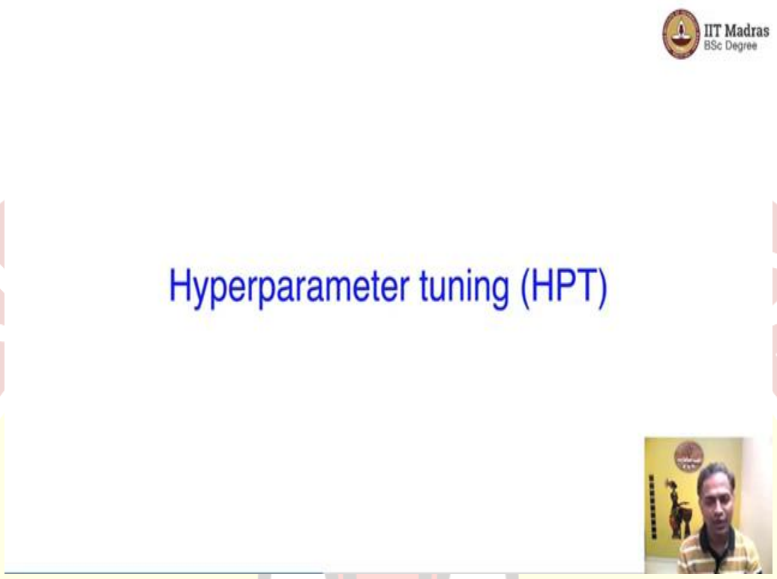
IIT Madras
ONLINE DEGREE

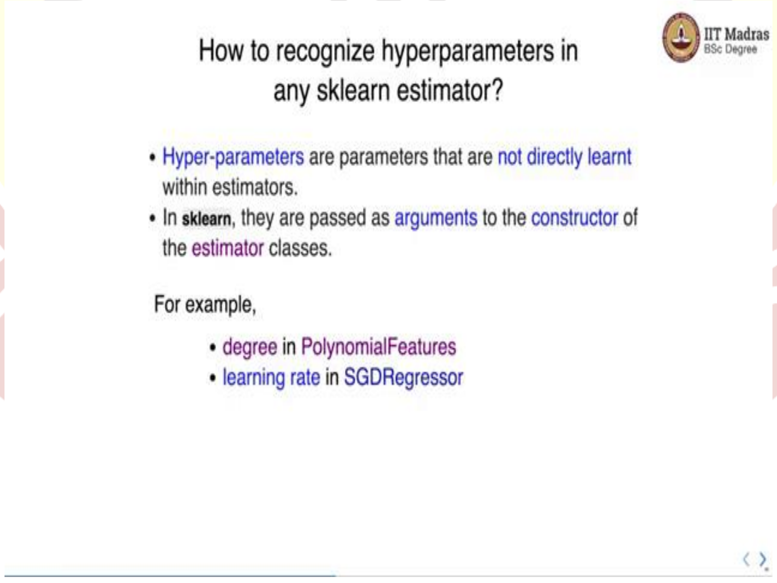
Machine Learning Practice
Indian Institute of Technology, Madras
Hyperparameter tuning

(Refer Slide Time: 0:11)


Hyperparameter tuning (HPT)

Namaste, welcome to the next video of Machine Learning Practice course. In this video, we will discuss how to perform hyperparameter tuning with sklearn.

(Refer Slide Time: 0:25)


How to recognize hyperparameters in any sklearn estimator?

