Mod 5 Python Parte 2

Model Evaluation and Refinement

- 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

Separate data into training and testing sets is an important part of model evaluation.

- Split dataset into:
  - Training set
  - Testing set
- 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

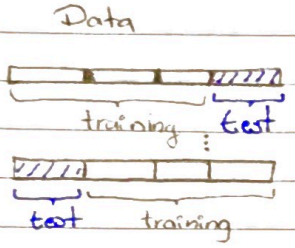
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.3,
                                                    random_state=0)
```

x_data: features or independent variables
y_data: data set 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

Cross Validation

- Most 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 )
```

type of model: type of model we're using to do the cross-validation
lr = linear regression.

x_data: the predictor variable

y_data: target variable data

cv: Manage the number of partitions

cv=3 means the data set is split into 3 equal partitions

```
np.mean(scores)
```

→ The function returns an array of scores, one for each partition that was chosen as the testing set

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)
```

Actual predicted values supplied by our model

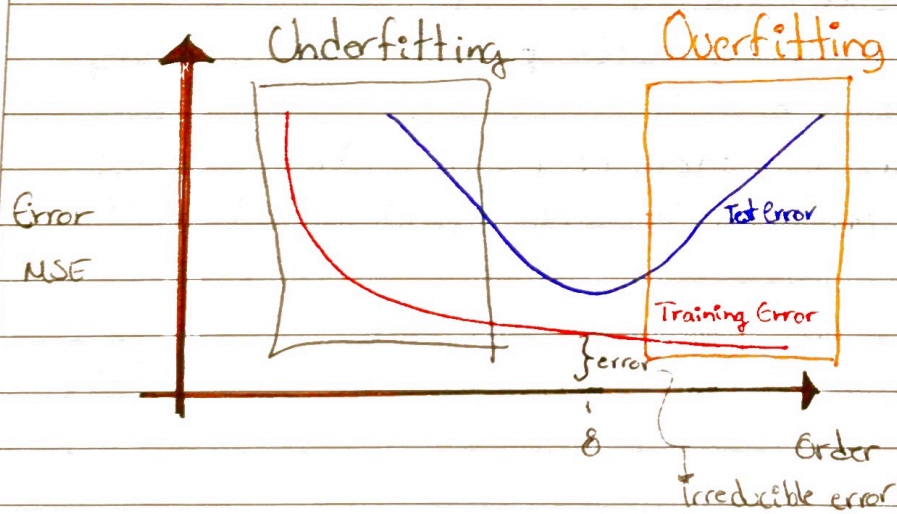
The input parameters are exactly the same as the cross_val_score() function, but the output is a prediction

Predictions are stored in arrays.

Model Selection - Overfitting and Underfitting

The goal of model selection is to determine the order of the polynomial to provide the best estimate of the function.

Overfitting : The model is too flexible and fits the noise rather than the function



Ridge Regression

Ridge regression prevents over-fitting

If we examine the expression for the estimated function, we see the estimated polynomial coefficients have a very large magnitud.
This is especially evident for the higher order polynomials.

Ride regression controls the magnitud of these polynomial coefficients by introducing the parameter alpha.

Alpha is a parameter we select before fitting or training the model.

$$\hat{y} = 1 + 2x - 3x^2 - 2x^3 - 12x^4 - 40x^5 + 80x^6 + 71x^7 - 141x^8 - 38x^9 + 75x^{10}$$

Alpha	x	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	x^{10}
0	2	-3	-2	-12	-40	80	71	-141	-38	75
0.001	2	-3	-7	5	4	-6	4	-4	4	6
0.01	1	-2	-5	-0.04	0.15	-1	1	-0.5	0.3	1
1	0.5	-1	-1	-0.61	0.70	-0.33	-0.56	-0.21	-0.5	-0.1
10	0	-0.5	-0.3	-0.37	-0.30	-0.30	-0.22	-0.22	-0.22	-0.17

As alpha increases, the parameters get smaller.
This is more evident for the higher order polynomial ~~order~~ features, but alpha must be selected carefully.

If alpha is too large, the coefficients will approach zero and under fit the data.
If alpha is zero, the over fitting is evident

Ridge Regression

In order to select alpha we use cross validation

```
from sklearn.linear_model import Ridge

RigeModel = Ridge(alpha=0.1)
RigeModel.fit(X,y)
Yhat = RigeModel.predict(x)
```

Así puedo saber cuál sería un buen valor de alpha.

Alpha		R^2
0.1		0.5
1	Train ⇒ Predict ⇒ R^2	0.75 ✓
10		0.55

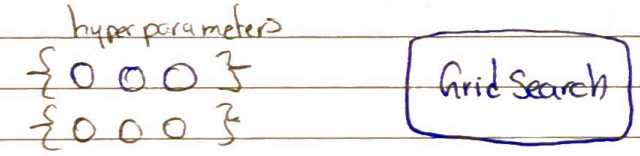
Alpha prevents overfitting

Grid Search

{ Busca los mejores parametros de un modelo

Hyperparameters

- In the last section, the term alpha in Ridge regression is called a hyperparameter.
- Scikit-learn has a means of automatically iterating over these hyperparameters using cross-validation called Grid Search



Use different hyperparameters to train the model
Each model produces an error, we select the hyperparameters that minimizes the error.

Grid Search

Ridge() → Model or object

Scoring

Number of folds
Number of partitions
Data set is split into X partitions

Grid Search CV

Alpha 1 10 100 1000 = Free parameter values

Scoring method : R^2, MSE, etc.

Output

Different scores for different free parameter values

e.i

Alpha	1	10	100	1000
R^2	0.74	0.35	0.073	0.008

Parameters = [{'alpha' : [1, 10, 100, 1000], 'normalize' : [True, False]}]

Alpha	1	10	100	1000
Normalize	True	True	True	True
	False	False	False	False

The Dictionary is a table or grid that contains two different values.