You can either define a **utility function (or fitness function) that measures how good your model is**, or you can define a **cost function that measures how bad it is.** For linear regression problems, people

**What I already know:**

.head()

.info()

.describe()

.value\_counts()

plotting

**What I learned:**

for data of numerical features, use hist()(pairplot on seaborn); this shows how the data is distributed along, its limits and boundaries, biases and disproportions.

creating test sets and validation sets;

typically use a cost function that measures the distance between the linear model’s predictions and the training examples; the objective is to minimize this distance.

Possible solutions for overfitting are:

to simplify the model

constrain it (i.e., regularize it)

or get a lot more training data

You should save every model you experiment with, so you can come back easily to any model you want. Make sure you save both the hyperparameters and the trained parameters, as well as the cross-validation scores and perhaps the actual predictions as well. The goal is to shortlist a few (two to five) promising models. **You can easily save Scikit-Learn models by using Python’s pickle module**