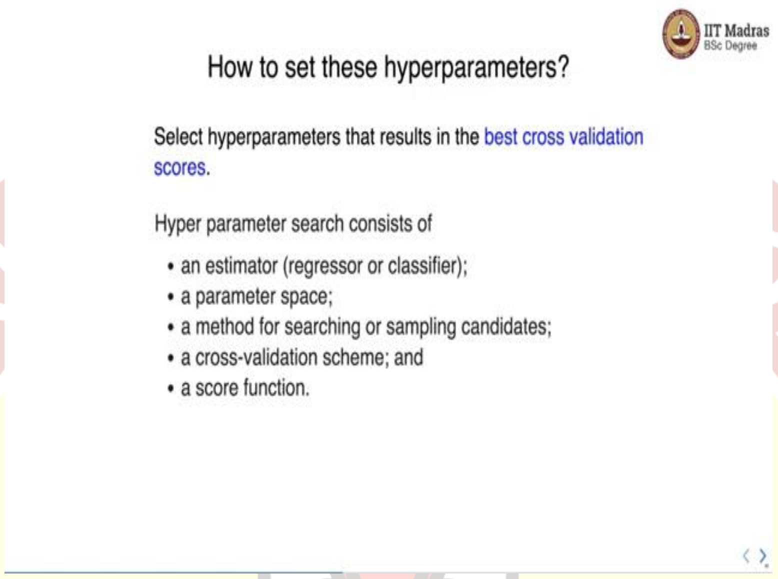
- Hyper-parameters are parameters that are not directly learnt within estimators.
- In **sklearn**, they are passed as arguments to the constructor of the estimator classes.

For example,

- degree in PolynomialFeatures
- learning rate in SGDRegressor

Before performing hyperparameter tuning, we need to recognize the hyper parameters in any sklearn estimator. So, hyper parameters are those parameters that are not directly learned within estimator. In sklearn, they are passed as arguments to the constructor of the estimator classes. For example, degree in polynomial features is a hyperparameter. In the same way, learning rate or penalty in SGD regressor is also examples of hyperparameters.

(Refer Slide Time: 1:03)



How to set these hyperparameters?

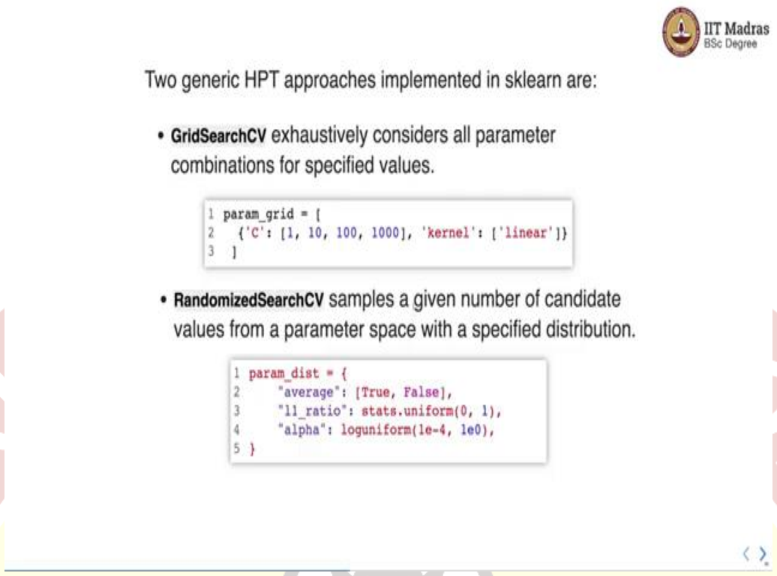
Select hyperparameters that results in the best cross validation scores.

Hyper parameter search consists of

- an estimator (regressor or classifier);
- a parameter space;
- a method for searching or sampling candidates;
- a cross-validation scheme; and
- a score function.

So, how do we set these hyperparameters? So, we normally select hyperparameters that result in the best cross validation scores. Hyperparameter search consists of five different objects. So, we need to have an estimator, which is either regressor, or classifier. We need to have a specification for the parameter space, a method for searching, or sampling candidates, a cross validation scheme and a scoring function. So, we can specify hyperparameter search with these five components.

