Introduction to Machine Learning

Giovanni Della Lunga

WORKSHOP IN QUANTITATIVE FINANCE

Bologna - May 17-29, 2021

Outline

Introduction

- These introductory lessons assume a basic level of statistical and mathematical knowledge.
- No previous knowledge of Machine Learning is assumed.
- For this reason I have decided to use two basic texts for the preparation of these lessons that you can consult for further details on the topics we are going to deal with.

John C. Hull, **Machine Learning in Business, An Introduction to the World of Data Science**, Amazon (2019)

Paul Wilmott, **Machine Learning, An Applied Mathematics Introduction**, Panda Ohana Publishing (2019)

What is Machine Learning

- Machine learning is a branch of AI
- The idea underlying machine learning is that we give a computer program access to lots of data and let it learn about relationships between variables and make predictions
- Some of the techniques of machine learning date back to the 1950s but improvements in computer speeds and data storage costs have now made machine learning a practical tool
- Machine Larning or data science can be described as the new world of statistics

Software

- There a several alternatives such as Python, R, MatLab, Spark, and Julia
- Need ability to handle very large data sets and availability of packages that implement the algorithms.
- Python seems to be winning at the moment
- Scikit-Learn has freely available packages for many ML tasks

A word of caution. It is tempting to learn a language such as Python and apply various packages to your data without really understanding what the packages are doing or even how the results should be interpreted. This would be a bit like a finance specialist using the Black-Scholes model to value options without understanding where it comes from or its limitations...

Traditional statistics

- Means, SDs
- Probability distributions
- Significance tests
- Confidence intervals
- Linear regression
- etc

The new world of statistics

- Huge data sets
- Fantastic improvements in computer processing speeds and data storage costs
- Machine learning tools are now feasible
- Can now develop non-linear prediction models, find patterns in data in ways that were not possible before, and develop multi-stage decision strategies
- New terminology: features, labels, activation functions, target, bias, supervised/unsupervised learning

Applications of ML

- Credit decisions
- Classifying and understanding customers better
- Portfolio management
- Private equity
- Language translation
- Voice recognition
- Biometrics
- etc

Types of Machine Learning

- Unsupervised learning (find patterns)
- Supervised learning (predict numerical value or classification)
- Semi-supervised learning (only part of data has values for, or classification of, target)
- Reinforcement learning (multi-stage decision making)

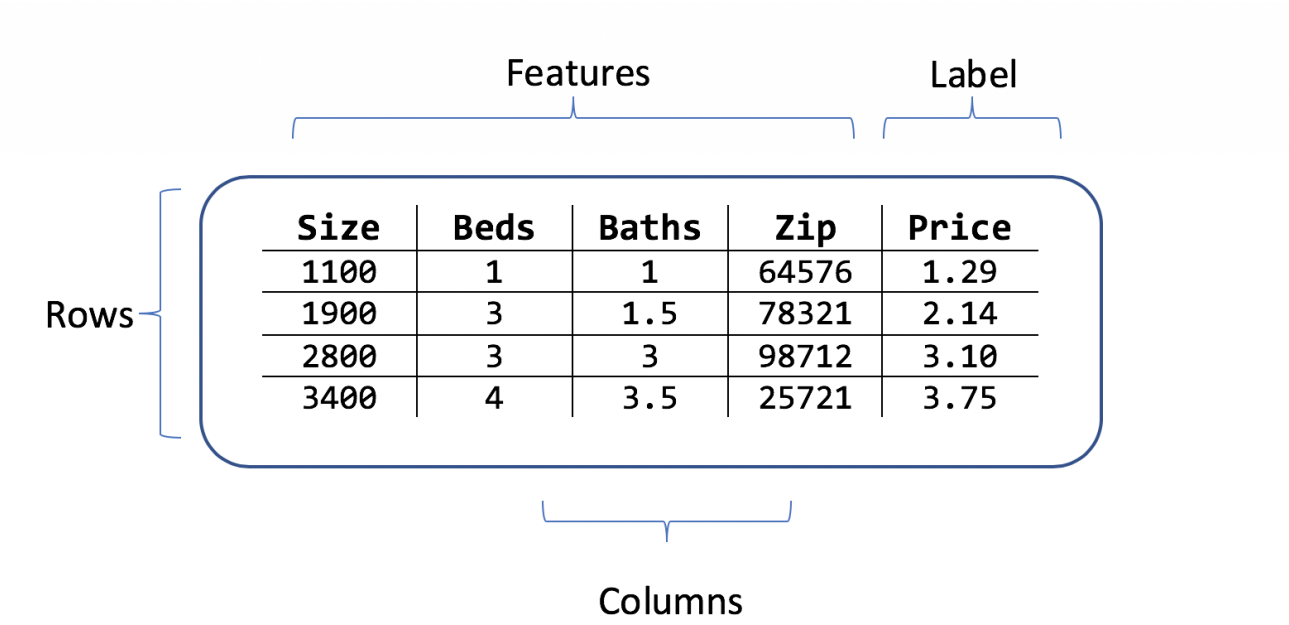
In these introductory lessons we will discuss only the first two types.

Types of Machine Learning

- **Supervised Learning**
  is concerned with using data to make prediction (for example a simple regression model). We can distinguish between supervised learning models that are used to predict a variable than can take a continuum of values and models that are used for classification (for example to learn if a potential borrower can be classified as acceptable or unacceptable credit risk);
- **Unsupervided Learning**
  is most concerned with recognizing patters in data. The main objective usually is not to forecast a particular variable, rather it is to undersand the environment represented by the data better. As we will see clustering is one of the most used tool in unsupervised learning.

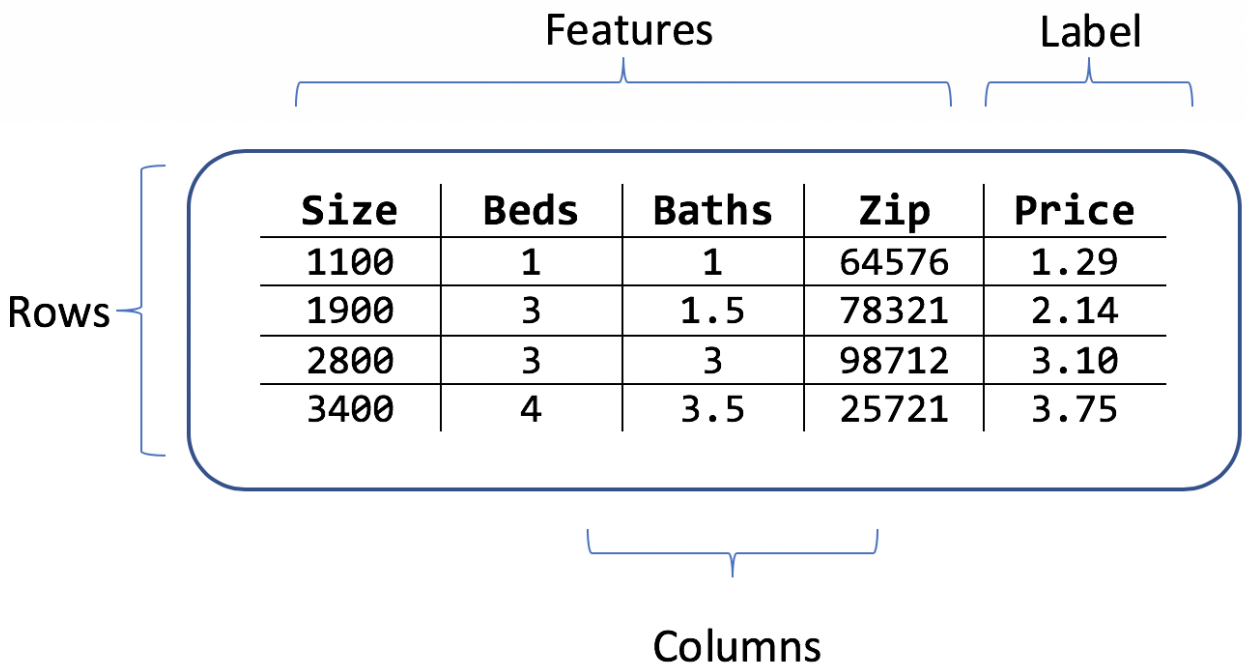
Features and Labels

- The data for **Supervised Learning** contains what are referred to as **features** and **labels**;
- *Labels* are the values of the target that is to be predicted;
- *Features* are the variables from which the predictions are to be made (if you are from statistics then think of it as an explanatory variable);



Features and Labels

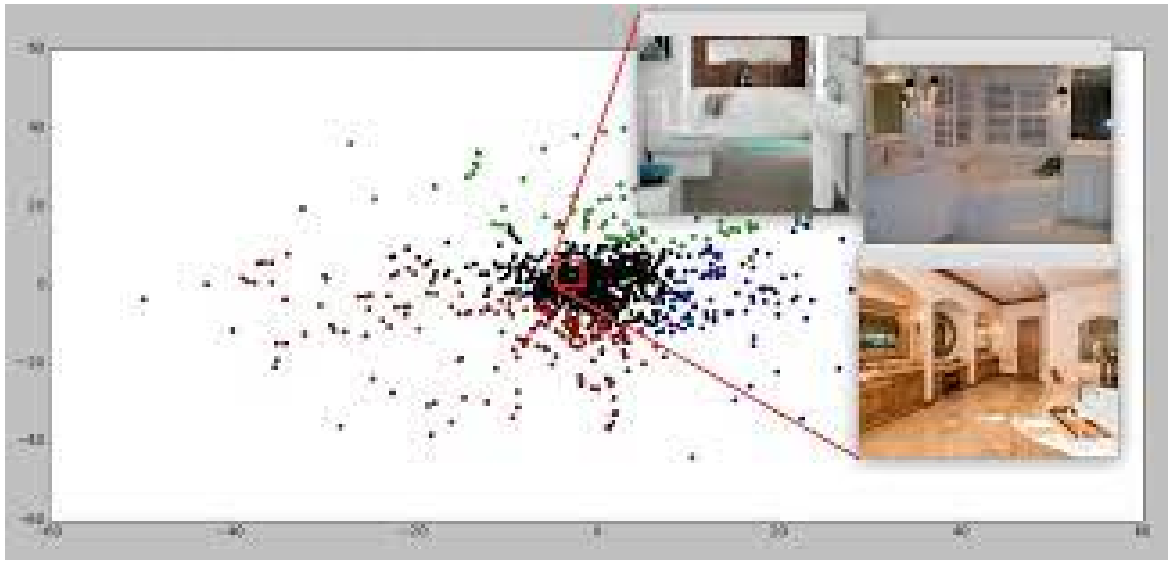
- For example, when predicting the **price of a house**, the *features* could be the *square meters of living space*, *the number of bedrooms*, *the number of bathrooms*, *the size of the garage* and so on.
- The *label* would be *the house price*;



Size	Beds	Baths	Zip	Price
1100	1	1	64576	1.29
1900	3	1.5	78321	2.14
2800	3	3	98712	3.10
3400	4	3.5	25721	3.75

Features and Labels

- The data for **Unsupervised Learning** consists of features but no labels because the model is being used to identify patterns not to forecast something.
- For example we could use an unsupervised model to classify the houses that exist in a certain neghborhood without trying to predict any price.



Feature Scaling

- Before using many ML algorithms (including those for unsupervised learning), it is important to scale feature values so that they are comparable.
- Z-score scaling involves calculating the mean (μ) and SD (σ) from the values of each feature from the training set. Scaled feature values for all data sets are then created by subtracting the mean and dividing by the SD.
- The scaled feature values have a mean of zero and SD of one.

$$\text{scaled feature value} = \frac{V - \mu}{\sigma} \tag{1}$$

Feature Scaling

- Min-max scaling involves calculating the maximum and minimum value of each feature from the training set.
- Scaled feature values for all data sets are then created by subtracting the minimum and dividing by the difference between the maximum and minimum.
- The scaled feature values lie between zero and one:

$$\text{scaled feature value} = \frac{V - V_{min}}{V_{max} - V_{min}} \tag{2}$$

Cleaning data

- Dealing with inconsistent recording
- Removing unwanted observations
- Removing duplicates
- Investigating outliers
- Dealing with missing items

Cost Function

- In Machine Learning a **Cost Function** or loss function is used to represent how far away a mathematical model is from the real data;
- One adjust the mathematical model usually by varying parameters within the model so as to minimize the const function;
- Let's take for example a very simple model of the form

$$y = \theta_0 + \theta_1 x$$

where the θs are the parameters that we want to find to give us the best fit to the data;

- Call this function $h_\theta(x)$ to emphasize the dependence on both the variable x and the two parameters θ_0 and θ_1;

Cost Function

- We want to measure how far away the data, the $y^{(n)}$s are from the function $h_\theta(x)$;
- A common way to do this is via the quadratic cost function

$$J(\theta) = \frac{1}{2N} \sum_{n=1}^{N} \left[h_\theta \left(x^{(n)} \right) - y^{(n)} \right]^2 \tag{3}$$

- We wants the parameters that minimize (**??**), almost always you are going to have to do this numerically;
- If we have a nice convex function then there is a numerical method that will converge to the solution, it is called **gradient descent**.

