

— An overview of Linear Regression

— Building supervised models

What do we need?

- A problem statement
- A dataset
- A choice of independent variables
- A choice of model
- A way to measure how good our model is

Problem statement

For supervised learning this needs to be in the format:

Predicting y using X

Our two problem statements for today are:

1. *(Main problem)*

Predicting total ridership of Capital Bikeshare in any given hour

2. *(Secondary, used for the explanation of Linear Regression)*

Predicting total ridership of Capital Bikeshare given the temperature


Dataset

9th

Playground Prediction Competition

Bike Sharing Demand

Forecast use of a city bikeshare system

 Kaggle · 3,242 teams · 6 years ago

Overview

Data

Code

Discussion

Leaderboard

Rules

Join Competition

...

Overview





Description

Evaluation

Get started on this competition through [Kaggle Scripts](#)

Bike sharing systems are a means of renting bicycles where the process of obtaining membership, rental, and bike return is automated via a network of kiosk locations throughout a city. Using these systems, people are able rent a bike from a one location and return it to a different place on an as-needed basis. Currently, there are over 500 bike-sharing programs around the world.

The data generated by these systems makes them attractive for researchers because the duration of travel, departure location, arrival location, and time elapsed is explicitly recorded. Bike sharing systems therefore function as a sensor network, which can be used for studying mobility in a city. In this competition, participants are asked to combine historical usage patterns with weather data in order to forecast bike rental demand in the Capital Bikeshare program in Washington, D.C.



Acknowledgements

Kaggle is hosting this competition for the machine learning community to use for fun and practice. This dataset was provided by Hadi Fanaee Tork using data from [Capital Bikeshare](#). We also thank the UCI machine learning repository for [hosting the dataset](#). If you use the problem in publication, please cite:

Fanaee-T, Hadi, and Gama, Joao, *Event labeling combining ensemble detectors and background knowledge*, Progress in Artificial Intelligence (2013): pp. 1-15, Springer Berlin Heidelberg.

Independent variable choice

These come from your exploratory data analysis (EDA) and will be chosen through looking at:

- Scatter plots
- Histograms
- Correlation coefficients
- Missing values

For classification tasks, we also need to check the balance of classes in our target variable.



Model choice

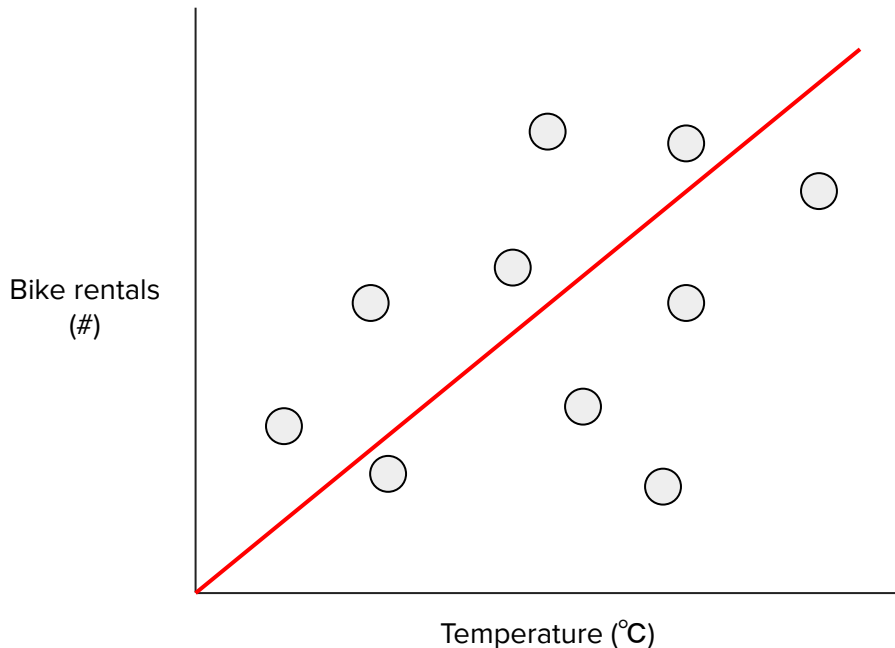
This will depend on:

- Whether the task is regression or classification
- How much training data is available
- Domain expertise

Simple, explainable models are often the best choice.

— Linear Regression

Supervised Machine Learning



Initial Problem: predict bike rentals using temperature

This is a **supervised learning** task because it fits the pattern of 'predict **y** using **X**'.

