UNDERFITTING

- A statistical model or a machine learning algorithm is said to have underfitting when it cannot capture the underlying trend of the data. *(It's just like trying to fit undersized pants!).*
- Underfitting destroys the accuracy of our machine learning model.
- Its occurrence simply means that our model or the algorithm does not fit the data well enough.
- It usually happens when we have fewer data to build an accurate model and also when we try to build a linear model with fewer non-linear data.

In a nutshell, **Underfitting – High bias and low variance**

Techniques to reduce underfitting:

1. Increase model complexity
2. Increase the number of features, performing feature engineering
3. Remove noise from the data.
4. Increase the number of epochs or increase the duration of training to get better results.

EXTRA NOTES ON MODEL BASICS

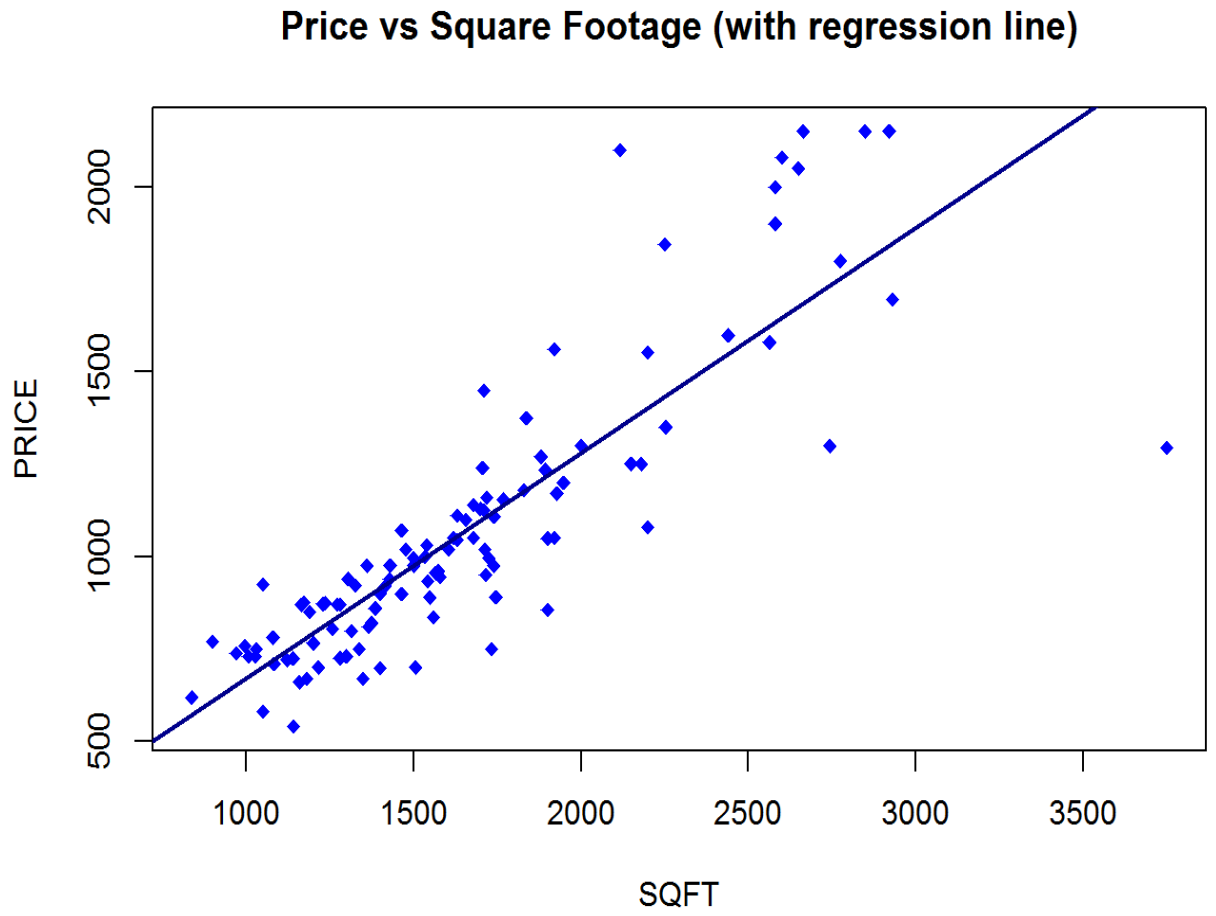
- what is a model?
- A model is simply a system for mapping inputs to outputs.
- For example, if we want to predict house prices, we could make a model that takes in the square footage of a house and outputs a price.
- A model represents a theory about a problem: there is some connection between the square footage and the price and we make a model to learn that relationship.
- Models are useful because we can use them to predict the values of outputs for new data points given the inputs.
- A model learns relationships between the inputs, called features, and outputs, called labels, from a training dataset.