Gradient Descent

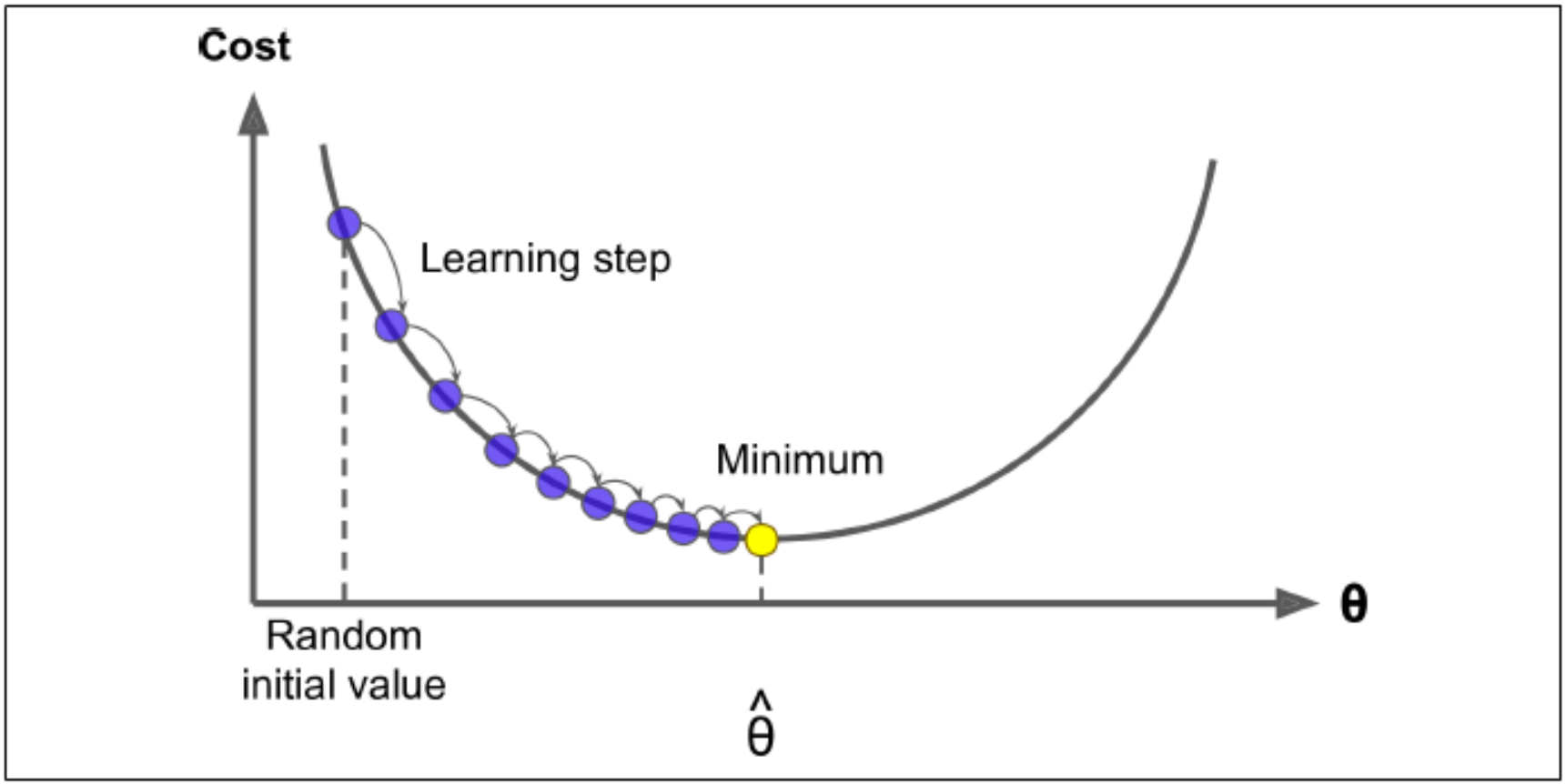
Linear Regression

- Gradient Descent is a very generic optimization algorithm capable of finding optimal solutions to a wide range of problems.
- Gradient Descent Algorithm measures the local gradient of the error function with regards to the parameter vector θ, and it goes in the direction of descending gradient.
- Once the gradient is zero, you have reached a minimum!
- Concretely, you start by filling θ with random values (this is called random initialization), and then you improve it gradually, taking one step at a time, each step attempting to decrease the cost function (e.g., the MSE), until the algorithm converges to a minimum

Gradient Descent

Linear Regression

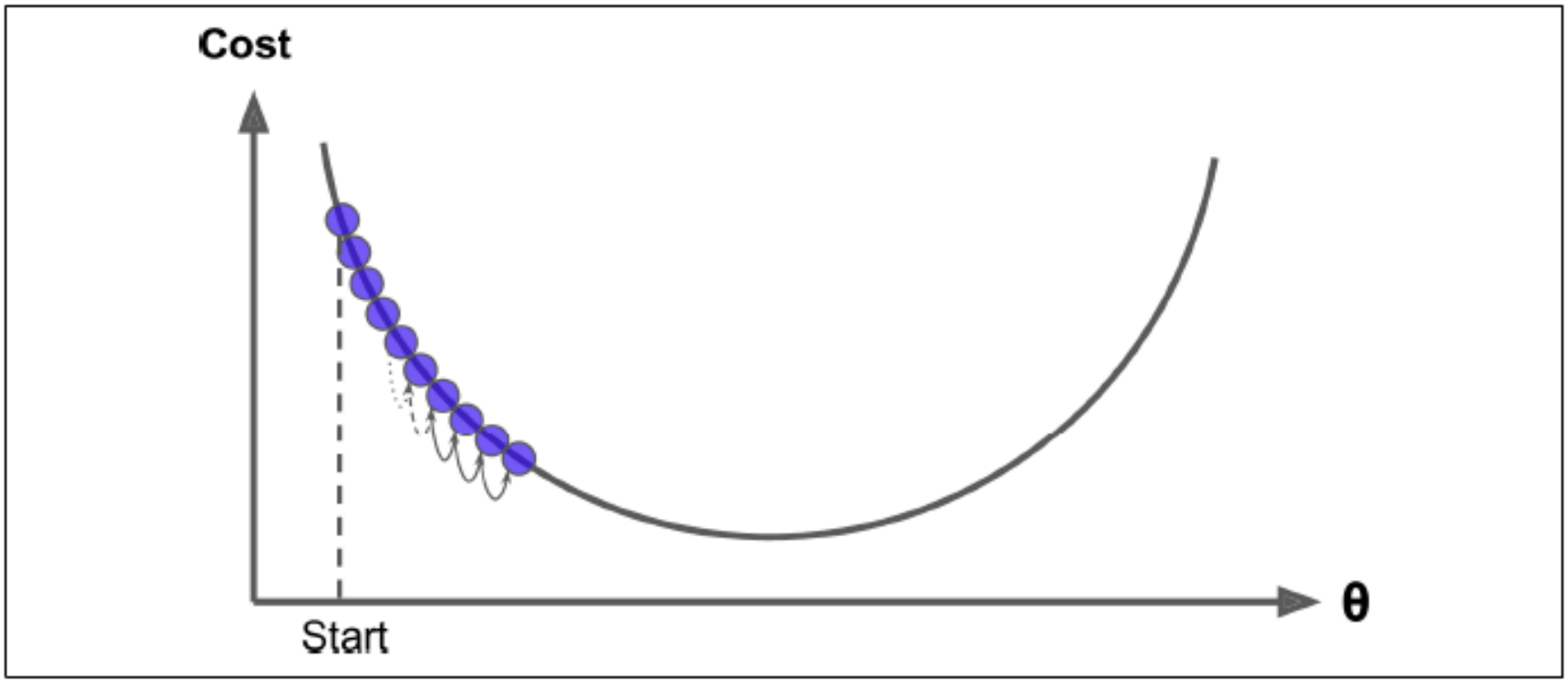
An important parameter in Gradient Descent is the size of the steps, determined by the learning rate hyperparameter.



Gradient Descent

Linear Regression

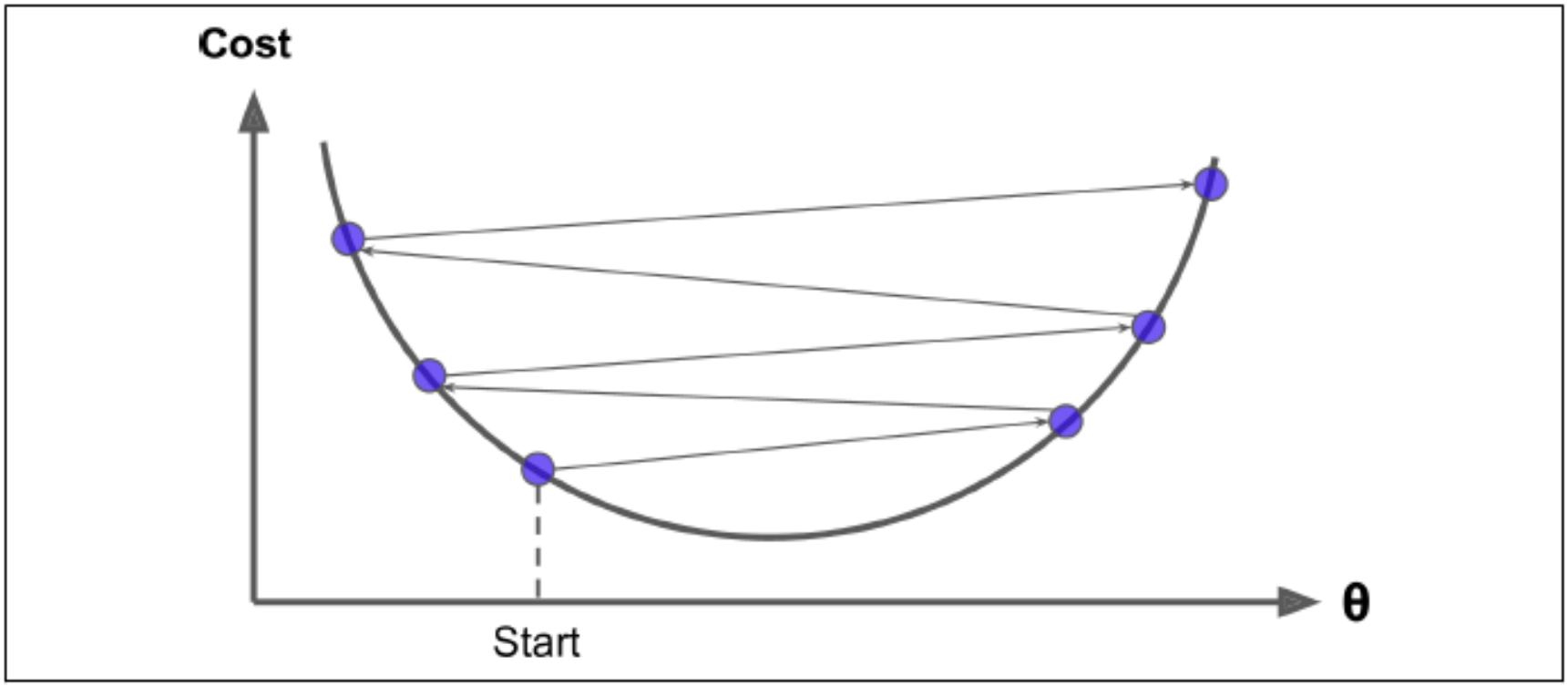
If the learning rate is too small, then the algorithm will have to go through many iterations to converge, which will take a long time...



Gradient Descent

Linear Regression

... on the other hand, if the learning rate is too high, you might jump across the valley. This might make the algorithm diverge failing to find a good solution.



Gradient Descent

Having defined an appropriate learning rate, the scheme works as follow

- Start with an initial guess for each parameter θ_k;
- Move θ_k in the direction of the slope

$$\text{New } \theta_k = \text{Old } \theta_k - \beta \frac{\partial J}{\partial \theta_k}$$

where β is our learning rate;

In the above description of gradient descent we have used all of the data points simultaneously. This is called **batch gradient descent**

Gradient Descent

- Rather then use all of the data in the parameter updating we can use a technique called **stochastic gradient descent**;
- This technique is the same as the batch gradient descent except that you only update using **one** of the data points each time and this data point is chosen **randomly**;
- Especially in high-dimensional optimization problems this reduces the computational burden, achieving faster iterations in trade for a lower convergence rate;
- When the learning rates decrease with an appropriate rate, and subject to relatively mild assumptions, stochastic gradient descent converges almost surely to a global minimum when the objective function is convex or pseudoconvex, and otherwise converges almost surely to a local minimum;

Validation and Testing

- When data is used for forecasting there is a danger that the machine learning model will work very well for data, but will not generalize well to other data;
- An obvious point is that it is important that the data used in a machine learning model be representative of the situations to which the model is to be applied;
- It is also important to test a model **out-of-sample**, by this we mean that the model should be tested on data that is different from the sample data used to determine the parameters of the model;
- Data scientist refer to the sample data as the **training set** and the data used to determine the accuracy of the model as the **test set**;
- Often a **validation set** is used as well as we explain later;

Validation and Testing

- We will illustrate the use of a training set and the test data set with a very simple Example (Hull, Chapter 1);
- We suppose that awe are interested in forecasting the salaries of people from their age;
- This simple regression model is an example of supervised learning...

