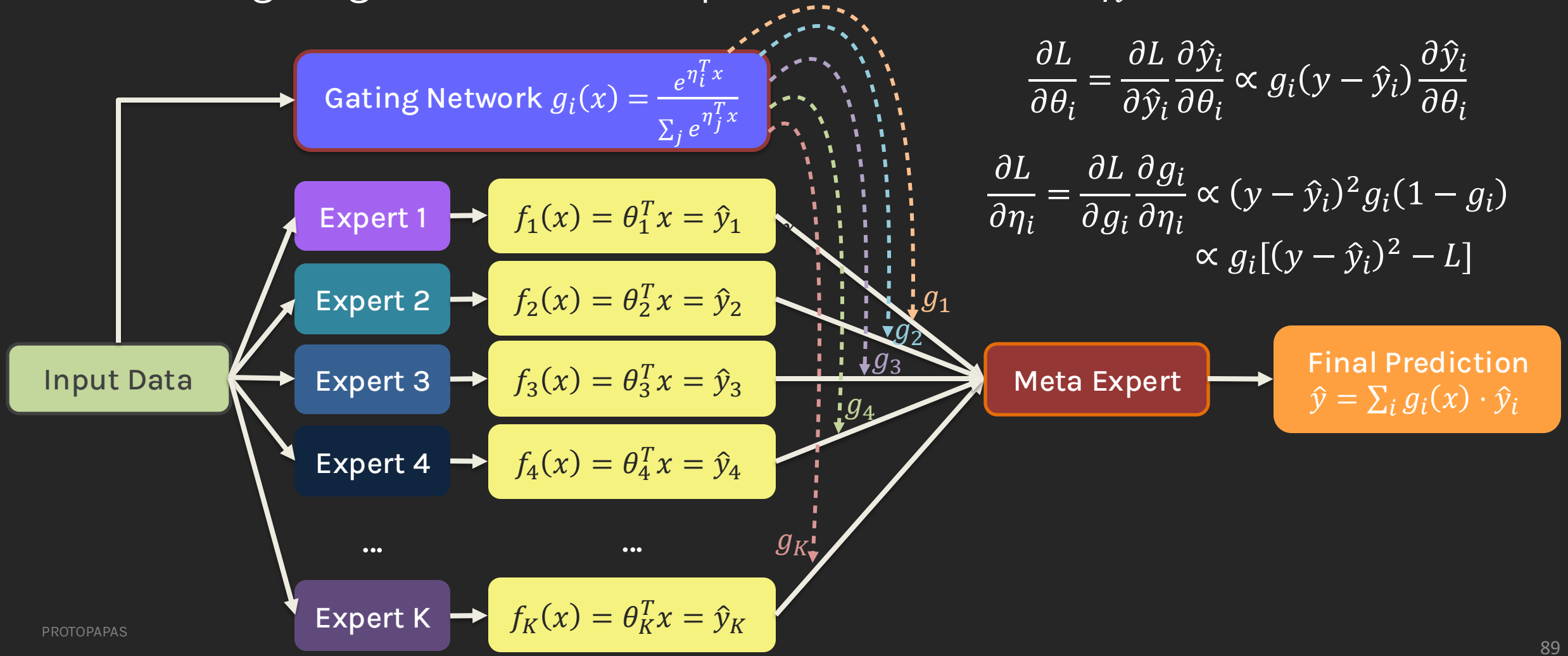
For simplicity, let's focus on only one sample x.

To train gating network, we take partial derivatives to η_i.

$$L = \sum_{i=1}^{K} g_i(x) \cdot (y - \hat{y}_i)^2$$

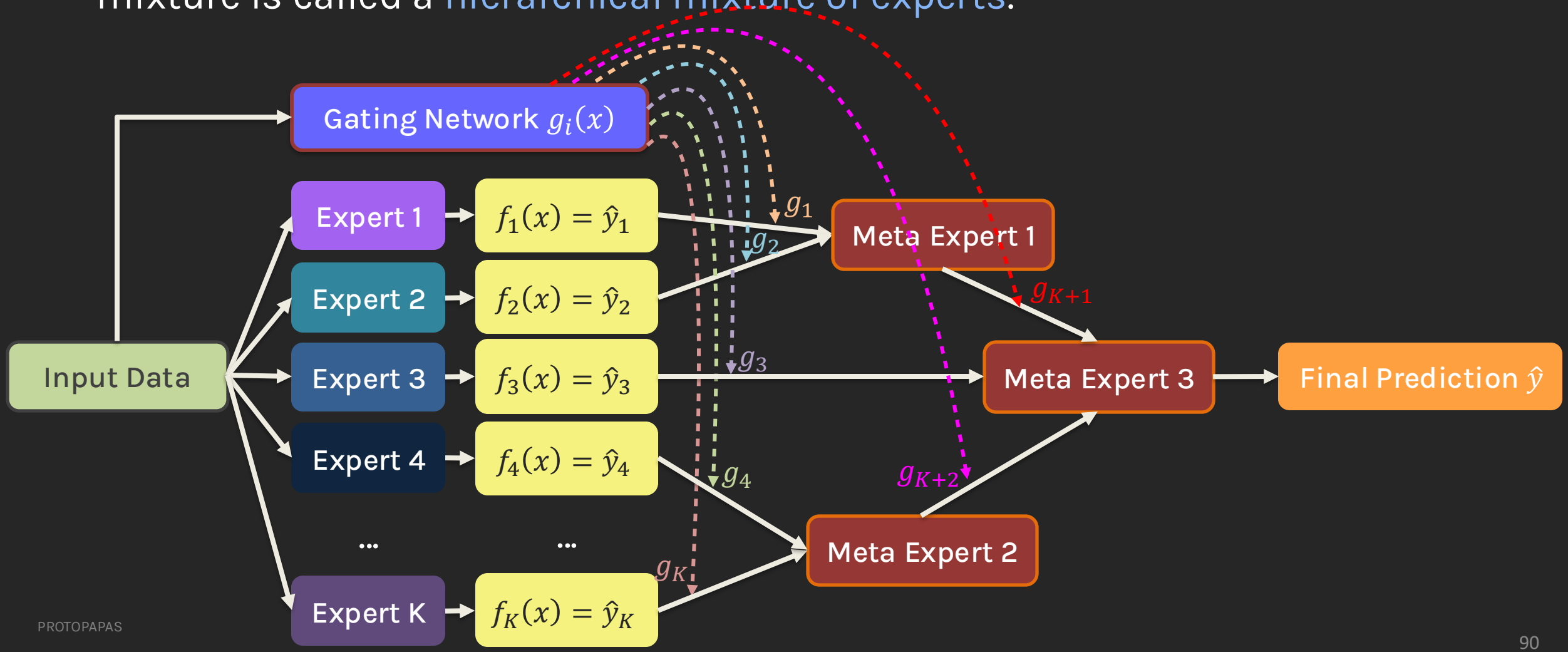
$$\frac{\partial L}{\partial \theta_i} = \frac{\partial L}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial \theta_i} \propto g_i(y - \hat{y}_i)\frac{\partial \hat{y}_i}{\partial \theta_i}$$

$$\frac{\partial L}{\partial \eta_i} = \frac{\partial L}{\partial g_i}\frac{\partial g_i}{\partial \eta_i} \propto (y - \hat{y}_i)^2 g_i(1 - g_i)$$
$$\propto g_i[(y - \hat{y}_i)^2 - L]$$



Hierarchical Mixture of Experts

If the output is conditioned on multiple levels of gating functions, the mixture is called a hierarchical mixture of experts.




GIFWAVE.COM