(Refer Slide Time: 1:42)


Two generic HPT approaches implemented in sklearn are:

- **GridSearchCV** exhaustively considers all parameter combinations for specified values.

```
param_grid = [
  {'C': [1, 10, 100, 1000], 'kernel': ['linear']}
 ]
```

- **RandomizedSearchCV** samples a given number of candidate values from a parameter space with a specified distribution.

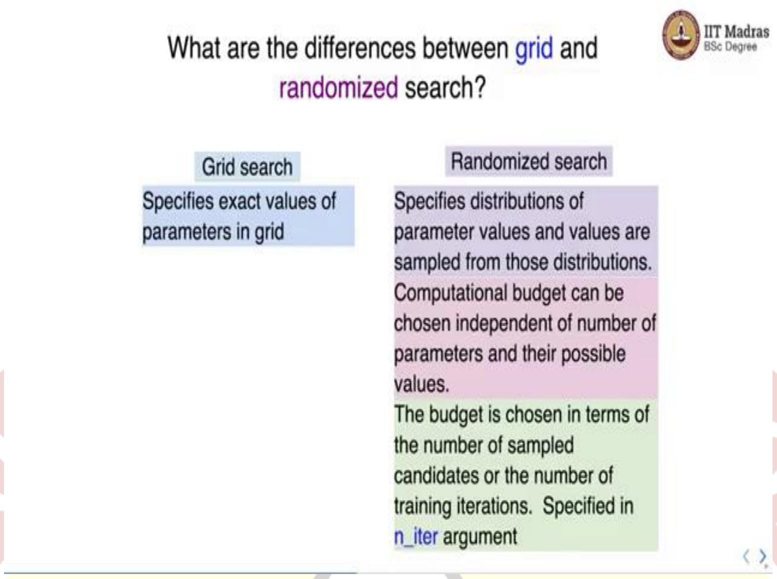
```
param_dist = {
    "average": [True, False],
    "l1_ratio": stats.uniform(0, 1),
    "alpha": loguniform(1e-4, 1e0),
}
```

There are two generic hyperparameter tuning approaches implemented in sklearn. The first one is GridSearchCV, that exhaustively considers all parameter combinations for specified values. For example, here we define a parameter grid for two hyperparameters, one is C and second is Kernel. In case of C, you want to try out values 1, 10, 100, and 1000. Whereas, in case of kernel, we want to try linear kernel. So, this is how we specify our parameter grid. So, this is an example of one parameter grid, where we are having four values specified for C and one value specified for kernel.

The other approach is called RandomizedSearchCV. So, instead of specifying such kind of values, exhaustive such kind of values, RandomizedSearchCV insist samples are given number of candidate values from a parameter space with a specified distribution. So, here instead of specifying the values, we are specifying the distributions from which we want to sample these values. So, here average is set to true, or false and l1_ratio, we have specified that it should be sampled from uniform distribution between 0 and 1. And alpha which is another hyperparameter, should be sampled from log uniform distribution with the specified ranges.

So, that is the difference between RandomizedSearchCV and GridSearchCV, in GridSearchCV, we specify the parameter values that we want to try explicitly, whereas in RandomizedSearchCV is specify the distributions from which we want to sample those values. And we in addition, we also specify how many candidate values we want to sample in the RandomizedSearchCV.

(Refer Slide Time: 3:52)