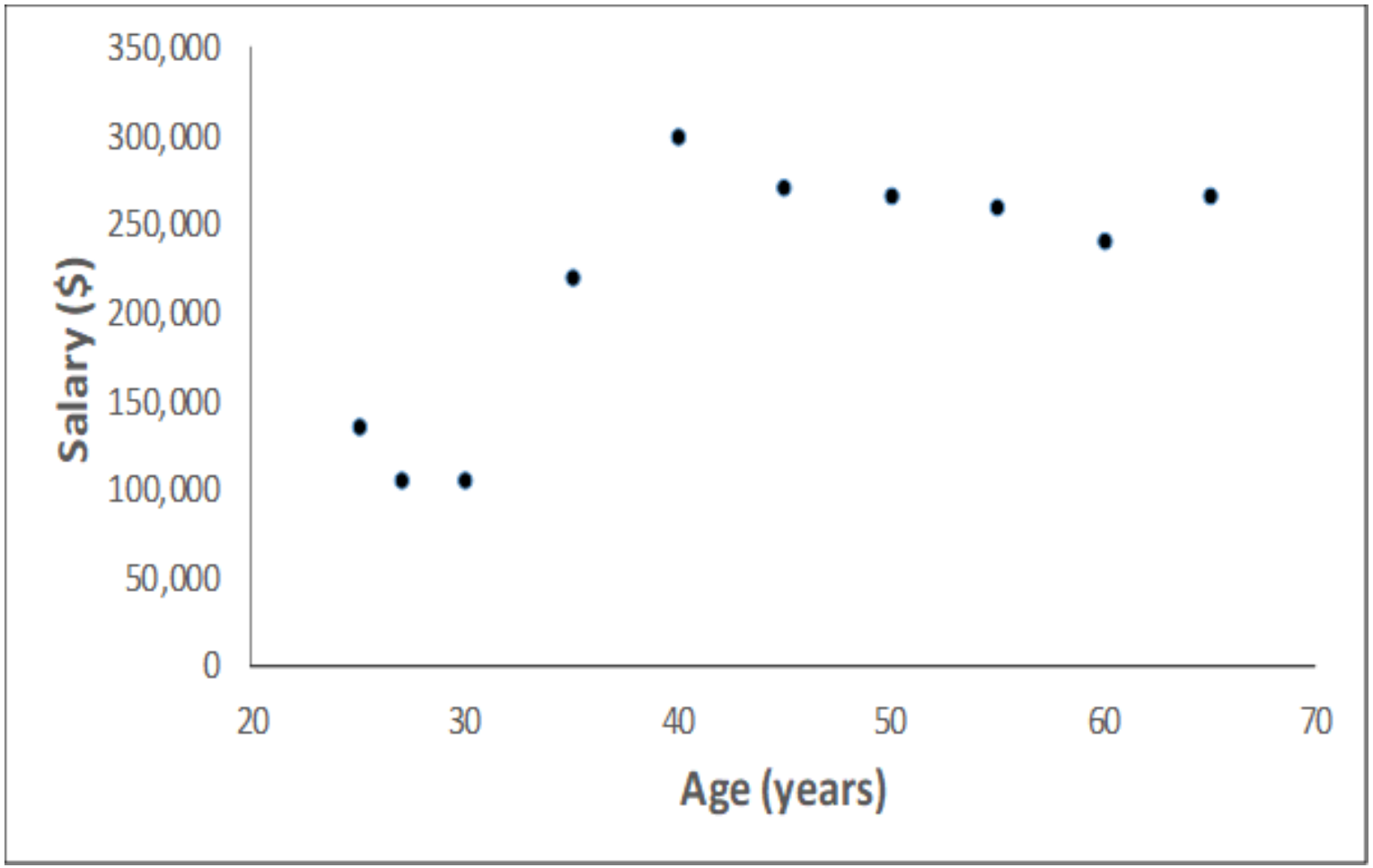
Validation and Testing: Training Set

Table 1. Salary as a function of Age for a certain profession in a certain area)

Age (years)	Salary ($)
25	135,000
55	260,000
27	105,000
35	220,000
60	240,000
65	265,000
45	270,000
40	300,000
50	265,000
30	105,000

Validation and Testing: Training Set

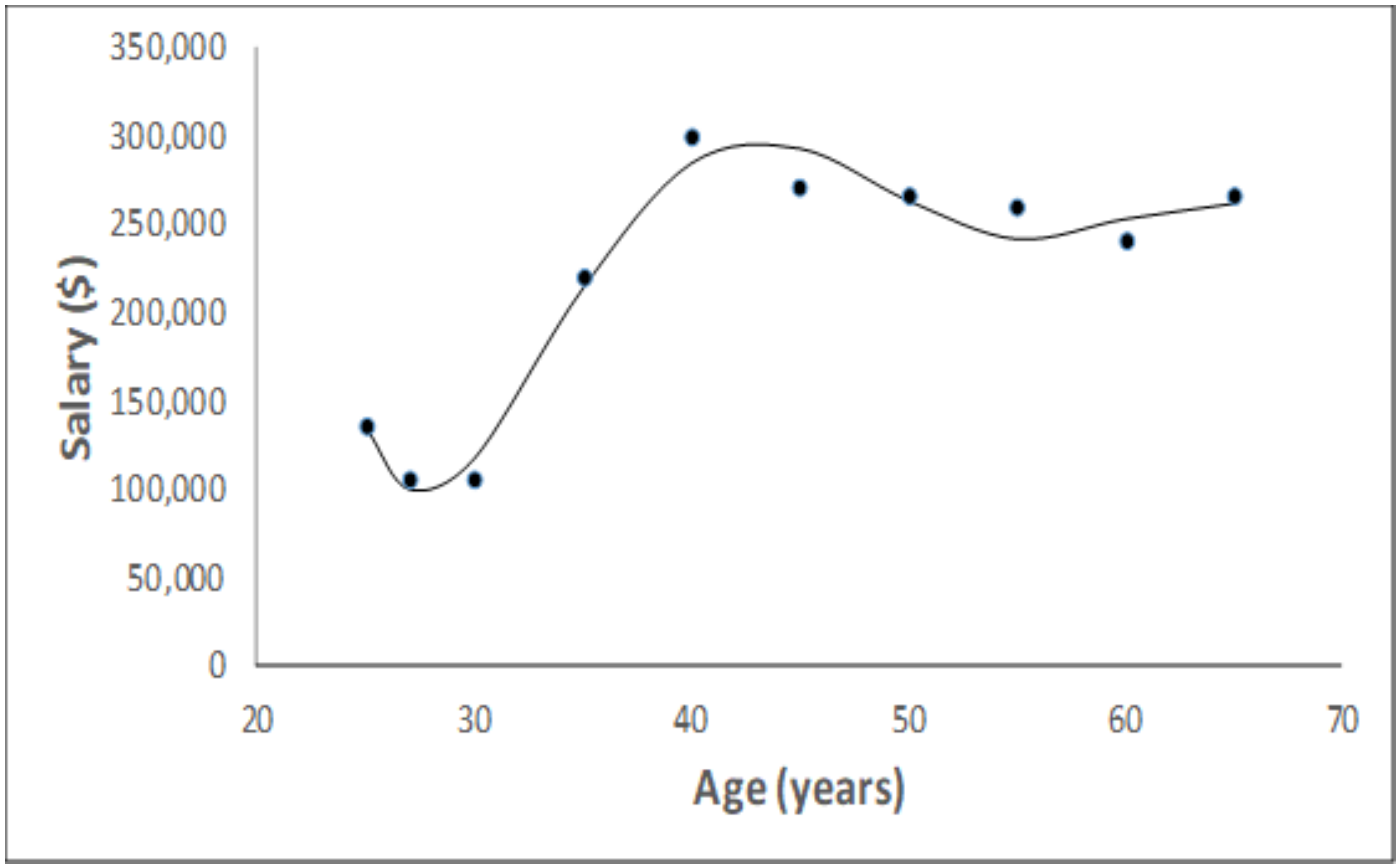
Figure 1. Scatter plot of Salary as a function of Age (see Table 1)



Validation and Testing: Training Set

Figure 2. It's tempting to choose a model that fits the data really well for example with a polynomial of degree five (Y = Salary, X = Age):

$$Y = a + b_1 X + b_2 X^2 + b_3 X^3 + b_4 X^4 + b_5 X^5$$



Validation and Testing: Discussion of Training Result

- The model provides a good fit to the data;
- The standard deviation of the difference between the salary given by the model and the actual salary for the ten individuals in the training data set (which is referred to as the **root mean square error (rmse)**) is \$12902;
- However common sense would suggest that we many over-fitted the data;
- We need to check the model out-of-sample;
- To use the language of *data science* we need to determine whether the model generalizes well to a new data set that is different from the training set.

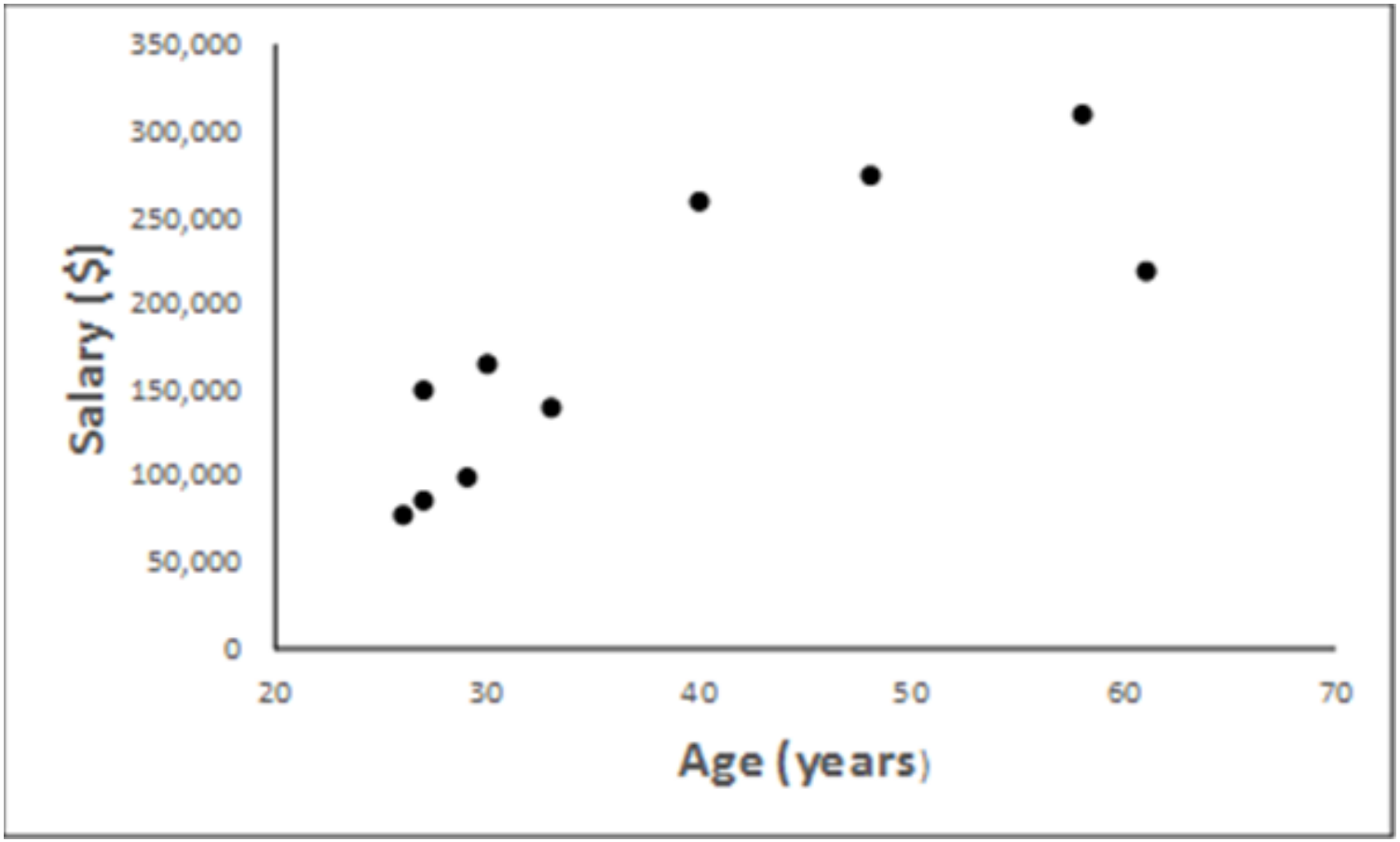
Validation and Testing: Validation Set

Table 2. An Out-of-Sample Validation Set

Age (years)	Salary ($)
30	166,000
26	78,000
58	310,000
29	100,000
40	260,000
27	150,000
33	140,000
61	220,000
27	86,000
48	276,000

