

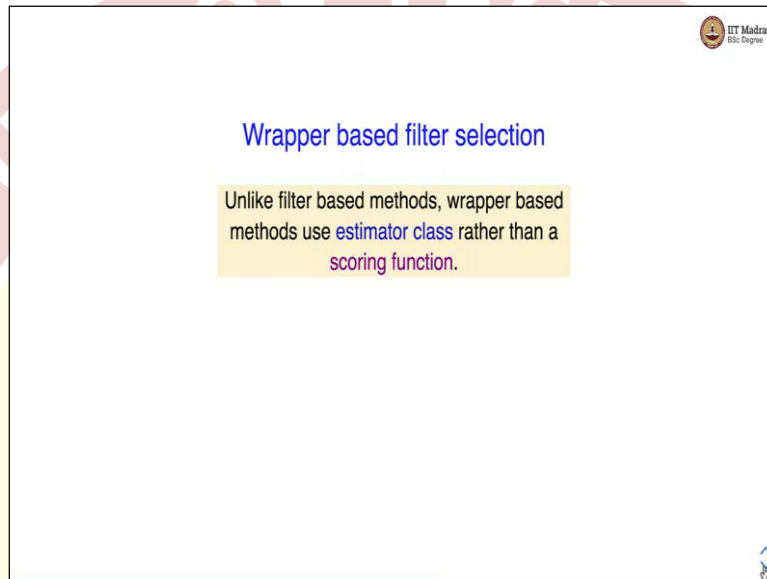
IIT Madras

ONLINE DEGREE

**Machine Learning Practice
Online Degree Programme
B. Sc in Programming and Data Science
Diploma Level
Dr. Ashish Tendulkar
Indian Institute of Technology – Madras**

Wrapper Based Filter Selection

(Refer Slide Time: 00:12)



Namaste welcome to the next video of the machine learning practice course. In this video, we will discuss wrapper based feature selection methods. In the last video, we have discussed filter-based methods. They used a scoring function in order to score different features. In the wrapper based method instead of using a scoring function, we use an estimator class to decide the importance of features. Let us look at wrapper based methods one by one.

(Refer Slide Time: 00:45)

Recursive Feature Elimination (RFE)

- Uses an **estimator** to **recursively remove features**.
 - Initially fits an estimator on all features.
 - Obtains **feature importance** from the estimator and **removes the least important feature**.
 - **Repeats the process** by removing features one by one, **until desired number of features** are obtained.
- Use **RFECV** if we do not want to specify the desired number of features in **RFE**.
 - It performs **RFE** in a **cross-validation loop** to find the **optimal number of features**.

The first wrapper based method is recursive feature elimination or RFE. RFE uses an estimator to recursively remove features. It initially fits an estimator on all features then it obtains feature importance from the estimator and removes the least important feature. It repeats the process by removing features one by one until the desired number of features is obtained. Here we have to specify the desired number of features if we do not want to do that we can use another method called RFECV which is recursive feature illumination with cross-validation.

So, what RFECV does is perform RFE or recursive feature illumination in a cross-validation loop to find out the optimal number of features thus we are freed from specifying the desired number of features as in RFE.

(Refer Slide Time: 01:54)

SelectFromModel

Selects **desired number of important features** (as specified with **max_features** parameter) above certain **threshold of feature importance** as obtained from the trained estimator.

- The feature importance is obtained via **coef_**, **feature_importances_** or an **importance_getter** callable from the trained estimator
- The feature importance threshold can be specified either **numerically** or **through string argument** based on **built-in heuristics** such as **'mean'**, **'median'** and float multiples of these like **'0.1*mean'**.

Let's look at a concrete example of SelectFromModel

SelectFromModel first trains an estimator on all features. Then it selects a desired number of features based on feature importance. The maximum number of features that we want to select is specified for parameter max_features. The feature importance is obtained via coef_ feature_importance_ member variables of the estimator class.

We can also use importance_getter callable from the estimator class. The feature importance threshold can be specified either numerically or through string argument based on building built-in heuristics such as mean, median and float multiples of these heuristics like 0.1 into mean. So, here as like other wrapper based methods we first train an estimator on all features, and then we select the desired number of features based on the feature importance threshold.