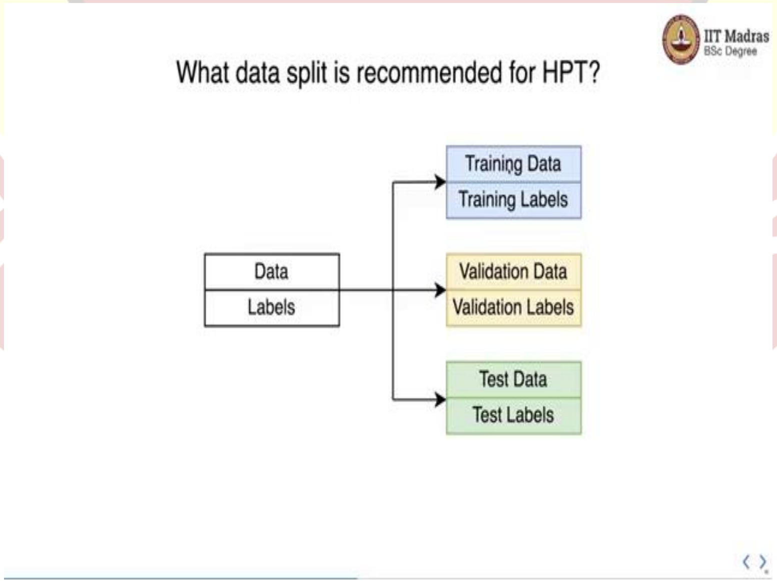


What are the differences between grid and randomized search?

Grid search	Randomized search
Specifies exact values of parameters in grid	Specifies distributions of parameter values and values are sampled from those distributions.
	Computational budget can be chosen independent of number of parameters and their possible values.
	The budget is chosen in terms of the number of sampled candidates or the number of training iterations. Specified in n_iter argument

So, let us study the differences between grid and randomized search. Grid search specifies exact values of the parameters in grid. Randomized search on the other hand specifies distribution of parameter values and values are sampled from those distribution. In randomized search, the computational budget can be chosen independent of number of parameters and their possible values. The budget it chosen in terms of the number of sampled candidates, or number of training iteration, it is specified in n_iter argument of randomized search CV.

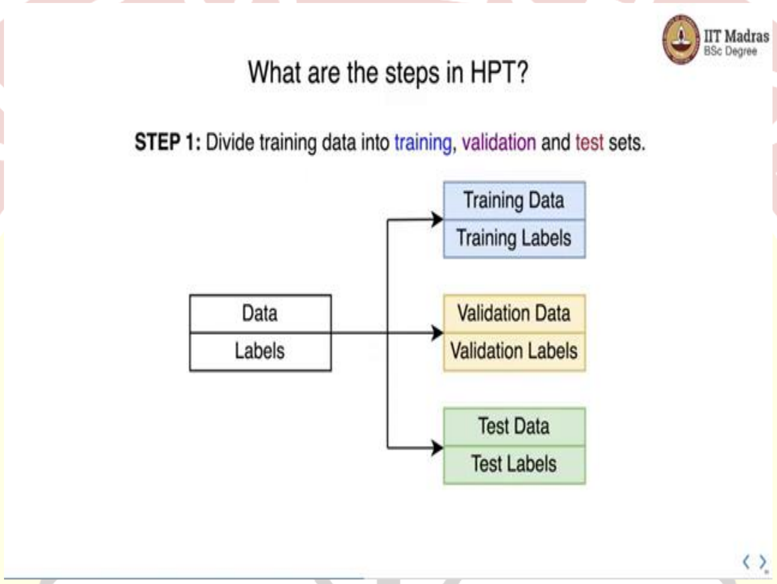
(Refer Slide Time: 4:34)



What data split is recommended for HPT?

Let us study what kind of data splits are recommended for hyperparameter tuning. So, we have the training data at our disposal, we divide it into three sets. The first is the training set, second is validation set and third is test set. So, each of these set consists of two parts, one is the feature matrix, which is denoted by data and labels, which could be label vector, or label metrics depending on whether you are solving a single, or output, or multi output regression problem. So, test set is normally set aside and not at all used during hyperparameter tuning stage, we only use training and validations validation sets for hyperparameter tuning.