Validation and Testing: Validation Set

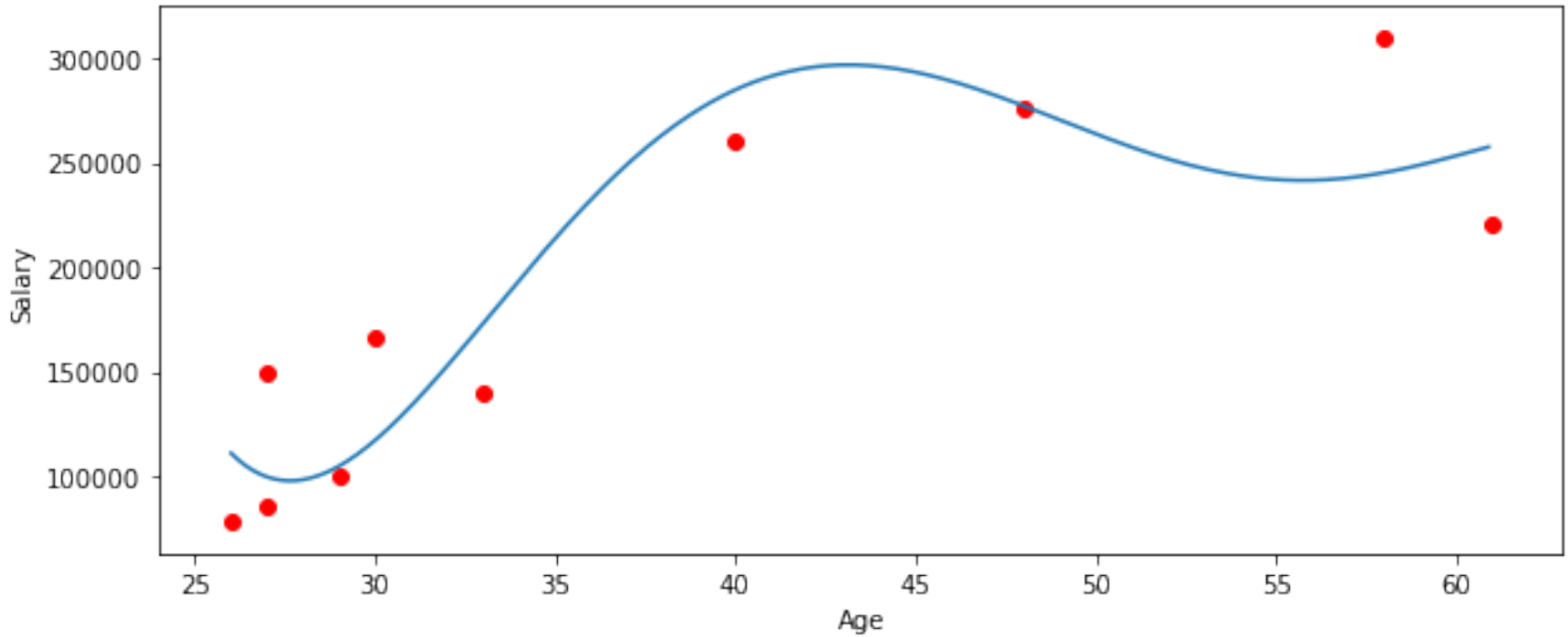
Figure 3. Scatter Plot for Validation Set



Validation and Testing: Discussion of Validation Result

The Fifth Order Polynomial Model Does Not Generalize Well

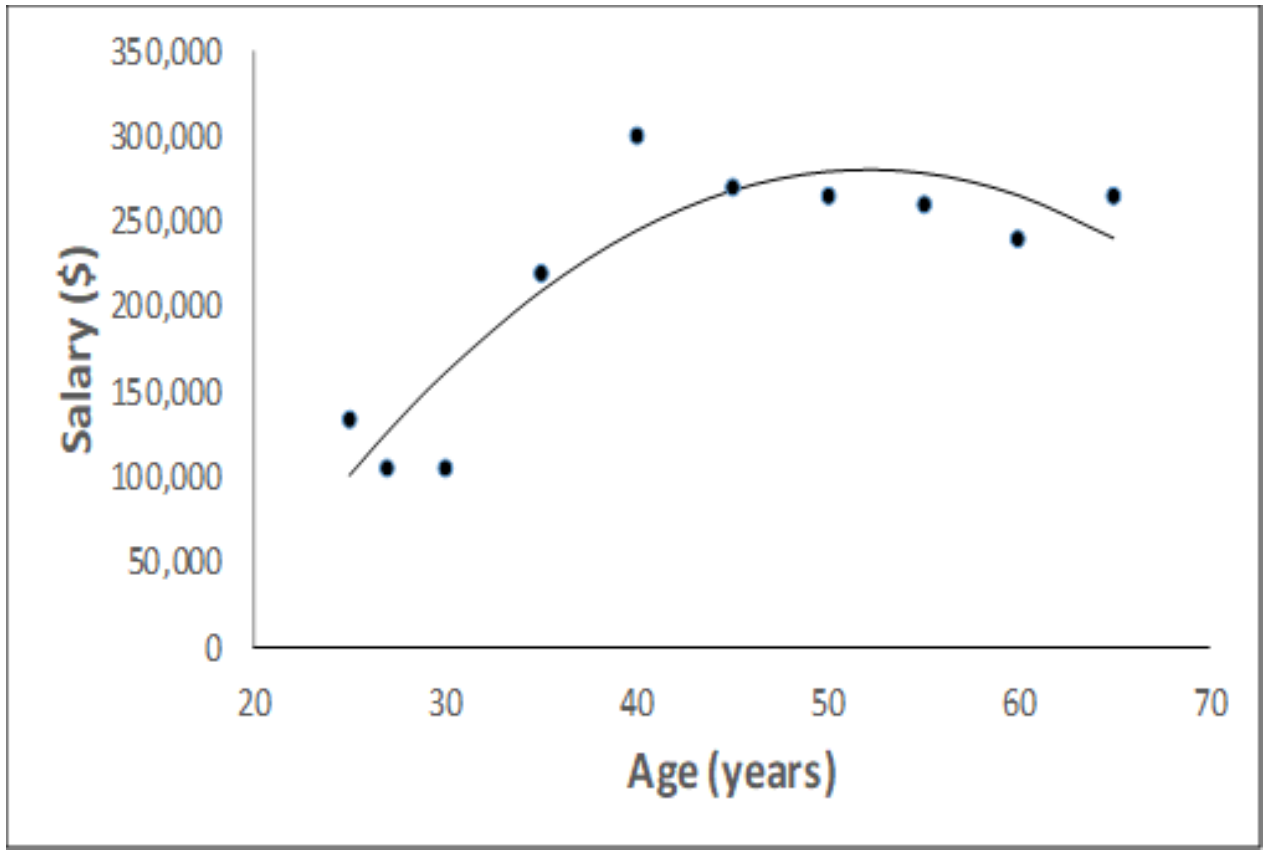
- The root mean squared error (rmse) for the training data set is $12,902
- The rmse for the test data set is $38,794
- We conclude that the model overfits the data



Validation and Testing

Figure 4. A Simpler Quadratic Model

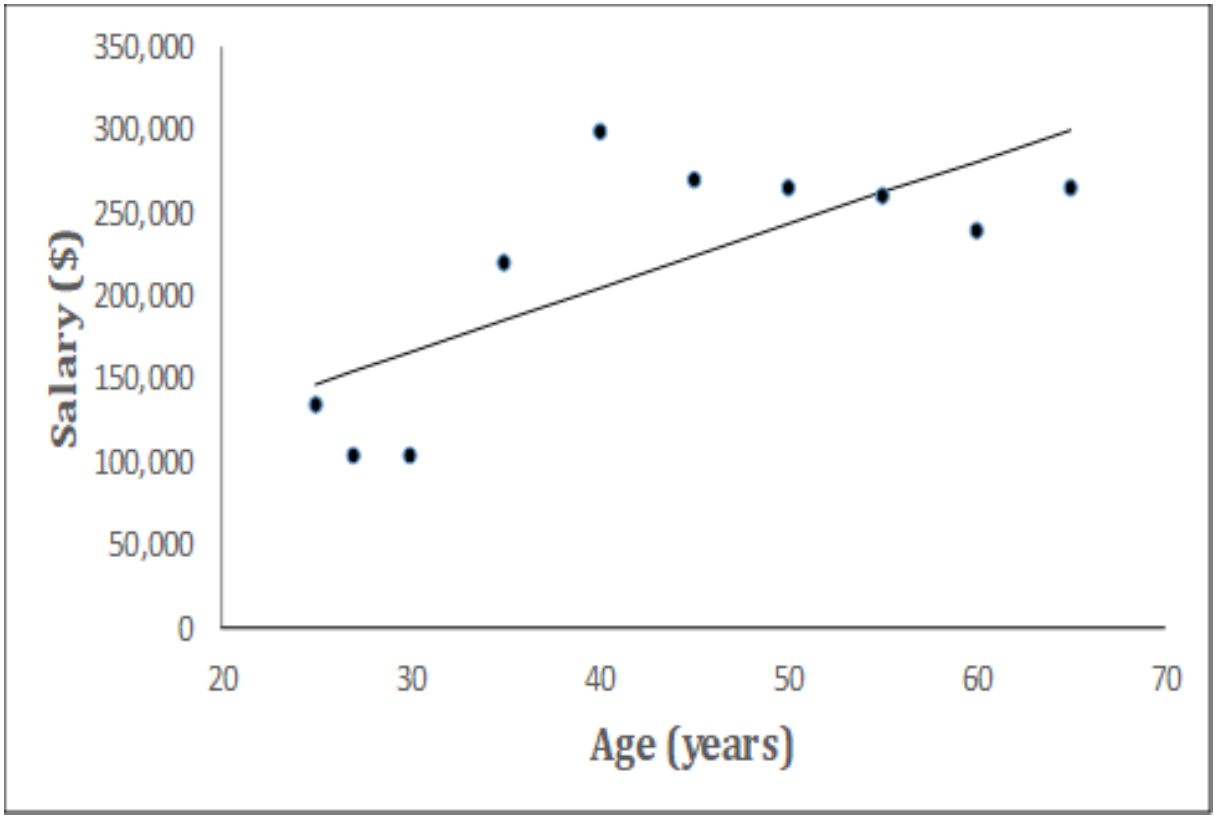
$$Y = a + b_1 X + b_2 X^2$$



Validation and Testing

Figure 5 . Linear Model

$$Y = a + b_1 X$$



Validation and Testing

Table 3. Summary of Results: The linear model under-fits while the 5th degree polynomial over-fits:

	Polynomial of degree 5	Quadratic model	Linear model
Training set	12, 902	32,932	49,731
Validation set	38,794	33,554	49,990

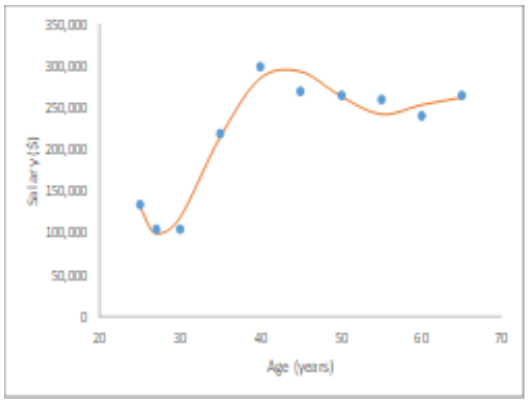
Validation and Testing

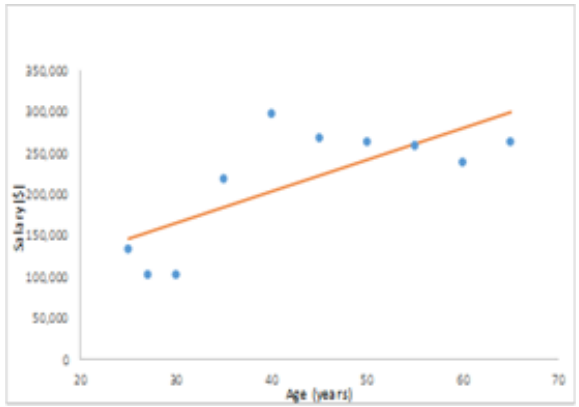
Table 3. Summary of Results: The linear model under-fits while the 5th degree polynomial over-fits:

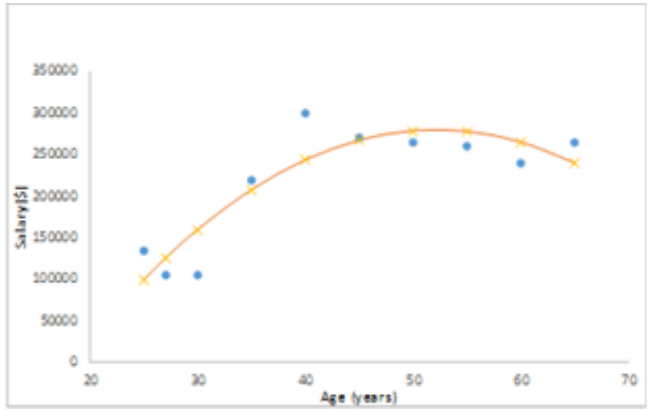
	Polynomial of degree 5	Quadratic model	Linear model
Training set	12,902	32,932	49,781
Validation set	38,794	33,554	49,990