We also specify the maximum features that we need in the parameter of the SelectFromModel feature selector. Let us look at a concrete example of SelectFromModel. **(Refer Slide Time: 03:36)**



```
clf = LinearSVC(C=0.01, penalty='l1', dual=False)
clf = clf.fit(X, y)
clf.coef_

model = SelectFromModel(clf, prefit=True)
X_new = model.transform(X)
```

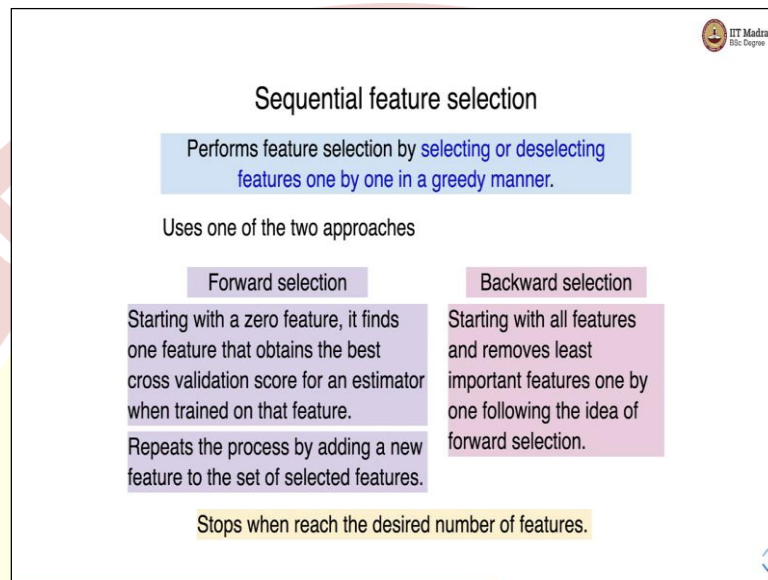
- Here we use a linear support vector classifier to get coefficients of features for `SelectFromModel` transformer.
- It ends up selecting features with non-zero weights or coefficients.

So, here we use a linear support vector classifier to get coefficients of features from the SelectFromModel transformer. Here we first defined an estimator which is linear svc and in SelectFromModel we use this particular estimator clf and then we apply the transform method on the original feature set to obtain a new feature set based on the features selected from the SelectFromModel feature selector.

So, this particular code ends up selecting features with non-zero weights or coefficients. So, here you should also notice that we have used penalty l1. So, this is an l1 regularized

l_1 regularized linear support vector classifier. And hence we select features with non-zero weights or coefficients. And we will study this regularization in the subsequent weeks and you will see that the l_1 regularizer essentially gets us non-zero weights only to features that are useful. All other weights get which all other weights are 0 or are taken to 0 by the l_1 regularizer.

(Refer Slide Time: 05:18)



Let us study sequential feature selection as the next wrapper-based feature selection method. It performs feature selection by selecting or deselecting one feature at a time in a greedy manner. There are two approaches one is the forward selection and the second is backward selection. Forward selection starts with zero feature and then go on adding a feature one by one until the desired number of features are obtained.

Whereas in the backward selection we start with all features and then go on deselecting or reducing features one by one until the desired number of features are obtained. Let us look at how forward and backward selection works. In forwarding selection, we start with zero feature and then find one feature that obtains the best cross-validation score for an estimator when we train that estimator on that feature.

And then we add that one feature to the set. So, now after the first iteration, we have one feature. We repeat this process by adding a new feature in every iteration. So, in the second iteration what will happen is we will add we already have one feature we will add another feature and we again train the estimator with those two features, and then we select the

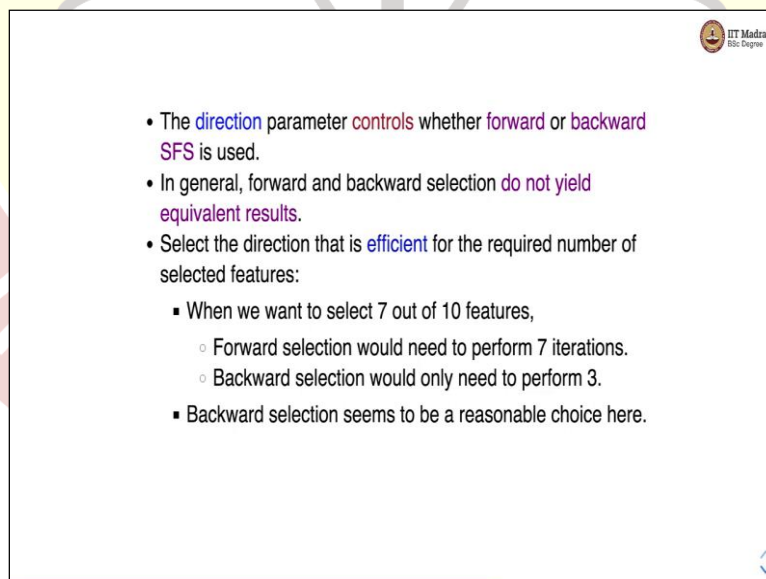
new feature which obtains the best cross-validation performance when selected along with the earlier selected feature.

