# Material Summary: Linear and Logistic Regression

## Linear Regression

* 1. **Linear Regression Intuition**
* Regression – predicting a continuous variable
* Problem statement
  + Given pairs of points, create a model
    - Input , output ; **goal: predict given** 
      * Under the assumption that depends linearly on (and nothing else)
* Modelling function
  + Many samples: for each sample :
    - ,
  + Many variables:
    - Trick:

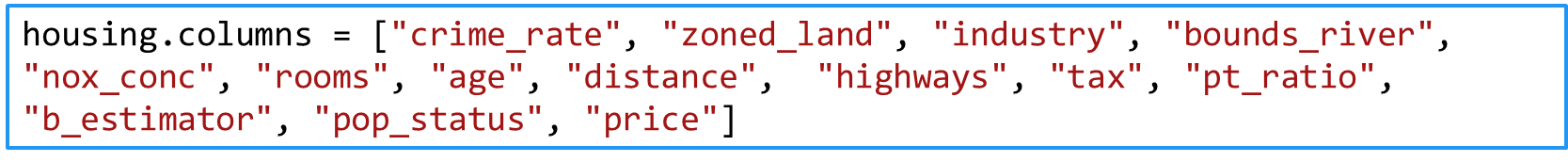
A graph of a graph

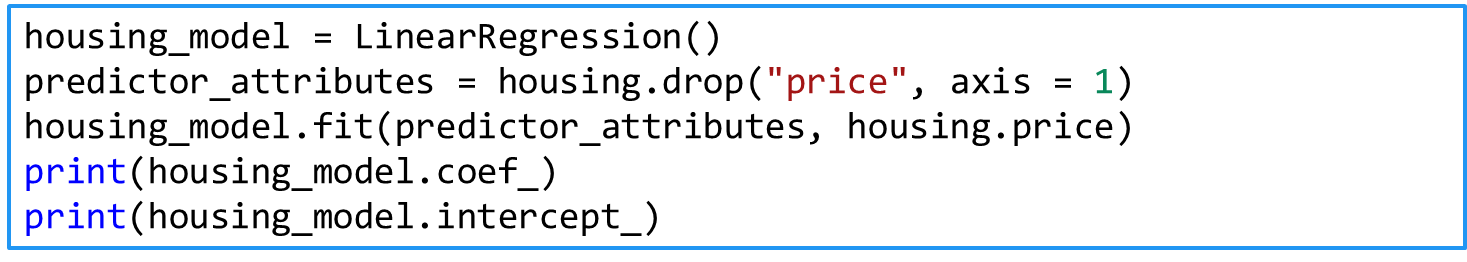
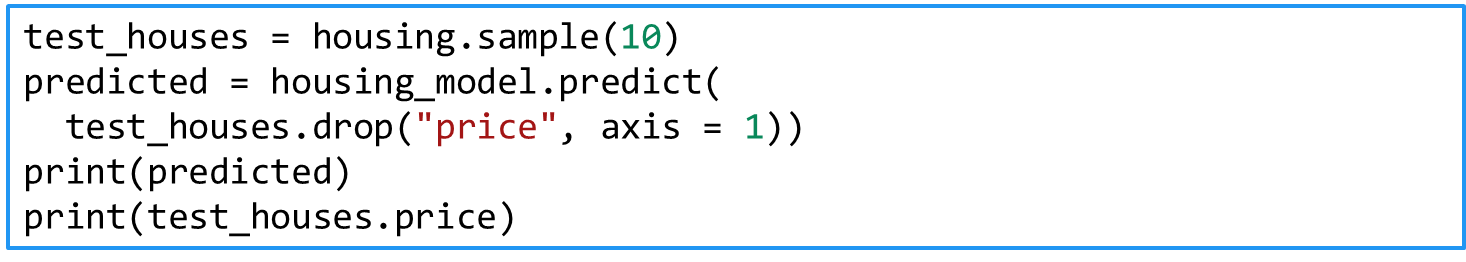
Description automatically generated**1.2 Training**