Validation and Testing

Figure 6. Overfitting/Underfitting Example: predicting salaries for people in a certain profession in a certain area (only 10 observations)

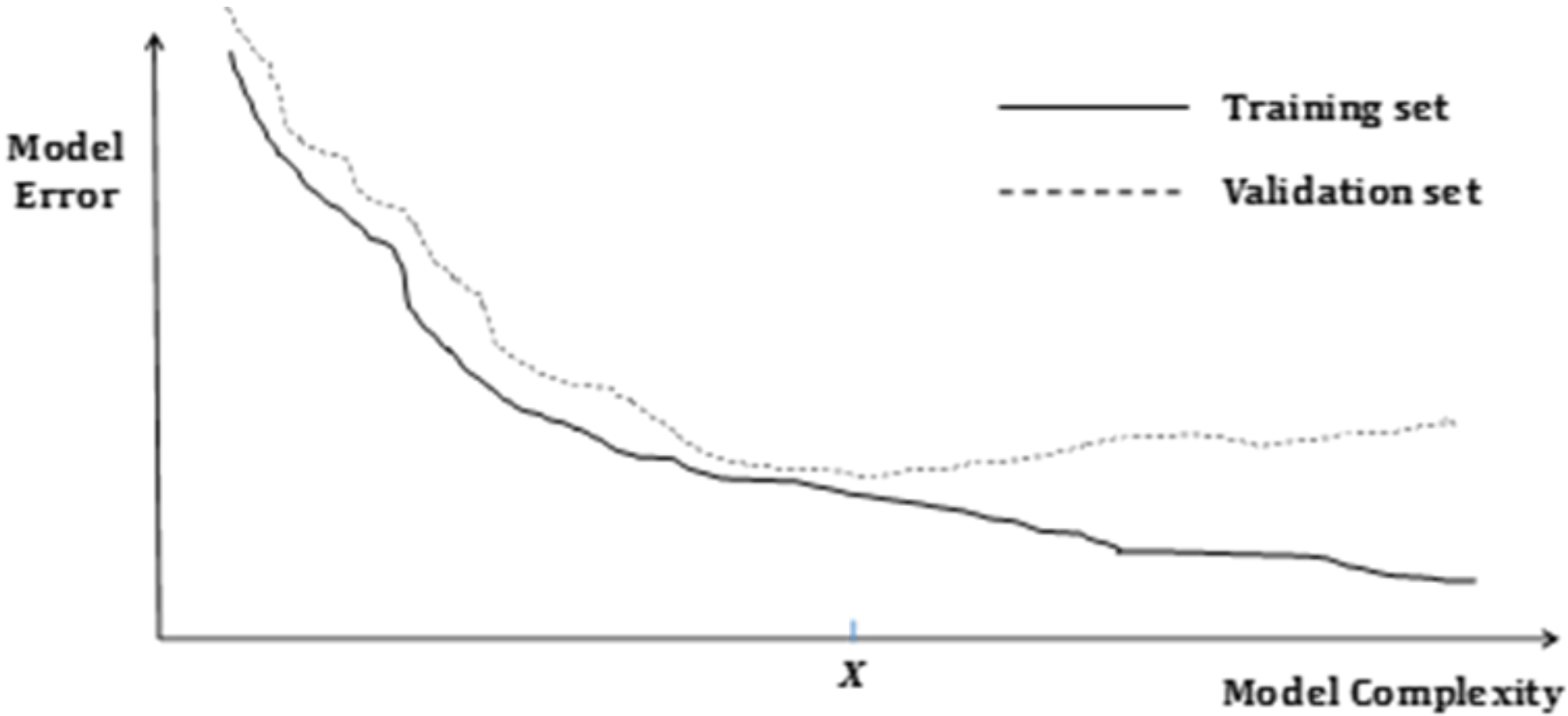






Overfitting ——— Underfitting ——— Best model?

Typical Pattern of Errors for Training Set and Validation Set



Validation and Testing

ML Good Practice

- Divide data into three sets
- Training set
- Validation set
- Test set
- Develop different models using the **training set** and compare them using the **validation set**;
- Rule of thumb: increase model complexity until model no longer generalizes well to the validation set;
- The **test set** is used to provide a final out-of-sample indication of how well the chosen model works;

Bias and Variance

- Suppose there is a relationship between an independent variable x and a dependent variable y:

$$y = f(x) + \epsilon \tag{4}$$

  where ϵ is an error term with mean zero and variance σ^2.
- The error term captures either genuine randomness in the data or noise due to measurement error.
- Suppose we find, with a Machine Learning technique, a deterministic model for this relationship:

$$y = \hat{f}(x) \tag{5}$$

Bias and Variance

- Now it comes a new data point x' not in the training set and we want to predict the corresponding y';
- The error we will observe in our model at point x' is going to be

$$\hat{f}(x') - f(x') - \epsilon \tag{6}$$

- There are two different sources of error in this equation.
- The first one is included in the factor ϵ;
- The second one, more interesting, is due to what is in our training set.
- A robust model should give us the same prediction whatever data we used for training out model.

Bias and Variance

- Let's look at the average error:

$$E\left[\hat{f}(x')\right] - f(x') \tag{7}$$

where the expectation is taken over random samples of training data (having the same distributio as the training data).

- This is the definition of the **bias**

$$\text{Bias}\left[\hat{f}(x')\right] = E\left[\hat{f}(x')\right] - f(x') \tag{8}$$

Bias and Variance

- We can also look at the mean square error

$$E\left[\left(\hat{f}(x') - f(x') - \epsilon\right)^2\right] = \left[\text{Bias}\left(\hat{f}(x')\right)\right]^2 + \text{Var}\left[\hat{f}(x')\right] + \sigma^2$$

  where we remember that $\hat{f}(x')$ and ϵ are independent.
- This show us that there are two important quantities, the **bias** and the **variance** that will affect our results and that we can control to some extent.

Bias and Variance

- **What is Bias?** It's the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data.
- **What is Variance?** It's the variability of model prediction for a given data point or a value which tells us spread of our data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before. As a result, such models perform very well on training data but has high error rates on test data.

Bias and Variance

Low Bias and Low Variance

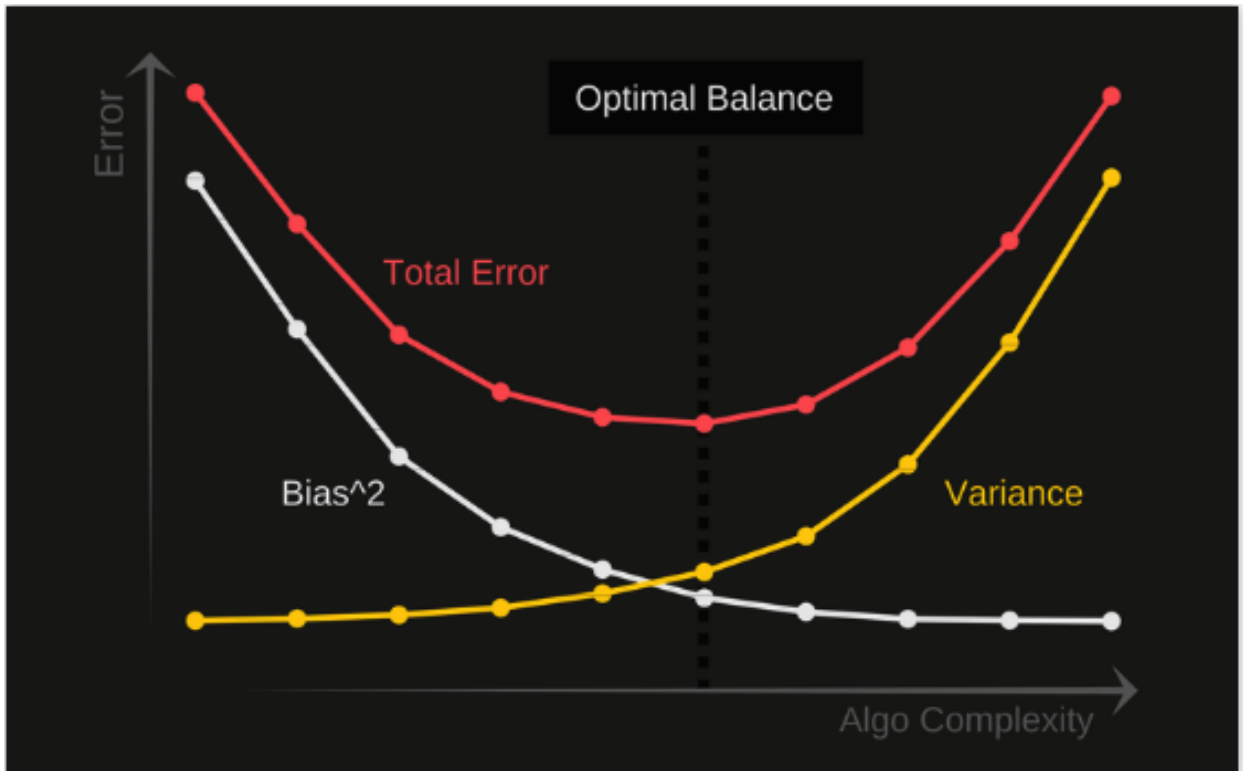
High Bias and Low Variance

Low Bias and High Variance

High Bias and High Variance

Bias and Variance

Unfortunately, we often find that there is a trade-off between bias and variance. As one is reduced, the other is increased. This is the matter of over- and under-fitting.



Regularization

- Linear regression can over-fit, particularly when there are a large number of correlated features.
- Results for validation set may not then be as good as for training set
- Regularization is a way of avoiding overfitting and reducing the number of features. Alternatives:
- Ridge
- Lasso
- Elastic net
- We must first scale feature values

Ridge regression (analytic solution)

- Ridge regression is a regularization technique where we change the function that is to be minimize;
- Reduce magnitude of regression coefficients by choosing a parameter λ and minimizing

$$\frac{1}{2N}\sum_{n=1}^{N}\left[h_\theta\left(x^{(n)}\right)-y^{(n)}\right]^2+\lambda\sum_{n=1}^{N}b_i^2 \tag{9}$$

- This change has the effect of encouraging the model to keep the weights b_j as small as possibile;
- The Ridge regression should only be used for determining model parameters using the training set. Once the model parameters have been determined the penalty term should be removed for prediction;
- What happens as λ increases?

Lasso Regression (must use gradient descent)

- Lasso is short for *Least Absolute Shrinkage and Selection Operator*;
- Similar to ridge regression except we minimize

$$\frac{1}{2N}\sum_{n=1}^{N}\left[h_\theta\left(x^{(n)}\right)-y^{(n)}\right]^2+\lambda\sum_{n=1}^{N}|b_n| \tag{10}$$

- This function cannot be minimized analytically and so a variation on the gradient descent algorithm must be used;
- Lasso regression also has the effect of simplifying the model. It does this by setting the weights of unimportant features to zero. When there are a large number of features, Lasso can identify a relatively small subset of the features that form a good predictive model.

Elastic Net Regression (must use gradient descent)

- Middle ground between Ridge and Lasso
- Minimize

$$\frac{1}{2N}\sum_{n=1}^{N}\left[h_\theta\left(x^{(n)}\right)-y^{(n)}\right]^2+\lambda_1\sum_{n=1}^{N}b_n^2+\lambda_2\sum_{n=1}^{N}|b_n| \tag{11}$$

- In Lasso some weights are reduced to zero but others may be quite large. In Ridge, weights are small in magnitude but they are not reduced to zero. The idea underlying Elastic Net is that we may be able to get the best of both by making some weights zero while reducing the magnitude of the others.

Salary Vs Age Example: The Effect of Regularization

Age (years)	Salary ($)
25	135,000
55	260,000
27	105,000
35	220,000
60	240,000
65	265,000
45	270,000
40	300,000
50	265,000
30	105,000

We apply regularization to the model:

$$Y = a + b_1 X + b_2 X^2 + b_3 X^3 + b_4 X^4 + b_5 X^5 \tag{12}$$

where Y is salary and X is age.

Salary Vs Age Example: The Effect of Regularization

Data with Z-score scaling

Observ.	X	X^2	X^3	X^4	X^5
1	−1.290	−1.128	−0.988	−0.874	−0.782
2	0.836	0.778	0.693	0.592	0.486
3	−1.148	−1.046	−0.943	−0.850	−0.770
4	−0.581	−0.652	−0.684	−0.688	−0.672
5	1.191	1.235	1.247	1.230	1.191
6	1.545	1.731	1.901	2.048	2.174
7	0.128	−0.016	−0.146	−0.253	−0.333
8	−0.227	−0.354	−0.449	−0.511	−0.544
9	0.482	0.361	0.232	0.107	−0.004
10	−0.936	−0.910	−0.861	−0.803	−0.745

Salary Vs Age Example: The Effect of Regularization

Ridge Results, $\lambda = 0.02$ is similar to quadratic model

λ	a	b_1	b_2	b_3	b_4	b_5
0	216.5	−32,623	135,403	−215,493	155,315	−42,559
0.02	216.5	97.8	36.6	−8.5	35.0	−44.6
0.10	216.5	56.5	28.1	3.7	−15.1	−28.4

Salary Vs Age Example: The Effect of Regularization

Lasso Results, $\lambda = 1$ is similar to the quadratic model

λ	a	b_1	b_2	b_3	b_4	b_5
0	216.5	−32,623	135,403	−215,493	155,315	−42,559
0.02	216.5	−646.4	2,046.6	0.0	−3,351.0	2,007.9
0.1	216.5	355.4	0.0	−494.8	0.0	196.5
1	216.5	147.4	0.0	0.0	−99.3	0.0

Let's code ...