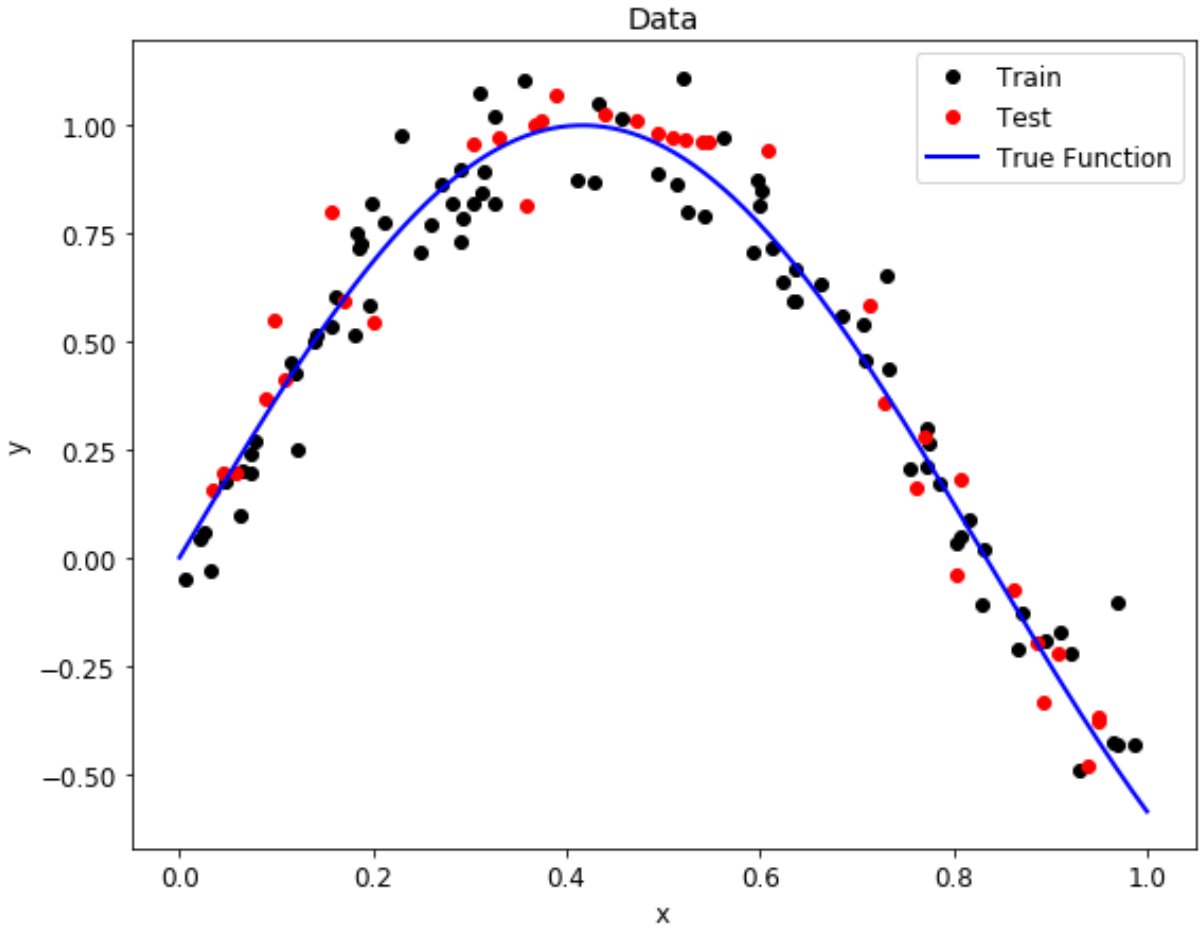
- During training the model is given both the features and the labels and learns how to map the former to the latter.
- A trained model is evaluated on a testing set, where we only give it the features and it makes predictions.
- We compare the predictions with the known labels for the testing set to calculate accuracy.
- Models can take many shapes, from simple linear regressions to deep neural networks, but all supervised models are based on the fundamental idea of learning relationships between inputs and outputs from training data.