(Refer Slide Time: 5:27)

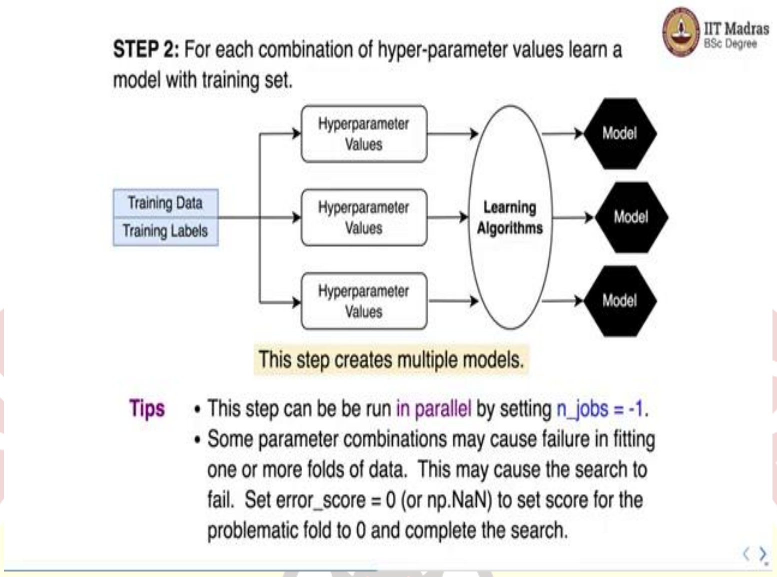


What are the steps in HPT?

STEP 1: Divide training data into training, validation and test sets.

So, as the first step, we divide training data into training, validation, and test sets.

(Refer Slide Time: 5:34)



STEP 2: For each combination of hyper-parameter values learn a model with training set.

Training Data
Training Labels
Hyperparameter Values
Hyperparameter Values
Hyperparameter Values
Learning Algorithms
Model
Model
Model

This step creates multiple models.

Tips
- This step can be run in parallel by setting n_jobs = -1.
- Some parameter combinations may cause failure in fitting one or more folds of data. This may cause the search to fail. Set error_score = 0 (or np.NaN) to set score for the problematic fold to 0 and complete the search.

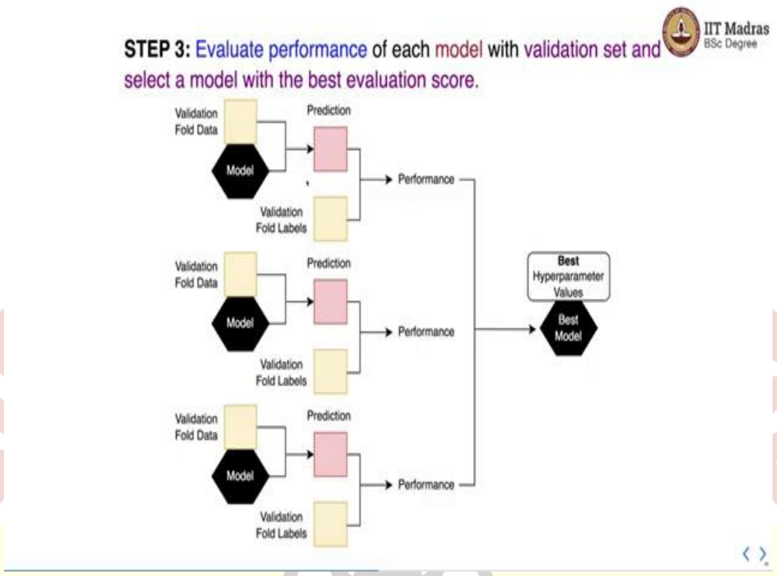
In the second step, for each combination of hyper parameter values, we learn a model with training set. So, we have training set, and we have different values of the hyperparameter. These values may be specified by a grid, or via randomized search CV. For each value of hyperparameter, we train for each possible values of hyperparameter set, we train a model.

So, depending on how many number of configurations of hyperparameter we want to take, we train those many models. So, as I said, this step creates multiple models. So, we can run since all these models are being trained in parallel, we can run the step in parallel. For that we have to set n_jobs parameter to -1.