Outline

Unsupervised Learning

- In unsupervised learning we are not trying to predict anything
- The objective is to cluster data to increase our understanding of the environment
- **Example - Clustering Customers**
  - Suppose you are a bank and have hundreds of thousands of customers and 100 features describing each one
  - Unsupervised learning algorithms can be used to divide your customers into clusters so that you can anticipate their needs and communicate with them more effectively

The *k*-Means Algorithm

- In this section we explain a simple clustering procedure known as the *k-means algorithm*;
- *k*-means clustering is one of the simplest and popular unsupervised machine learning algorithms.
- Typically, unsupervised algorithms make inferences from datasets using only input vectors without referring to known, or labelled, outcomes.
- The objective of K-means is simple: group similar data points together and discover underlying patterns.
- To achieve this objective, K-means looks for a fixed number (k) of clusters in a dataset.

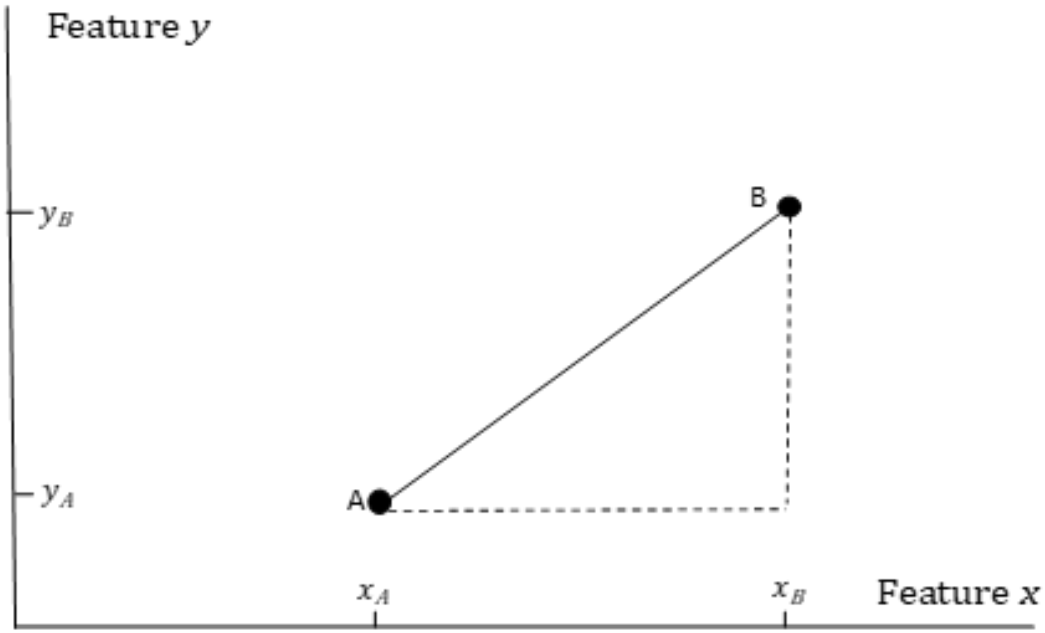
The *k*-Means Algorithm

- A cluster refers to a collection of data points aggregated together because of certain similarities.
- You'll define a target number k, which refers to the number of centroids you need in the dataset.
- A centroid is the imaginary or real location representing the center of the cluster.
- Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares.
- In other words, the K-means algorithm identifies k number of centroids, and then allocates every data point to the **nearest** cluster, while keeping the centroids as small as possible.
- The 'means' in the K-means refers to averaging of the data; that is, finding the centroid.

The k-Means Algorithm

A Distance Measure

- For clustering we need a distance measure
- The simplest distance measure is the Euclidean distance measure.
  Distance $= \sqrt{(x_B - x_B)^2 + (y_B - y_A)^2}$



The k-Means Algorithm

Distance Measure

- In general when there are m features the distance between P and Q is

$$d = \sqrt{\sum_{j=1}^{m} (\nu_{pj} - \nu_{qj})^2} \tag{13}$$

where ν_{pj} and ν_{qj} are the values of the $j-th$ feature for P and Q

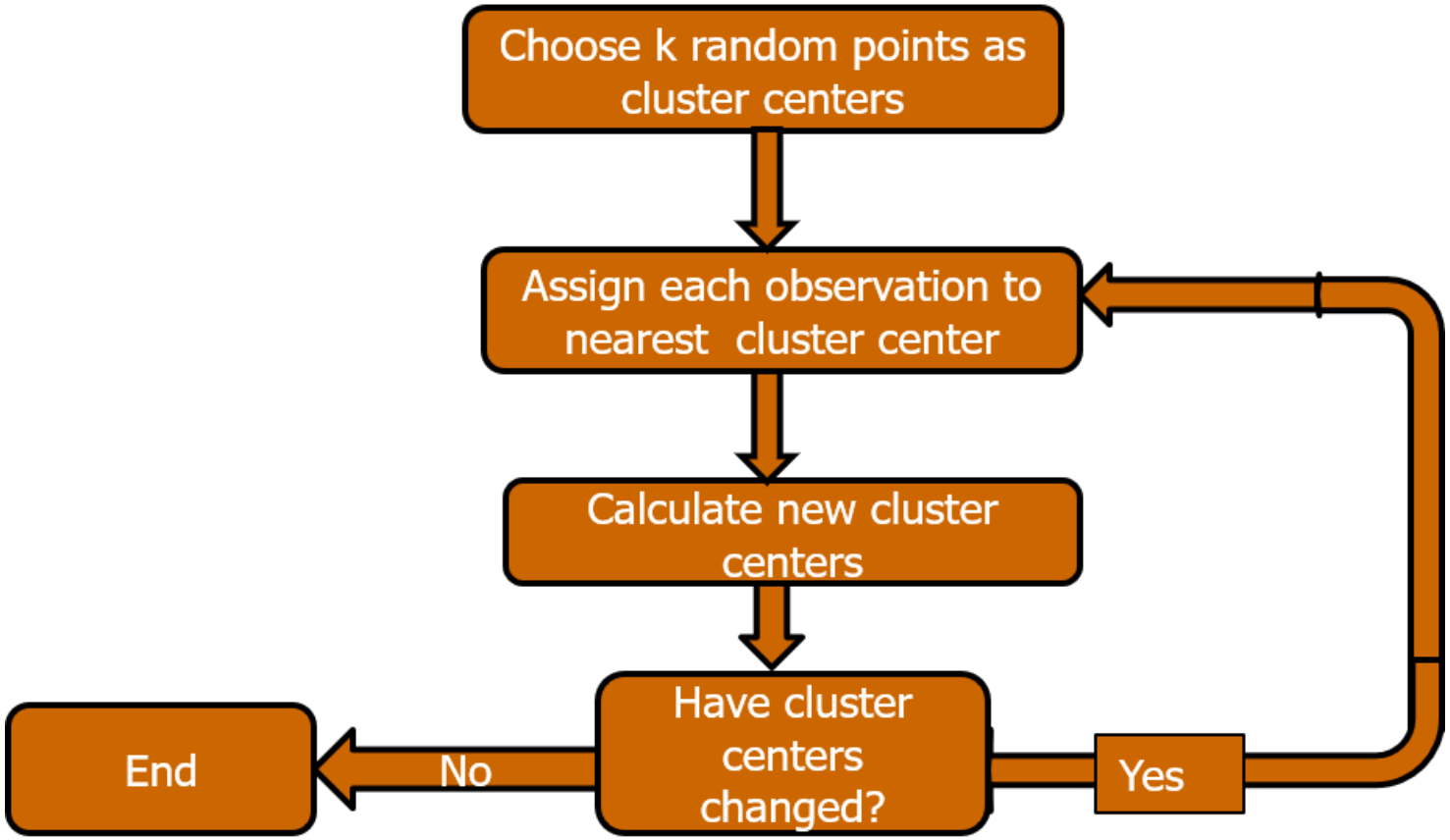
The k-Means Algorithm

Cluster Centers

The center of a cluster (sometimes called the **centroid**) is determined by averaging the values of each feature for all points in the cluster.
Example:

Observ.	Feature 1	Feature 2	Feature 3	Feature 4	Distance to center
1	1.00	1.00	0.40	0.25	0.145
2	0.80	1.20	0.25	0.40	0.258
3	0.82	1.05	0.35	0.50	0.206
4	1.10	0.80	0.21	0.23	0.303
5	0.85	0.90	0.37	0.27	0.137
Center	0.914	0.990	0.316	0.330	

The k-Means Algorithm to find k Clusters



The k-Means Algorithm

Cluster Centers

Inertia

- A measure of the performance of the algorithm is the within cluster sum of squares also known as *inertia*;
- For any given k the objective is to minimize inertia:

$$Inertia = \sum_{i=1}^{n} d_i^2 \tag{14}$$

where d_i is the distance of observation i from its cluster center

- In practice we use the k-means algorithm with several different starting points and choose the result that has the smallest inertia

The k-Means Algorithm

Choosing k

- The elbow approach (see next slide)
- The silhouette method:
  - For each observation i calculate $a(i)$, the average distance from other observations in its cluster, and $b(i)$, the average distance from observations in the closest other cluster. The silhouette score for observation i, $s(i)$, is defined as

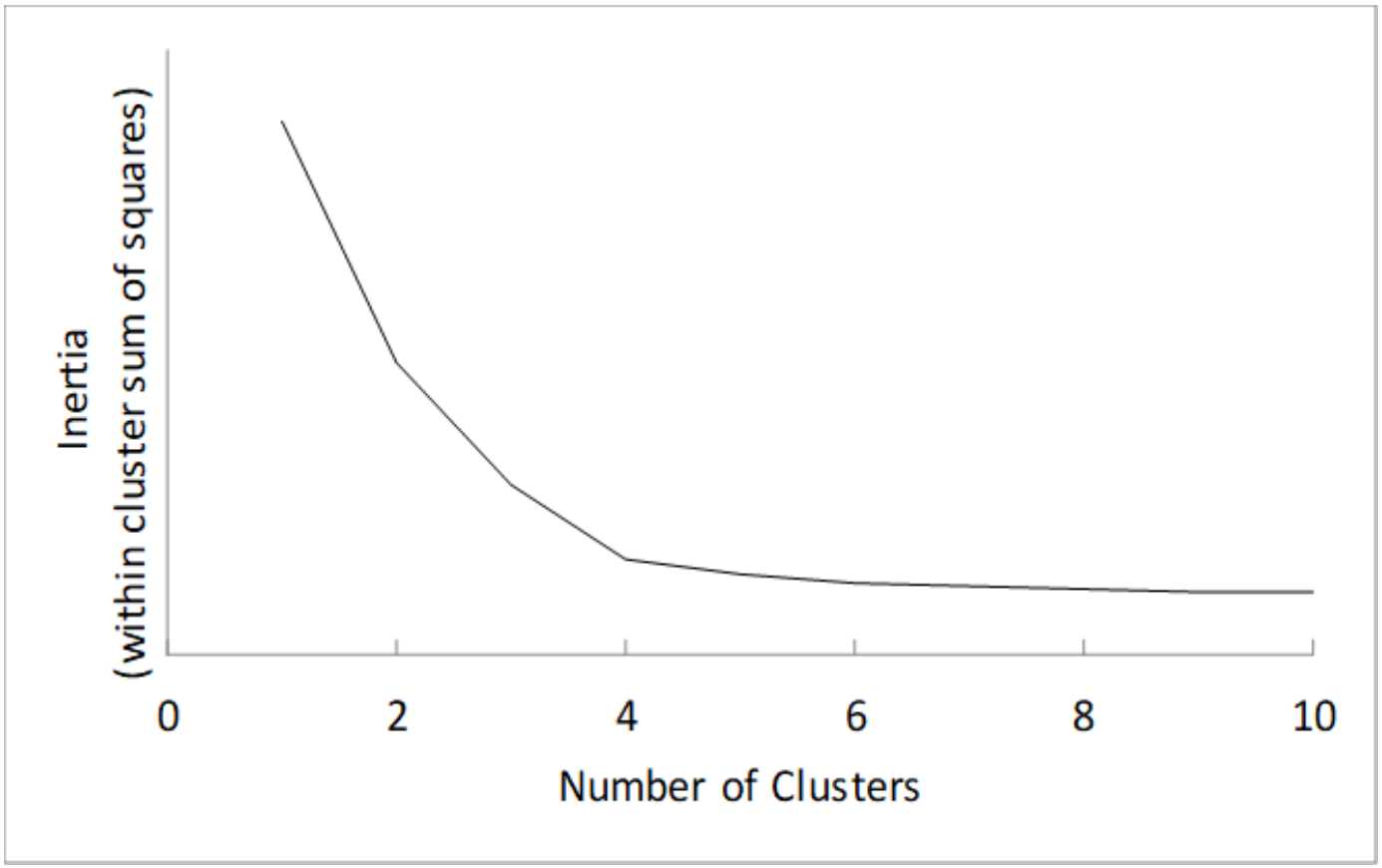
$$s(i) = \frac{b(i) - a(i)}{\max[a(i), b(i)]} \tag{15}$$