So, then we now have two features the in the third iteration we will try to add a third feature that obtains again the best cross-validation score for an estimator when trained with the three features and this is how we repeat the process until we get the desired number of features. Whereas in backward selection what we do is we start with all features and then we remove the least important features one by one.

Here also what we do is we start with all features we train an estimator we get the feature importance score and when we remove the least important feature in. Let us say in the first step. So, then we have $m - 1$ feature. In the second step again we will repeat the process we will train the estimator we will get the feature importance and then we will remove the least important feature then we have $m - 2$ features after the second iteration.

We keep repeating this process until we find the desired number of features. We stop both of these methods when we reach the desired number of features.

(Refer Slide Time: 08:06)



The slide contains the following text:

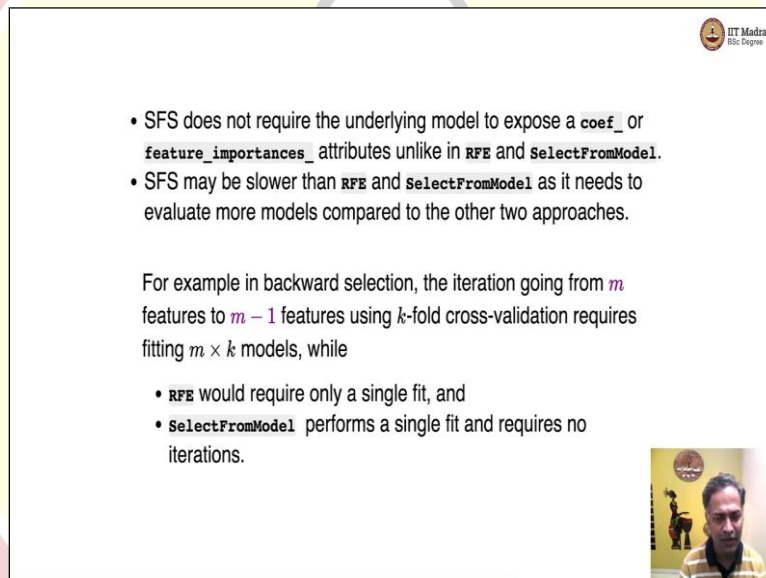
- The **direction** parameter **controls** whether **forward** or **backward SFS** is used.
- In general, forward and backward selection **do not yield equivalent results**.
- Select the direction that is **efficient** for the required number of selected features:
 - When we want to select 7 out of 10 features,
 - Forward selection would need to perform 7 iterations.
 - Backward selection would only need to perform 3.
 - Backward selection seems to be a reasonable choice here.

So, now forward and backward are the direction and these directions control whether forward or backward feature selection is used in general forward and backward selection do not yield equivalent results. So, how do we select a direction we have to select the direction that is efficient for the required number of selected features. So, let us say we have 10 features and we want to select 7 features out of these 10 features.

So, now if we apply if we understand forward and the backward selection and apply the strategy the forward selection would need to perform 7 iterations because it starts with zero features and then goes on adding features one by one. So, there will be seven iterations that will be needed to add seven features whereas backward selection which will start with 10 features after the first iteration it will have 9 features then it will have eight features and then it will have seven features.

So, it needs to perform only three iterations. So, in this case, backward selection seems to be a reasonable choice. So, we will have to select forward or backward based on what is efficient based on the required number of selected features.

(Refer Slide Time: 09:30)



• SFS does not require the underlying model to expose a `coef_` or `feature_importances_` attributes unlike in `RFE` and `SelectFromModel`.

• SFS may be slower than `RFE` and `SelectFromModel` as it needs to evaluate more models compared to the other two approaches.

For example in backward selection, the iteration going from m features to $m - 1$ features using k -fold cross-validation requires fitting $m \times k$ models, while

- `RFE` would require only a single fit, and
- `SelectFromModel` performs a single fit and requires no iterations.

Sequential feature selection does not require the underlying model to expose a coefficient of feature importance attributes like recursive feature elimination and `SelectFromModel`. The sequential feature selection may be slower than recursive feature elimination and `SelectFromModel` add as it needs to evaluate more models compared to the other two approaches.

For example, in backward selection, the iteration going from m features to $m - 1$ feature using k -fold cross validation would require us to fit m cross k models in sequential feature selection. While in recursive feature elimination we would require only a single fit and in `SelectFromModel` we need to perform a single fit and it requires no iterations. So,

sequential feature selection needs more models to evaluate in comparison to recursive feature elimination and SelectFromModel.

So, now we have a good idea about feature selection methods we looked at the filter based feature selection and wrapper based feature selection and depending on your need you can use either feature based or wrapper based feature selection methods.