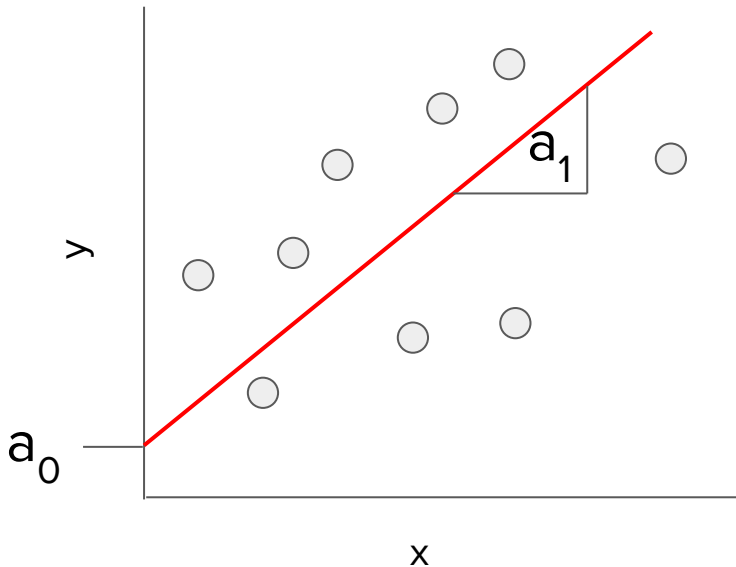
We have **training data** () which gives us paired values of bike rentals per hour for almost 2 years, and the temperature during each hour.

We **fit a model** to this data. In this case, it's a very simple straight line (**linear**) model.

We can use the **trained model** (the red line) to predict the number of rentals for a new datapoint that **isn't** in our training dataset, as long as we know the temperatures.

Linear regression

Modelling the relationship between some **features** and a **continuous response variable** using a **linear equation**.



$$y = a_0 + a_1x$$

Linear regression

Where are the independent and dependent variables?

What do the coefficients and constants mean?

We can have one feature...

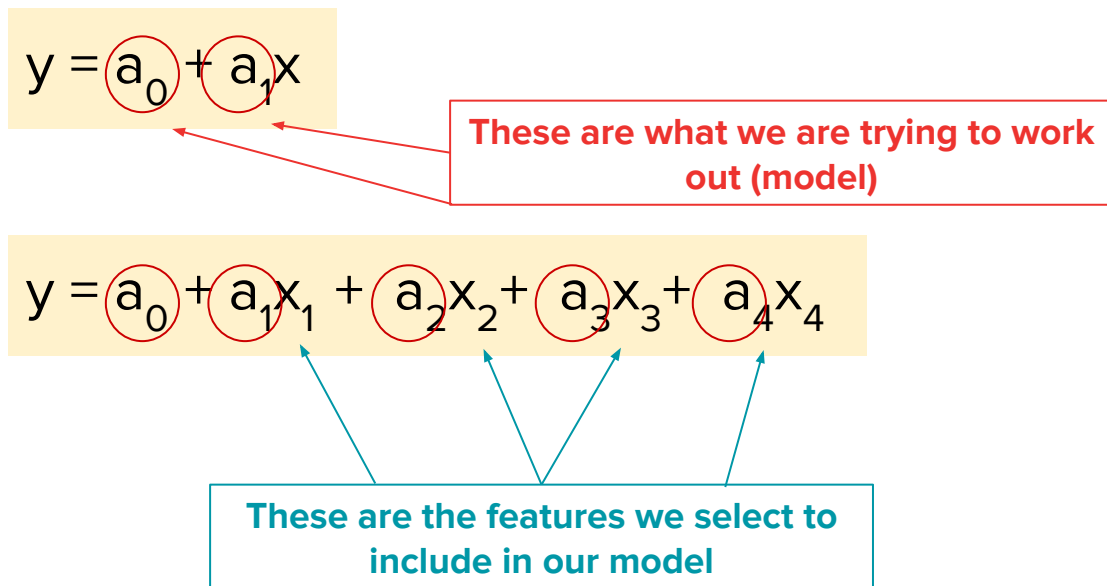
$$y = a_0 + a_1x$$

Or multiple features...