  - Choose the number of clusters that maximizes the average silhouette score across all observations
- Use the gap statistic which compares the within cluster sum of squares with what would be expected with random data

The k-Means Algorithm

The elbow method

The **elbow method** (In this example k=4 is suggested)



The k-Means Algorithm

The Curse of Dimensionality

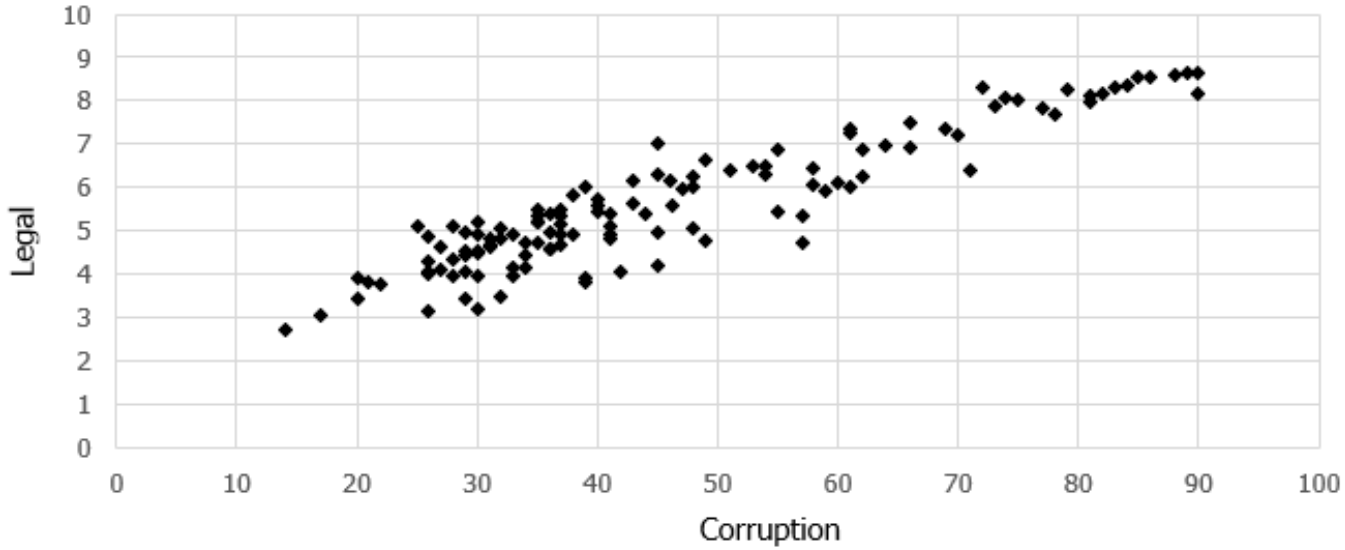
- The Euclidean distance measure increases as the number of features increase.
- This is referred to as the curse of dimensionality
- Consider two observations that have values for feature j equal to x_j and y_j. An alternative distance measure that always lies between 0 and 2 is

$$d = 1 - \frac{\sum_{j=1}^{m} x_j y_j}{\sqrt{\sum_{j=1}^{m} x_j^2 \sum_{j=1}^{m} y_j^2}} \tag{16}$$

Country Risk Case

- Objective is to cluster countries according to their riskiness for foreign investment
- Measures of Country Risk
- GDP growth rate (IMF)
- Corruption index (Transparency international)
- Peace index (Institute for Economics and Peace)
- Legal Risk Index (Property Rights Association)
- Collected data on 122 countries. Used Z-score scaling.

Country Risk Case

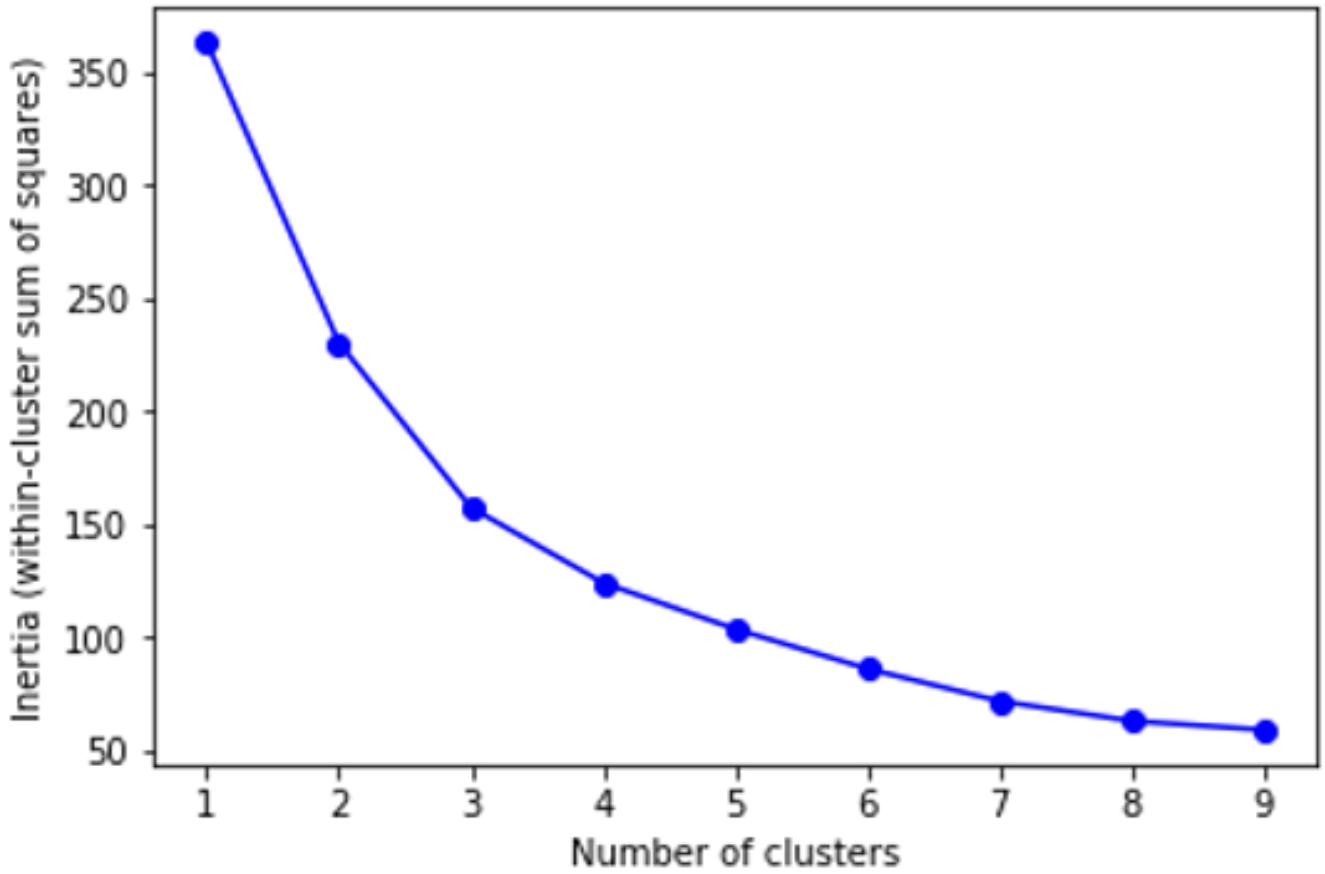


Corruption and legal risk were highly correlated therefore analysis based on

- GDP growth rate
- Peace index
- Legal risk index

Country Risk Case

How the total within-cluster sum of squares declines as k increases when k-means algorithm is used



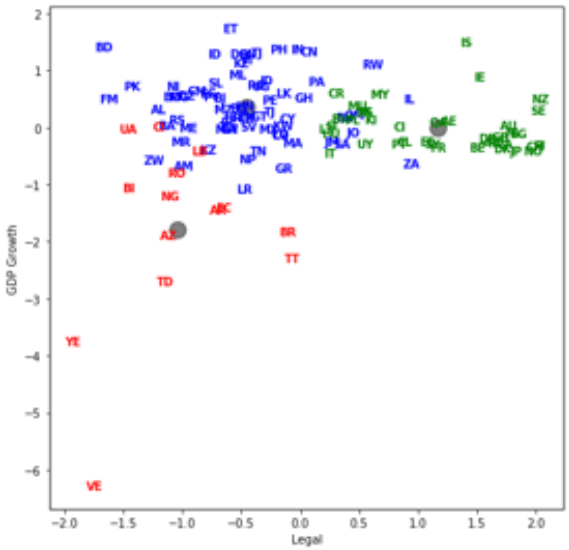
Country Risk Case

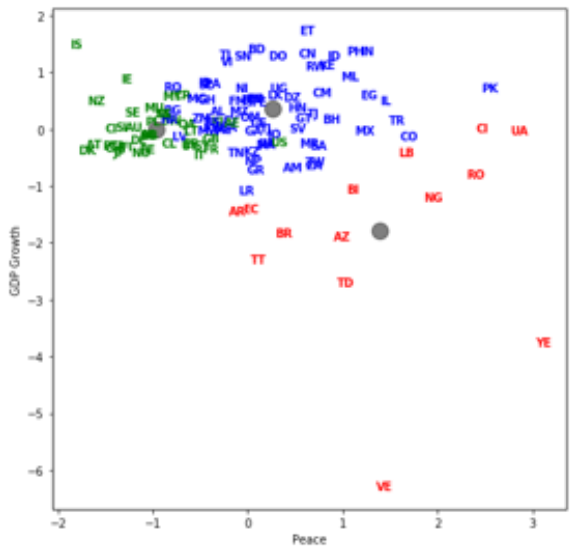
Silhouette scores suggest k=3

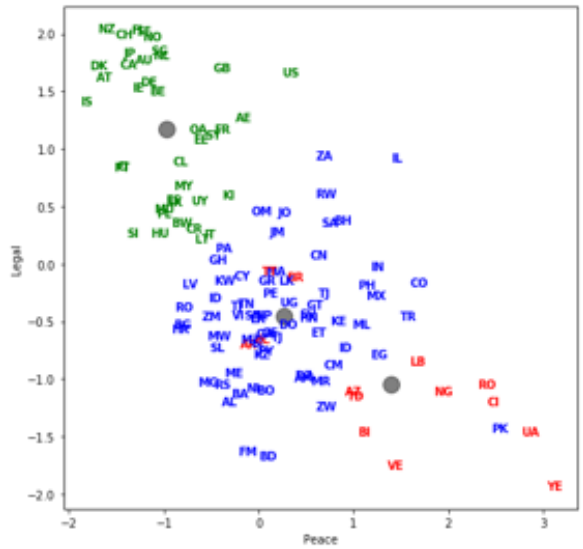
Number of clusters	Average silhouette score
2	0.363
3	0.388
4	0.370
5	0.309
6	0.303
7	0.315
8	0.321
9	0.292
10	0.305

Country Risk Case

The three-cluster results: Green = Low country risk, Blue = Medium country risk, Red = High country risk







Country Risk Case

Cluster centers (scaled values). Note that high values for the peace index are bad whereas high values for the legal risk index are good.