Here there is another, caution that we need to take, there are some parameter combinations that may cause failure in fitting one, or more folds of data. This may cause the entire search to fail. In such cases, set error_score to 0, or np.NaN. This helps us to set score for problematic folds to 0 and did not complete a search.

So, there are these two things that you should keep in mind. One is that we can run the jobs in parallel. And second, whenever error occurs in a particular hyperparameter search, what should be done with error of that particular fold? So, we can set it to 0, or np.NaN. And you know that sets the error to 0 and we are able to complete the search.

(Refer Slide Time: 7:32)

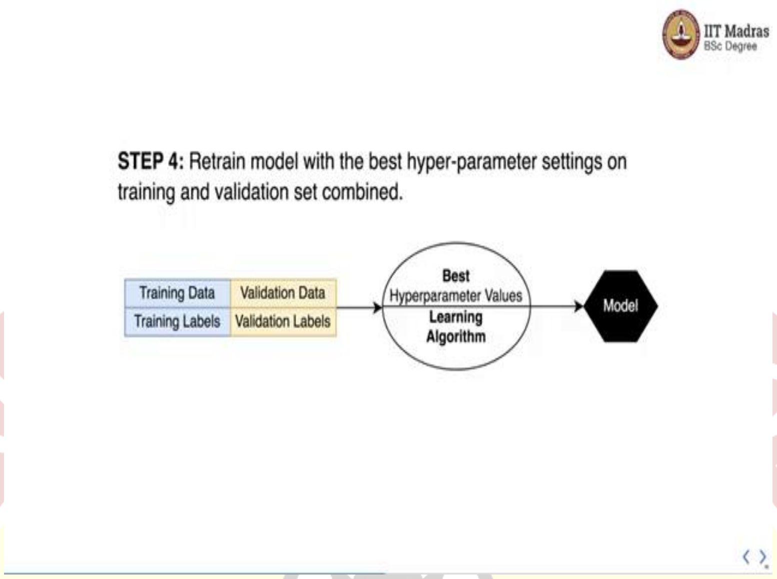


STEP 3: Evaluate performance of each model with validation set and select a model with the best evaluation score.

In the step 3, we evaluate performance of each model with validation set, and select a model with the best evaluation score. So, these are the models that we learned in the in the previous step. And now what we do is we use the validation data. And we use the validation feature matrix specifically to get predictions, we compare those predictions with the actual predictions of the validation fold, and come up with and calculate a performance measure.

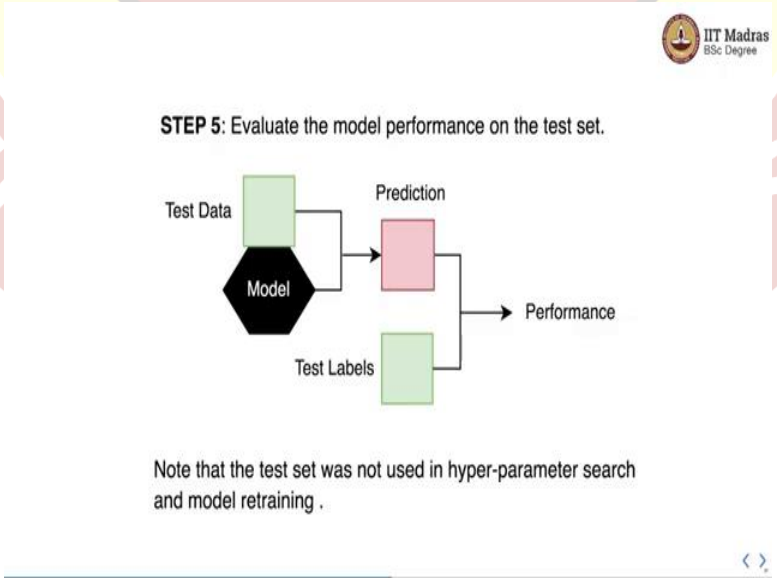
We repeat this process for each model that was learned in the previous step. And we select the base model that is based on the performance score. So, the corresponding hyperparameters, hyperparameter values are the best hyperparameter values for us, because those hyperparameter values result into the model that give us the best performance score on the validation set.

