**WEEK 5: MODEL EVALUATION AND REFINEMENT**

**MODEL EVALUATION**

* In-sample evaluation tells us how well our model will fit the data used to train it
* Problem: it does not tell us how well the trained model can be used to predict new data
* Solution:
  + In-sample data or training data
  + Out-of-sample evaluation or test set

**TRAINING/TESTING SETS**

* Split dataset into:
* Training set (70%)
* Testing set (30%)
* Build and train the model with a training set
* Use testing set to assess the performance of a predictive model

When we have completed testing our model we should use all the data to train the model to get the best performance

**FUNCTION train\_test\_split()**

Split data into random train and test subsets

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x\_data, y\_data, test\_size = 0.3, random\_state=0)

With:

x\_data: features or independent variables

y\_data: dataset target: df[‘price’]

x\_train, y\_train: parts of available data as training set

x\_test, y\_test: parts of available data as testing set

test\_size: percentage of the data for testing (here 30%)

random\_state: number generator used for random sampling

**GENERALIZATION PERFORMANCE**

* Generalization error is a measure of how well our data does at predicting previously unseen data
* The error we obtain using our testing data is an approximation of this error

Chart, line chart

Description automatically generated

Actual value: red

Predicted value: blue

**CROSS VALIDATION**

* Chart

  Description automatically generatedMost common out-of-sample evaluation metrics
* More effective use of data (each observation is used for both training and testing)

**FUNCTION cross\_val\_score()**

From sklearn.model\_selection import cross\_val\_score

Scores = cross\_val\_score(lr, x\_data, y\_data, cv=3)

With:

Lr: the type of model we are using to do the cross-validation (lr = linear regression)

X\_data: predicted variable data

Y\_data: target variable data

Cv: number of partitions

The function returns an array of scores. Thus, we use the below function to average the result and get the r-squared

Np.mean(scores)

**FUNCTION cross\_val\_predict()**

* It returns the prediction that was obtained for each element when it was in the test set
* Has a similar interface to cross\_val\_score()

From sklearn.model\_selection import cross\_val\_predict

Yhat = cross\_val\_predict(lr2e, x\_data, y\_data, cv=3)

**OVERFITTING, UNDERFITTING, AND MODEL SELECTION**

**Graphical user interface

Description automatically generated with low confidenceMODEL SELECTION:** determine the order of the polynomial to provide the best estimate of the function y(x)

Underfitting

Chart, line chart

Description automatically generated

Diagram

Description automatically generated with medium confidence

Chart, line chart

Description automatically generated

Overfitting

Chart, line chart

Description automatically generated

Left: underfitting

Right: overfitting

Choose the middle for the best fitting

Noise is an irreducible error since we can’t predic it

We can calculate different R-squared values as follows:

* First, create an empty list to store the values
* Creating a list containing different polynomial orders
* Then, iterate through the list using a loop
* Create a polynomial feature object with the order of the polynomial as a parameter
* Transform the training and test data into a polynomial using the fit transform method
* Fit the regression model using the transform data
* Graphical user interface, text, application

  Description automatically generatedCalculate the R-squared using the test data and store it in the array

**RIDGE REGRESSION INTRODUCTION**

Ridge regression is a regression that is employed in a Multiple regression model when Multicollinearity occurs. Multicollinearity is when there is a strong relationship among the independent variables. Ridge regression is very common with polynomial regression. The next video shows how Ridge regression is used to regularize and reduce the standard errors to avoid over-fitting a regression model.

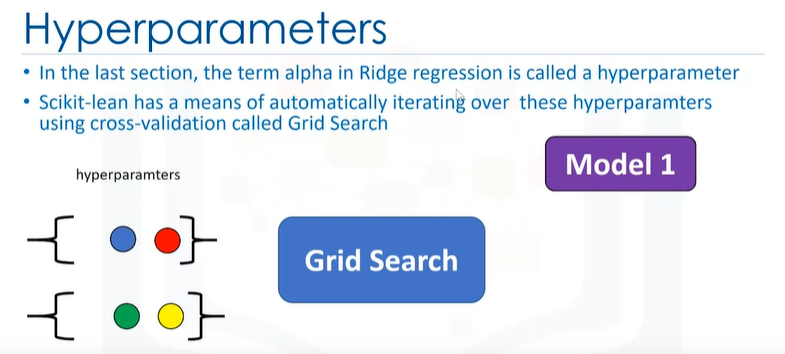
From sklearn.linear\_model import Ridge

RidgeModel = Ridge(alpha=0.1) #Create a ridge object using the constructor

RidgeModel.fit(X,y) #Train the model using the fit method

Yhat = RidgeModel.predict(X) #Make a prediction using the predict method

GRID SEARCH: allows you to scan through multiple free parameters with few lines of code.

Parameters like the alpha term discussed above are not part of the fitting or training process but it is a hyperparameters.

Graphical user interface, text, application, email

Description automatically generated

First, import the libraries we need. Including GridSearchCV, the dictionary of parameter values.

Create a ridge regression object or model

Create a GridSearchCV object. Inputs are ridge regression object, parameter values, and the number of folds

Fit the object

Find the best values for the free parameters using the attribute best estimator

Advantages of Grid Search is how quickly we can test multiple parameters