Training and Testing Data

- To make a model, we first need data that has an underlying relationship.
- For this example, we will create our own simple dataset with x-values (features) and y-values (labels).
- An important part of our data generation is adding random noise to the labels.
- In any real-world process, whether natural or man-made, the data does not exactly fit to a trend.
- There is always noise or other variables in the relationship we cannot measure.
- In the house price example, the trend between area and price is linear, but the prices do not lie exactly on a line because of other factors influencing house prices.

EXAMPLE


Price vs Square Footage (with regression line)
PRICE
500
1000
1500
2000
1000
1500
2000
2500
3000
3500
SQFT



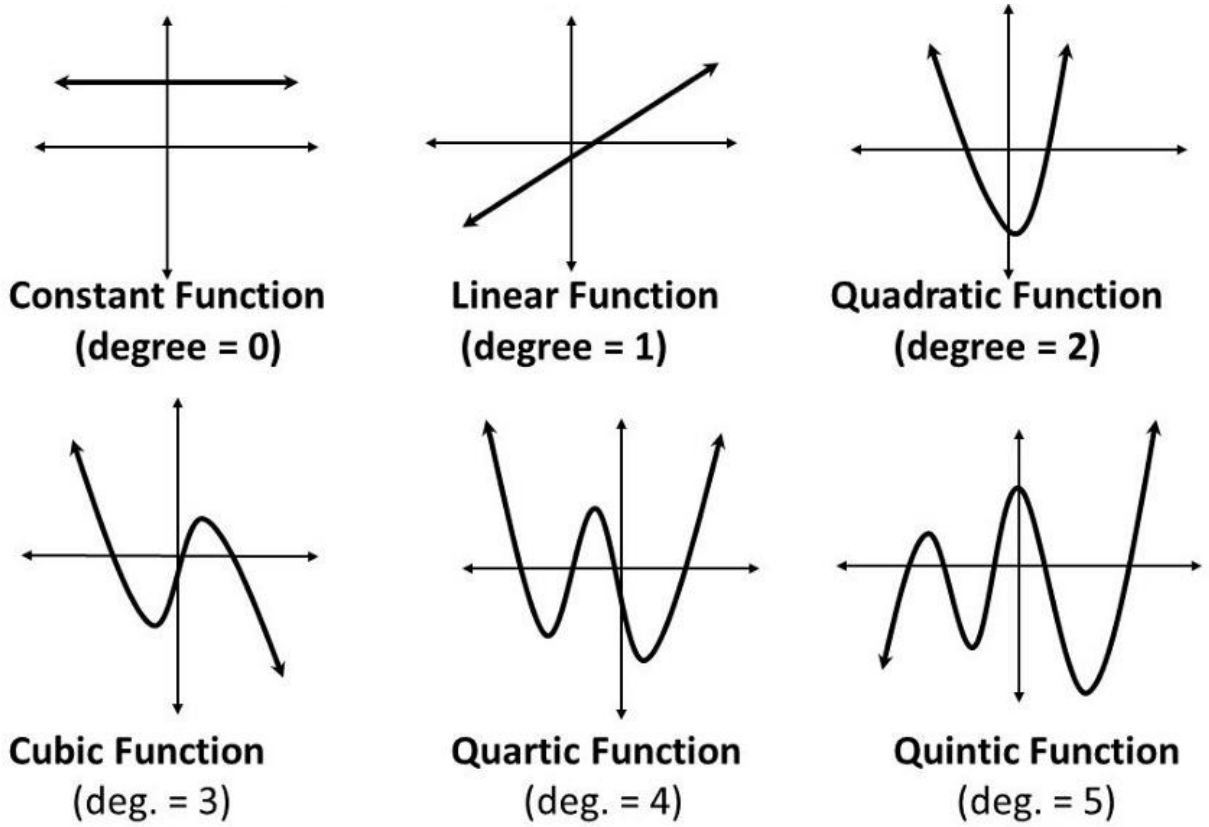
- Our data similarly has a trend (which we call the true function) and random noise to make it more realistic.
- After creating the data, we split it into random training and testing sets. The model will attempt to learn the relationship on the training data and be evaluated on the test data.
- In this case, 70% of the data is used for training and 30% for testing. The graph shows the data we will explore.
- We can see that our data are distributed with some variation around the true function (a partial sine wave) because of the random noise we added.
- During training, we want our model to learn the true function without being “distracted” by the noise.