(Refer Slide Time: 8:33)


STEP 4: Retrain model with the best hyper-parameter settings on training and validation set combined.

In step four, we retrain the model with best hyperparameter settings, on training and validation set combined. Here we combine training and validation set and form a bigger training set and use that for training the learning algorithm. By training this algorithm, we set the values of the hyperparameter to the best hyperparameter values that were discovered in the previous step. So, we get a final model, which has the hyperparameter values set to the best hyperparameter discovered to the hyperparameter tuning process.

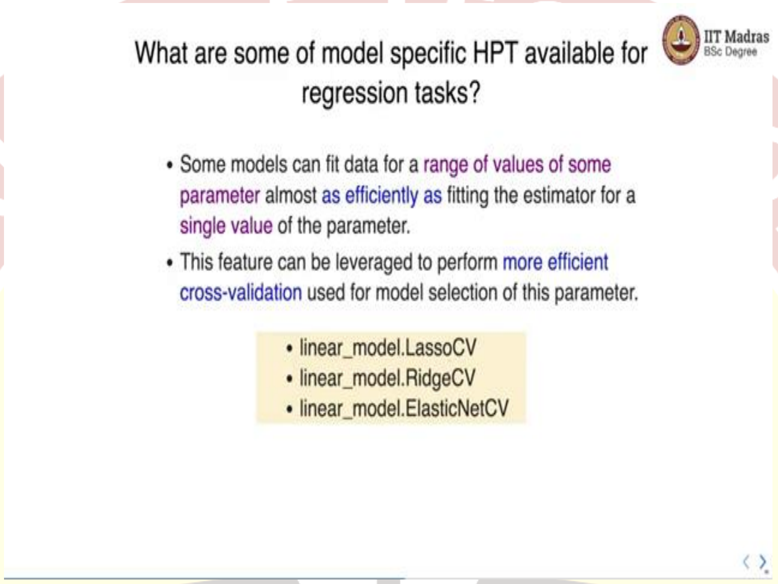
(Refer Slide Time: 9:06)


STEP 5: Evaluate the model performance on the test set.

Note that the test set was not used in hyper-parameter search and model retraining .

Finally, we evaluate the model performance on the test set. So, in order to report the performance of the model, we always do performance evaluation on a test set. And remember we have not used test set during the hyperparameter tuning process. And that is why this performance measure is likely to give us true performance measure on the unseen data.

(Refer Slide Time: 9:36)



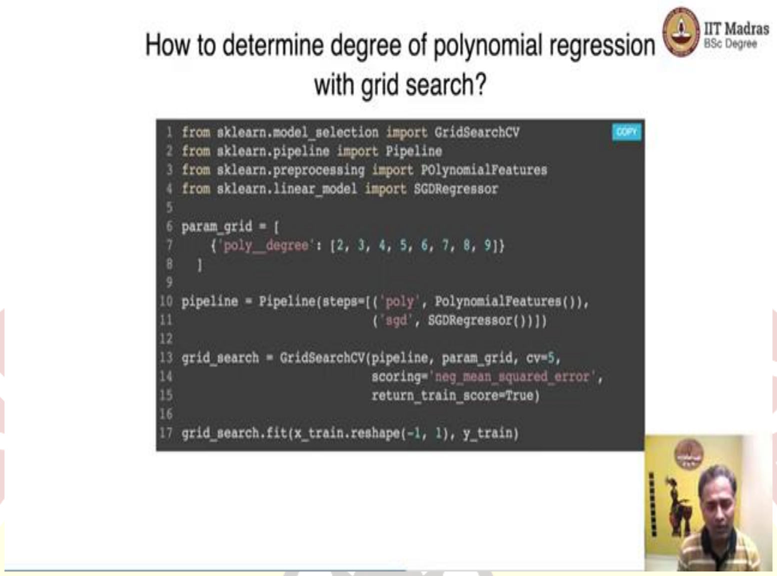
What are some of model specific HPT available for regression tasks?