	Peace index	Legal index	GDP
High risk	1.39	−1.04	−1.79
Moderate risk	0.27	−0.45	0.36
Low risk	−0.97	1.17	0.00

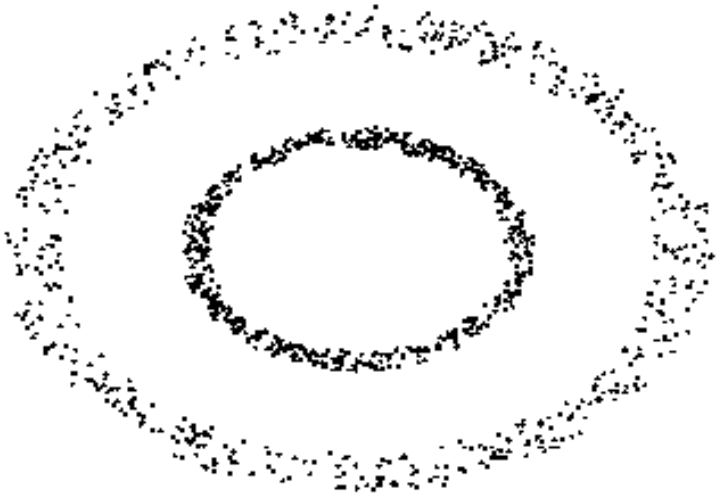
Hierarchical Clustering

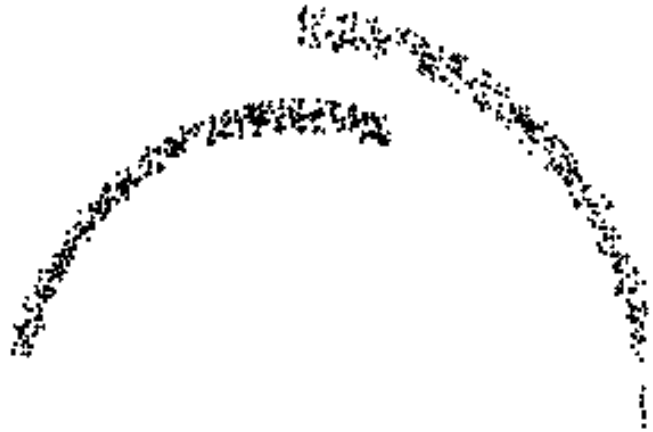
- Start with each observation in its own cluster
- Combine the two closest clusters
- Continue until all observations have been combined into a single cluster
- Can be implemented in Python with AgglomerativeClustering.
- Measures of closeness of clusters:
- Average Euclidean distance between points in clusters
- Maximum distance between points in clusters
- Minimum distance between points in clusters
- Increase in inertia (a version of Ward's method)

Density-based clustering

- Forms clusters based on the closeness of individual observations
- Unlike k-means the algorithm, it is not based on cluster centers.
- We might initially choose 8 observations that are close. After that we add an observation to the cluster if it is close to at least 5 other observations in the cluster, and repeat.

Density-based Clustering Examples





Outline

Linear Regression

- Linear regression is a very popular tool because once you have made the assumption that the model is linear you do not need huge amount of data. In ML we refer to the constant term as the *bias* and the coefficients as *weights*
- Assume n observations and m features. Model is

$$Y = a + b_1 X_1 + b_2 X_2 + \cdots + b_m X_m + \epsilon$$

- Standard approach is to choose a and the b_i to minimize the mean square error (mse):

$$mse = \frac{1}{n} \sum_{j=1}^{n} [Y_j - (a + b_1 X_{1,j} + b_2 X_{2,j} + \cdots + b_m X_{m,j})]^2 \qquad (17)$$

- This can be done analytically by inverting a matrix, in practice a numerical (gradient descent) is used.

Categorical Features

- Categorical features are features where there are a number of non-numerical alternatives
- We can define a dummy variable for each alternative. The variable equals 1 if the alternative is true and zero otherwise. This is known as **one-hot encoding**
- But sometimes we do not have to do this because there is a natural ordering of variables, e.g.:
  - small=1, medium=2, large=3
  - assist. prof=1, assoc. prof=2, full prof =3

Dummy Variably Trap

- Suppose we have a constant term and a number of dummy variables (equal to 0 or 1)
- There is then no unique solution because, for any C, we can add C to the constant term and subtract C from each of the dummy variables without changing the prediction
- A side effect of regularization is that it solves this problem

Iowa House Price Case Study

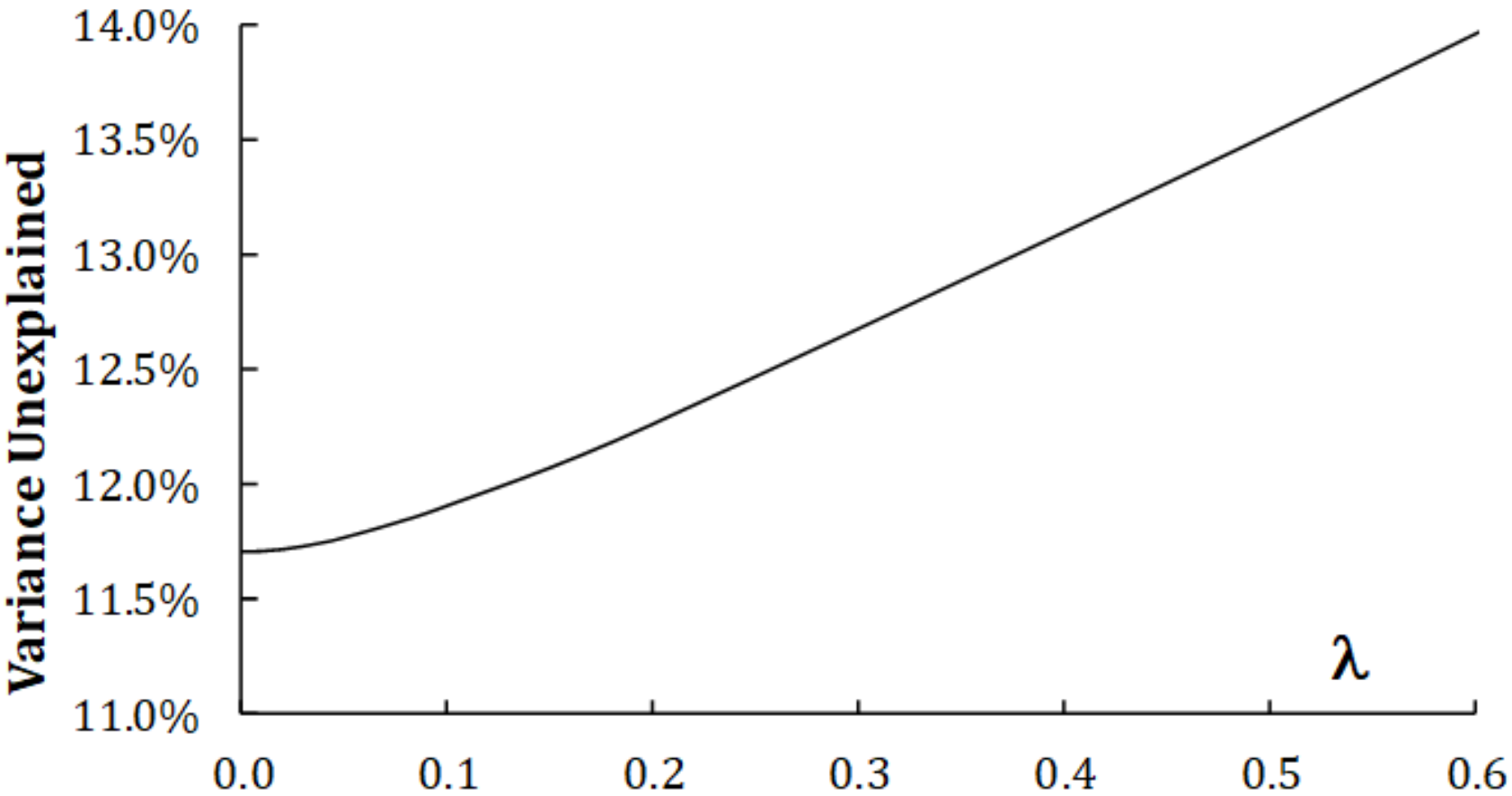
- The objective is to predict the prices of house in Iowa from features
- 800 observations in training set, 600 in validation set, and 508 in test set
- Here the original competition description: **https://www.kaggle.com/c/house-prices-advanced-regression-techniques**

Iowa House Price Results (No regularization)

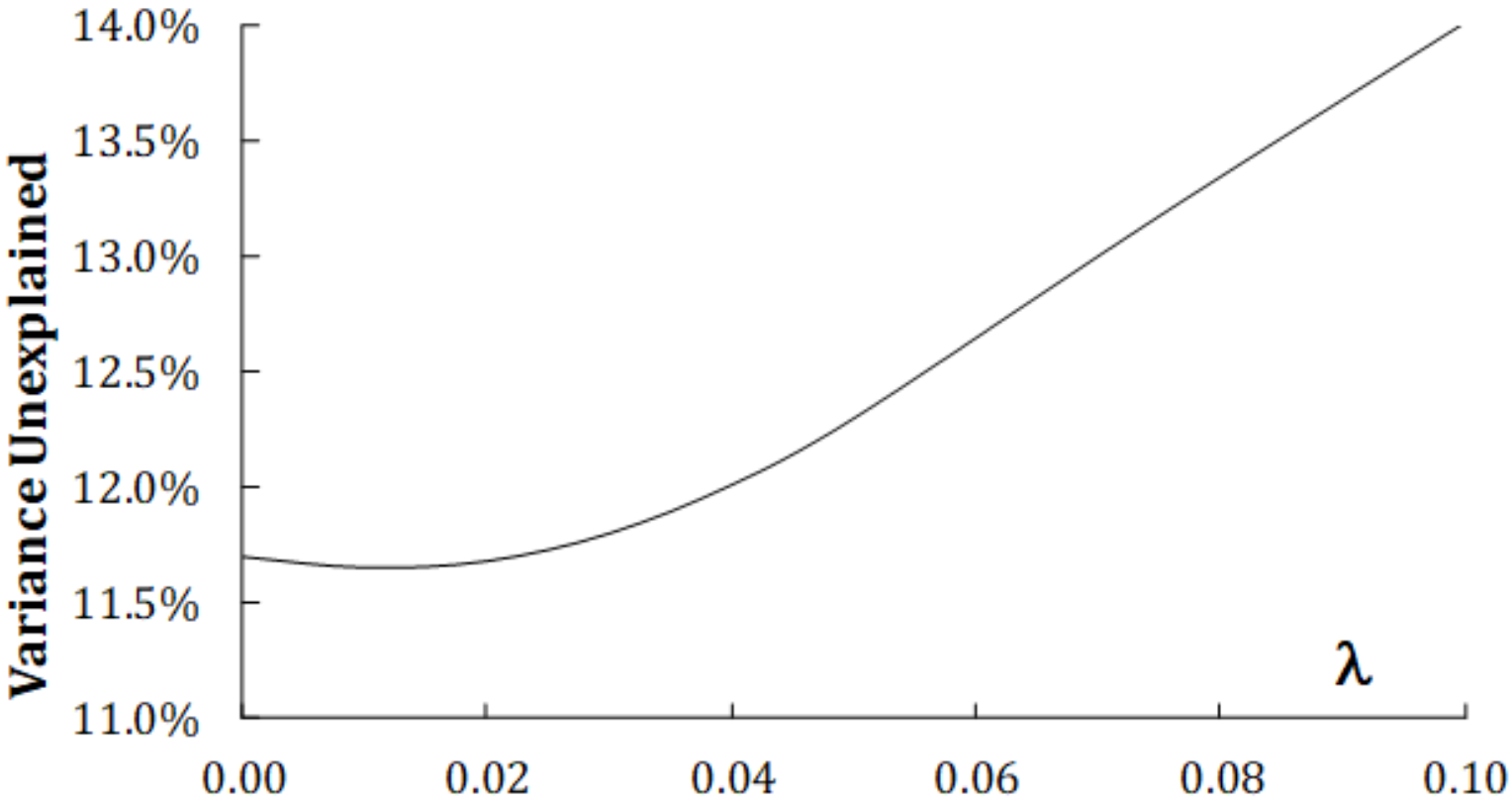
2 categorical variables included. Natural ordering for Basement quality. 25 dummy variables created for neighborhood

Lot area (squ ft)	0.08	Number of half bathrooms	0.02
Overall quality (scale from 1 to 10)	0.21	Number of bedrooms	−0.08
Overall condition (scale from 1 to 10)	0.10	Total rooms above grade	0.08
Year built	0.16	Number of fireplaces	0.03
Year remodeled	0.03	Parking spaces in garage	0.04
Basement finished squ ft	0.09	Garage area (squ ft)	0.05
Basement unfinished squ ft	−0.03	Wood deck (squ ft)	0.02
Total basement squ ft	0.14	Open porch (squ ft)	0.03
1st floor squ ft	0.15	Enclosed porch (squ ft0	0.01
2^{nd} floor squ ft	0.13	Neighborhood (25 alternatives)	−0.05 to 0.12
Living area	0.16	Basement quality (6 natural ordering)	0.01
Number of full bathrooms	−0.02		

Ridge Results for validation set



Lasso Results for validation set



Iowa House Price Results

Non-zero weights for Lasso when $\lambda = 0.1$ (overall quality and total living area were most important)

Feature	Weight
Lot Area (square feet)	0.04
Overall quality (Scale from 1 to 10)	0.30
Year built	0.05
Year remodeled	0.06
Finished basement (square feet)	0.12
Total basement (square feet)	0.10
First floor (square feet)	0.03
Living area (square feet)	0.30
Number of fireplaces	0.02
Parking spaces in garage	0.03
Garage area (square feet)	0.07
Neighborhoods (3 out of 25 non-zero)	0.01, 0.02, and 0.08
Basement quality	0.02

Summary of Iowa House Price Results

- With no regularization correlation between features leads to some negative weights which we would expect to be positive
- Improvements from Ridge is modest
- Lasso leads to a much bigger improvement in this case
- Elastic net similar to Lasso in this case
- Mean squared error for test set for Lasso with $\lambda = 0.1$ is 14.7% so that 85.3% of variance is explained

Bibliography

- John C. Hull, *Machine Learning in Business: An Introduction to the World of Data Science*, Amazon, 2019.
- Paul Wilmott, *Machine Learning: An Applied Mathematics Introduction*, Panda Ohana Publishing, 2019.