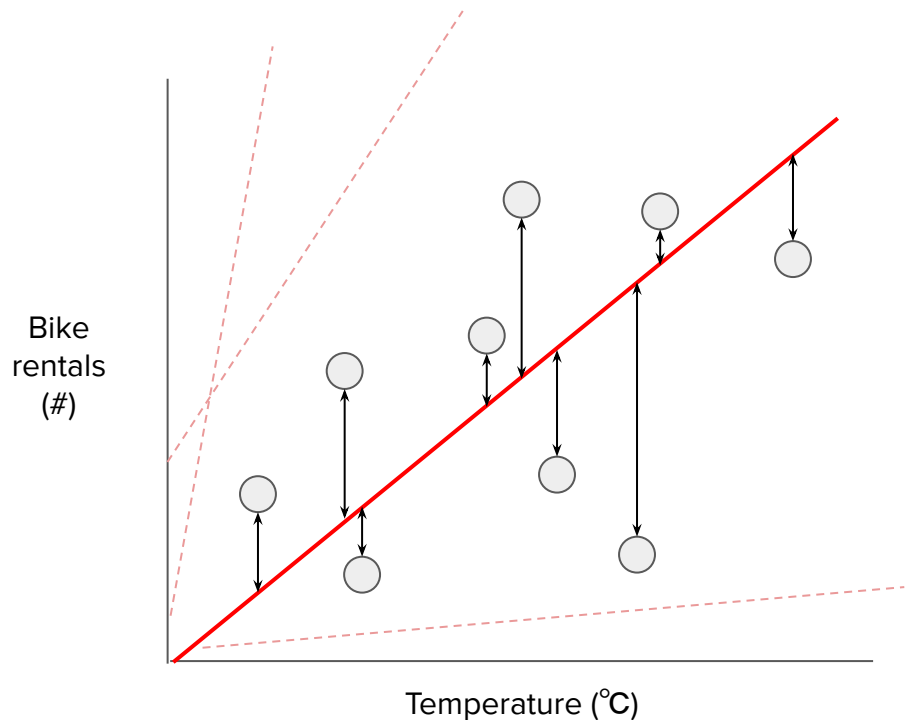
$$y = a_0 + a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4$$

Linear regression

The process of fitting or training a linear regression model is the process of **finding these coefficients** or in simple cases, finding the 'line of best fit'



Fitting the model



The line of best fit is one that minimises the distance between the dataset and our model.

So the process of fitting a linear regression model is about finding the coefficients that give us a straight line that passes as close to the points in our dataset as possible.

The original Linear Regression dataset...

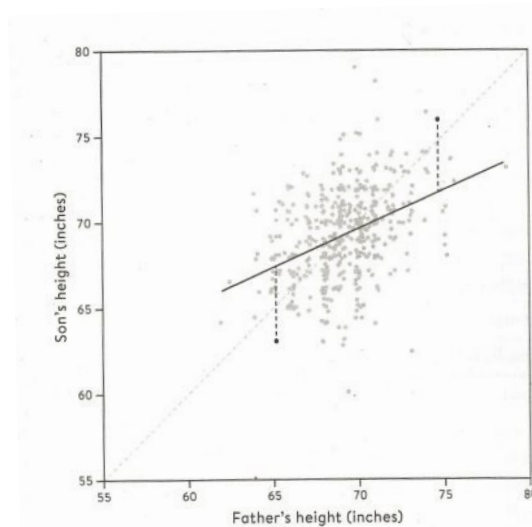


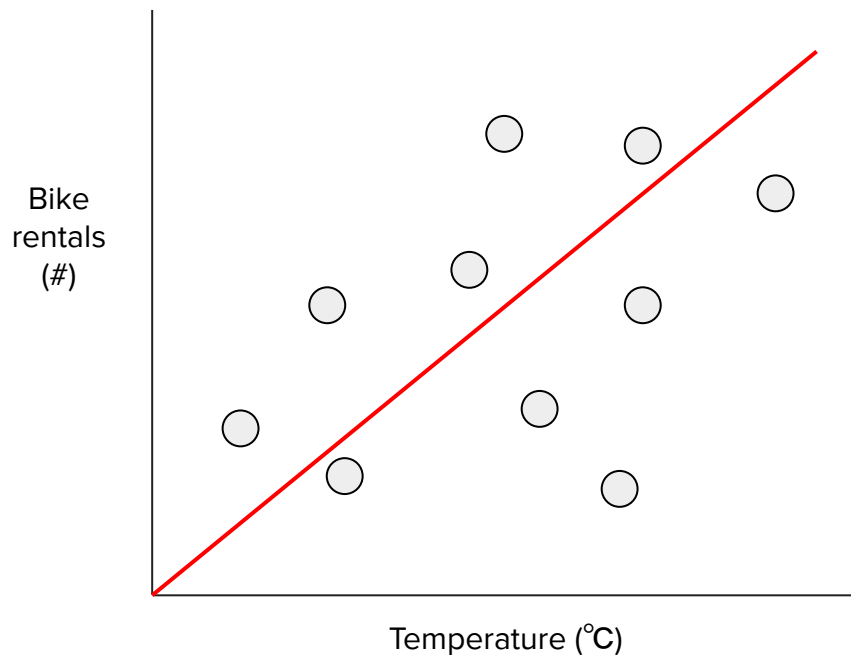
Figure 5.1

Scatter of heights of 465 fathers and sons from Galton's data (many fathers are repeated since they have multiple sons). A jitter has been added to separate the points, and the diagonal dashed line represents exact equality between son and father's heights. The solid line is the standard 'best-fit' line. Each point gives rise to a 'residual' (dashed line), which is the size of the error were we to use the line to predict a son's height from his father's.