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_n x^n + \varepsilon.$$

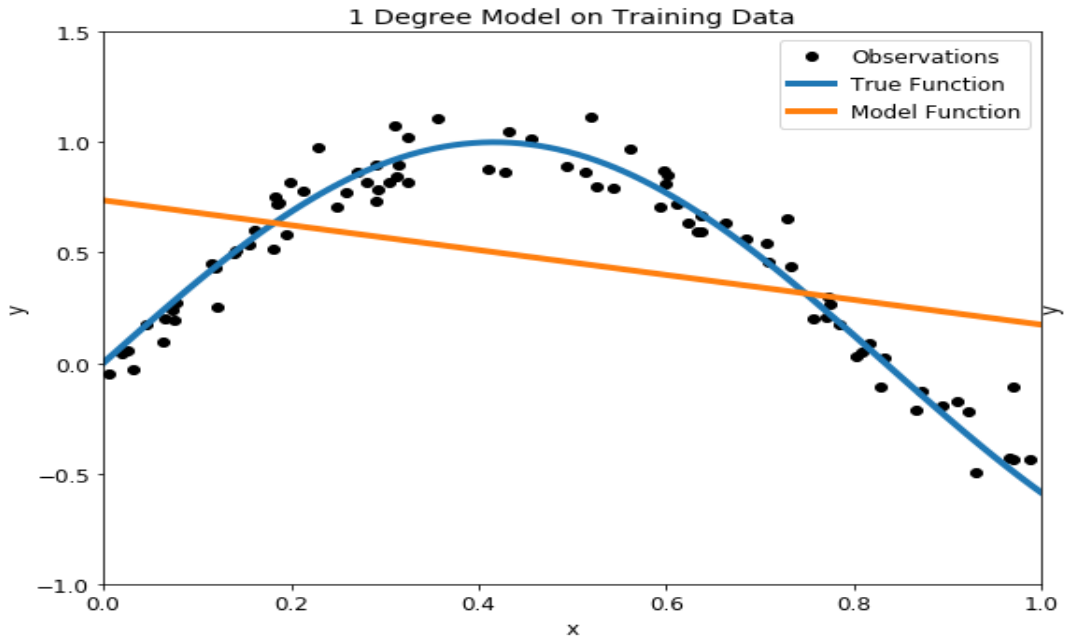
Model Building:

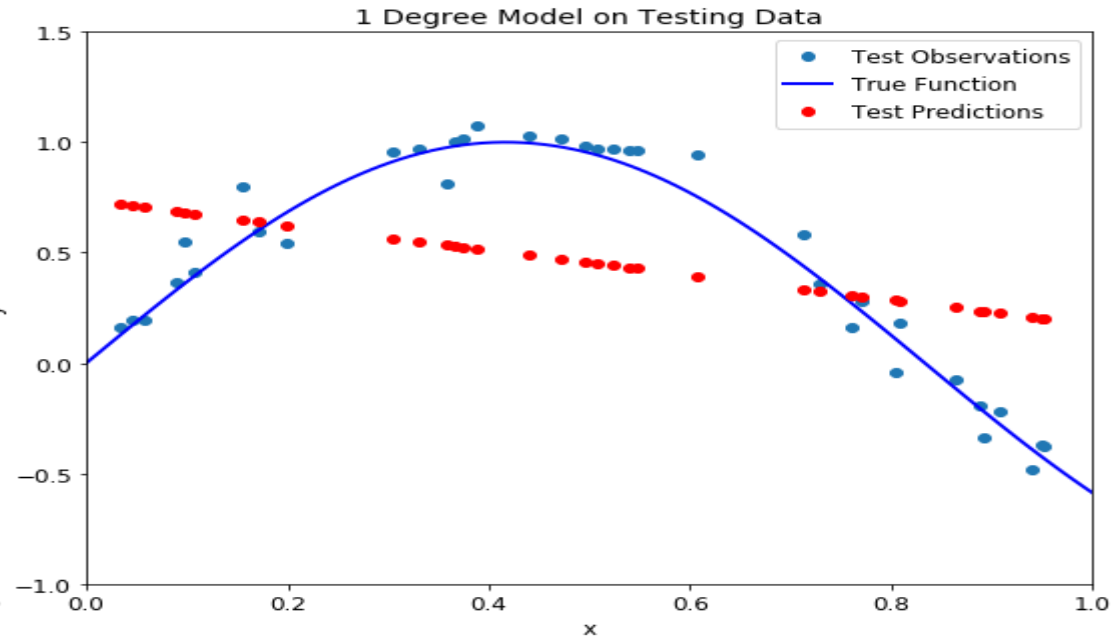
- Choosing a model can seem intimidating, but a good rule is to start simple and then build your way up.
- The simplest model is a linear regression, where the outputs are a linearly weighted combination of the inputs.
- In our model, we will use an extension of linear regression called polynomial regression to learn the relationship between x and y. Polynomial regression, where the inputs are raised to different powers, is still considered a form of “linear” regression even though the graph does not form a straight line.
- The general equation for a polynomial is: (left side)
- Here y represents the label and x is the feature.
- The beta terms are the model parameters which will be learned during training,
- The epsilon is the error present in any model.
- Once the model has learned the beta values, we can plug in any value for x and get a corresponding prediction for y. A polynomial is defined by its order, which is the highest power of x in the equation. A straight line is a polynomial of degree 1 while a parabola has 2 degrees.

- FIG: POLYNOMIALS OF VARYING DEGREE



- The degree represents how much flexibility is in the model, with a higher power allowing the model freedom to hit as many data points as possible.
- **UNDERFITTING:**
- An underfit model will be less flexible and cannot account for the data.
- EG:--an underfit model with a 1 degree polynomial fit. In the image on the left, model function in orange is shown on top of the true function and the training observations. On the right, the model predictions for the testing data are shown compared to the true function and testing data points.





- Our model passes straight through the training set with no regard for the data!
- This is because an underfit model has low variance and high bias.
- Variance refers to how much the model is dependent on the training data.
- For the case of a 1 degree polynomial, the model depends very little on the training data because it barely pays any attention to the points.
- Instead, the model has high bias, which means it makes a strong assumption about the data.
- For this example, the assumption is that the data is linear, which is evidently quite wrong. When the model makes test predictions, the bias leads it to make inaccurate estimates.
- The model failed to learn the relationship between x and y because of this bias.
- A low degree leads to underfitting.
- A natural conclusion would be to learn the training data, we should just increase the degree of the model to capture every change in the data.
- ----------