- Some models can fit data for a range of values of some parameter almost as efficiently as fitting the estimator for a single value of the parameter.
- This feature can be leveraged to perform more efficient cross-validation used for model selection of this parameter.

- linear_model.LassoCV
- linear_model.RidgeCV
- linear_model.ElasticNetCV

So, apart from these two methods, which is grid search CV and randomized search CV, there are some model specific hyperparameter tuning that is available for regression task. So, there are some models that can fit data for a range of values of some parameter almost as efficiently as fitting the estimator for a single value of that parameter. This feature can be leveraged to perform more efficient cross validation used for model selection of those parameters. So, there are three such models which are LassoCV, RidgeCV and ElasticNetCV, that help us to come up with the ideal, or the best value of the regularization rate used in Lasso, Ridge, as well as in elastic net.

(Refer Slide Time: 10:27)



How to determine degree of polynomial regression with grid search?

```
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import POlynomialFeatures
from sklearn.linear_model import SGDRegressor

param_grid = [
    {'poly__degree': [2, 3, 4, 5, 6, 7, 8, 9]}
  ]

pipeline = Pipeline(steps=[('poly', PolynomialFeatures()),
                           ('sgd', SGDRegressor())])

grid_search = GridSearchCV(pipeline, param_grid, cv=5,
                           scoring='neg_mean_squared_error',
                           return_train_score=True)

grid_search.fit(x_train.reshape(-1, 1), y_train)
```

Let us take an example of how to determine degree of polynomial regression with grid search. So, here we import the grid search from model selection module, we have a pipeline class, then we have a preprocessing with polynomial features this O should be in small case, we will make that correction in the in the slides.

So, we import polynomial features as well as SGD regressor. First, we specify the parameter grid, where we set the value of the degree parameter 2, 3, 4, 5, 6, 7, 8, and 9, there are these eight values that we select that we set to this degree. So, you want to try these eight values in polynomial features. And remember this particular way of accessing the parameter is what we studied in the pipeline.

So, in the pipeline object we have denoted the polynomial feature with the key poly, so we are accessing the degree parameter of this polynomial features with the name which is poly double underscore and name of the parameter which is degree. And we are setting these degrees to eight different values. And we will try those eight different values and find out the best possible value of the degree through the grid search. We set up the pipeline with two steps one is the polynomial features and second is SGD regressor.

Finally, we call the GridSearchCV, on this particular pipeline object with the specified parameter grid setting, number of cross validation is equal to 5. And we also specify the performance measure, which is negative mean squared error. And we want, we want this GridSearchCV to also return the training scores, that is why we are setting return_train_score to True.

And finally, we call the fit method on the training set. So,we are specifying just like fit of any estimator we are specifying the feature matrix as well as the labels to the fit procedure and this will perform the grid search and get us the best possible hyperparameter in this case we will get the suitable value for the degree hyperparameter of the polynomial features. So, this was a short discussion about how we can use hyperparameter tuning for polynomial regression.

So, in this video we studied how to perform hyperparameter tuning. And there are two ways for performing hyperparameter tuning either using grid search, or Randomized searchCV. And there is a third way, where there are certain methods that provides efficient way of performing hyperparameter tuning. And those methods correspond to the regularization methods which is Lasso CV, Ridge CV and elastic net CV.

Then we looked at how to perform data splits in hyperparameter tuning, and how training and validation folds are used for coming up with the best possible values of the hyperparameter. And then we use test fold for reporting the final performance of the model. So,we will use these techniques that we studied in this video in the colab for more extensive demonstration.