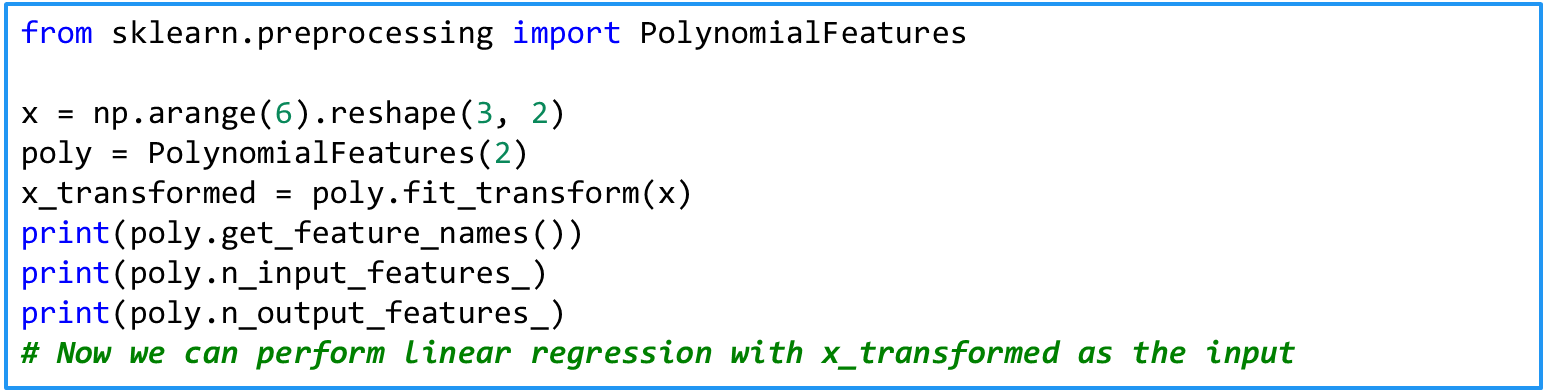
* Loss function
  + For each sample ,
* Total cost function
  + Also called simply "cost function"
  + depends on
* Training process
  + Minimize the cost function
    - We're looking for parameters that lead to
    - Written as
  1. **Gradient Descent**
* Input: ; output
* **A graph of a graph

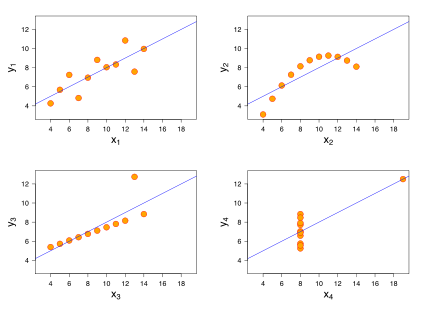
  Description automatically generated with medium confidence**Paraboloid (3D parabola)
  + It has exactly one min value
    - And we can see it
* Intuition
  + If the plot was a real object (say, a sheet of some sort), we could slide a ball bearing on it
  + After a while, the ball bearing will settle at the "bottom"due to gravity
  + We can "simulate" this: **gradient descent**
* **A black background with a black square

  Description automatically generated with medium confidence**Reminder: gradient
  + "Multi-dimensional derivative"
* Iterative algorithm – perform as many times as needed
  + Start from some point in the space:
  + Decide how big steps to take: number
    - Called learning rate in ML terminology
  + Use the current and to compute
    - tells us how much to move in the direction in order to get to the minimum
    - Similar for
  + Take a step with size in each direction
    - are the new coordinates
  + Repeat the two preceding steps as needed
    - Usually, we do this for a fixed number of iterations
  1. **Example: Housing Prices**
* Multiple linear regression
  + Many predictor variables
* Let's use this model to try and predict housing prices (a classical dataset located [*here*](https://github.com/rupakc/UCI-Data-Analysis/tree/master/Boston%20Housing%20Dataset/Boston%20Housing))



* First, we want to explore the datasets
  + A more thorough exploration is "left as an exercise to the reader"
  + But we want to see what model would be appropriate
    - In addition to usual data analysis techniques, let's plot all correlations between any pair of features
  1. **Creating a Model**
* Modelling is very simple
  + Like in the 2D example
* So what?
  + We might want to predict some prices
  + Let's just pass some random rows and see the result
  + **Note: Never test on the training dataset!**
  1. **Delving Deeper into Matrices**
* Dataset:
* Parameters:
* **Modelling function:** 
  1. **Regression with Outliers**
* As we saw, the data has outliers
  + A few points which are far from the others
* Our goal is to exclude outliers
  + There are several methods
  + One very common – RANSAC (**RAN**dom **SA**mple **C**onsensus)
* Algorithm
  + Fit a model to a random subsample ("inliers")
  + Test all data points and include those which are "near" the model
    - Small enough error, tolerance provided by developer
  + Fit the model again
  + Estimate the error of the model (difference between first and second)
  + **A cartoon of a person drawing a line

    Description automatically generated**Iterate steps 1-4 until performance reaches a threshold or number of iterations
  1. **Polynomial Regression**
* Extension of the linear regression algorithm
  + We can use the linear regression algorithm to perform polynomial regression (e.g., fitting a quadratic curve)
    - **Just precompute the columns**
    - Example: if we have columns x, y and z, compute x \* z, y \* z, x \* z   
      and perform linear regression on these 6 features
    - Example 2: polynomial terms: multiply by itself: x \* x, x \* x \* x, etc.
* ****This can be achieved easily with scikit-learn
  1. **Common Mistakes**
* There are two main types of errors we can make while trying regression models
  + Use a **wrong model**
    - Anscombe's quartet
  + **Extrapolate** without knowing (especially if we have interacting features)

**A cartoon of a person pointing at a graph

Description automatically generated**

## Logistic Regression

**2.1 Classification**

* Predict one of several known classes
  + Based on the input parameters
  + Example: classify whether a picture is of a cat or a dog
* Regression and classification make up most of the machine   
  learning problems
* Choosing an algorithm
  + "No free lunch": no single algorithm works best
  + It's best to compare some algorithms to select the best for a particular model
    - Also, we might want to tune them first
* Reminder: ML process
  + Select features, choose a performance metric (cost function), choose a classifier, evaluate and fine-tune the performance

**2.2 Logistic Regression**

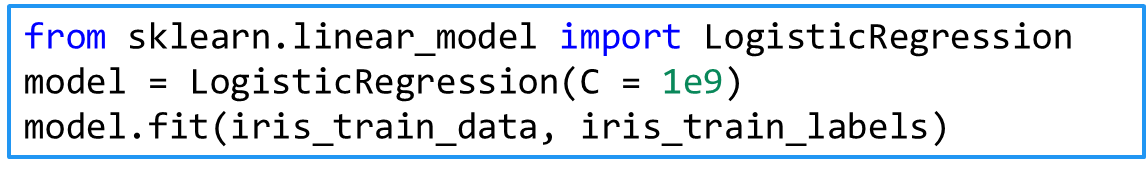
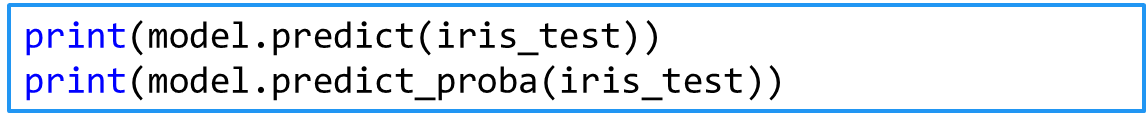
* **A graph of a function

  Description automatically generated**Classification algorithm (despite its name)
* Two classes: negative (0) and positive (1)
  + Can be extended to more classes
* How does it work?
  + Linear regression can give us all kinds of values
  + We want to constrain them between 0 and 1
  + Approach
    - Perform linear regression:
    - Use the sigmoid function to constrain the output:
    - Quantization: if return 1, and 0 otherwise
      * Remember that we only need to return 0 or 1
      * We can also use the raw values as probability measures

**2.3 Example: Classifying Iris Flowers**

* A classic dataset for classification is the Iris dataset
  + Located[*here*](https://archive.ics.uci.edu/dataset/53/iris)
  + 3 classes (setosa, virginica, versicolor)
  + 4 attributes: petal width / height; sepal width / height (all in cm)
    - Some features are highly correlated to the class
  + **A purple flower with yellow center

    Description automatically generated**A close up of a flower

    Description automatically generatedExplore and inspect the data before modelling
* Perform logistic regression
* Test (output classes or probabilities)
* In the model, there's a "mysterious" parameter C
  + Regularization: how powerful the data is (more – next time)
  + A large number means no regularization
    - We just take the data "as-is", with no other constraints

**2.4 Many Classes**

* Two main approaches
  + One-vs-all: several predictors
    - One predictor for each class vs. the others
  + Overall: calculate probabilities of each class
* scikit-learn takes care of multiple classes (multinomial logistic regression) by default
  + We don't even need to transform the labels
  + This applies to all algorithms in the library