Discussion:

Measuring how good a model is



Here's a trained linear regression model that's been fitted to the grey data points.

How do we know how 'good' this model is, or how accurately it will predict house prices when it's deployed in the real world?

How can we estimate the accuracy of this model on data it hasn't seen before?

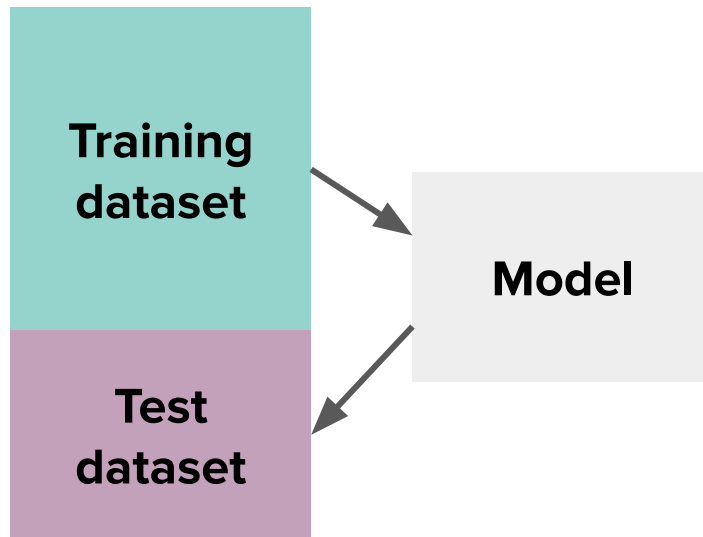
Training vs testing

It wouldn't be a fair assessment of our model to test its performance using data it's already been trained on.

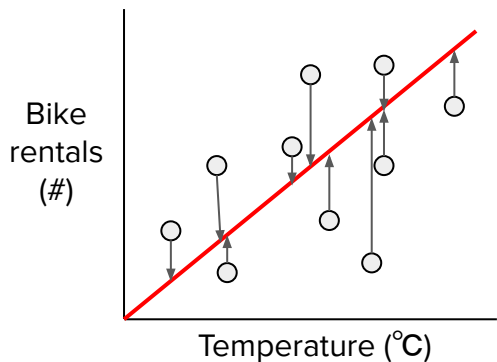
So, we split our overall dataset into two parts: a training dataset and a testing dataset.

We train or fit our model on the training dataset.

We then test its performance on the testing dataset.

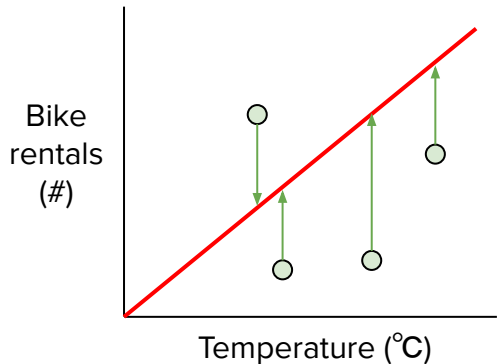


Training vs testing



We can then calculate two types of model performance:

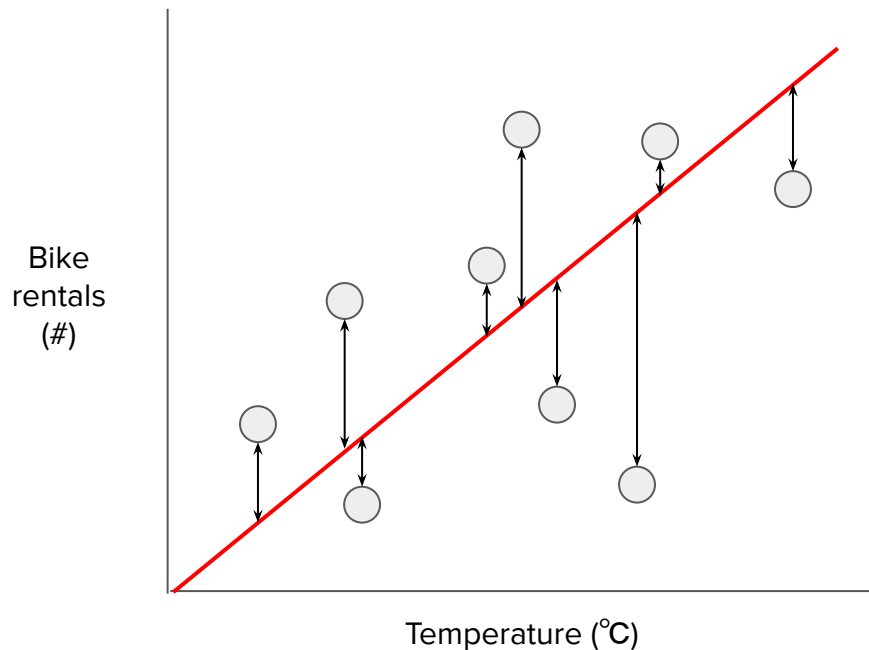
1. **Training error**: this is a measure of how well our model fits the training data. It doesn't tell us how well our model will **generalise**



2. **Testing error**: this measures how well our model performs on data points in the testing set, i.e. data it wasn't trained on.

We know the true values of the target variable in the testing set, so we can measure how far off the model's predictions were from the true values.

Root mean squared error



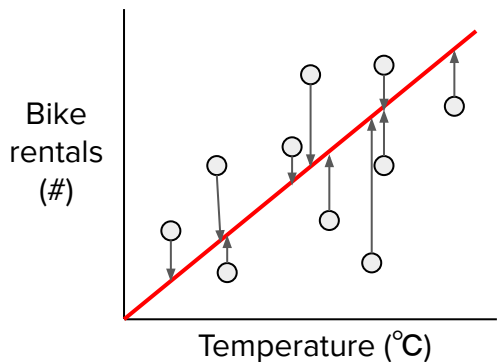
We can measure the performance of a regression model by calculating something called the **root mean squared error (RMSE)**

The RMSE is in the same units as the target variable.

It's a measure of how 'far away' the model's predictions are from the real values.

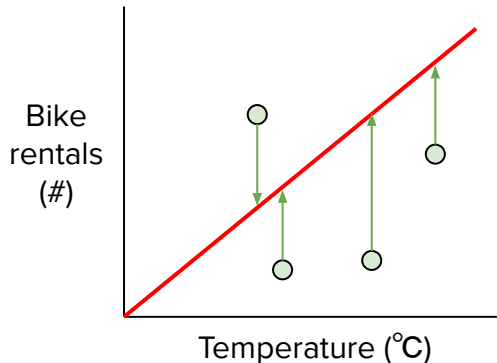
So an RMSE of 200 could be interpreted as 'on average, my model will be wrong by +/- 200 every time it makes a prediction'

Training vs testing



So we'll calculate the RMSE of our linear regression model on the training dataset and on the testing dataset:

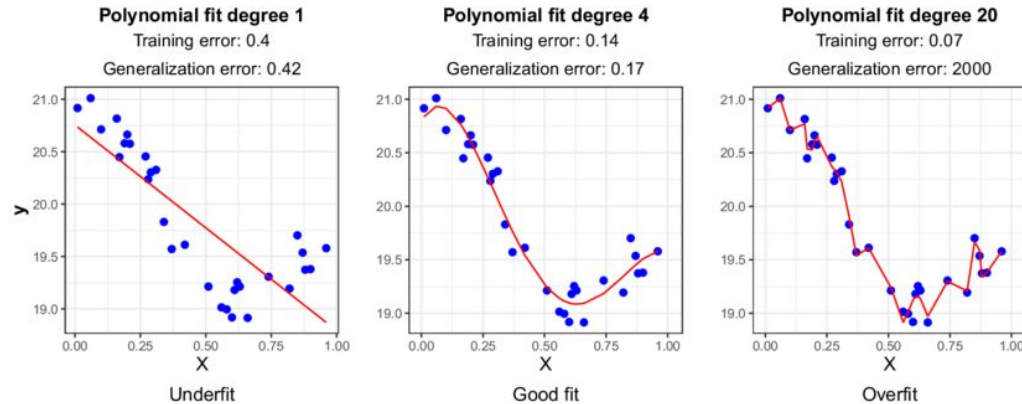
1. **Training RMSE:** This is just a measure of how well our model fits the training data. If we obsess about minimising the training error, we're in danger of **overfitting**



2. **Testing RMSE:** This measures how well our model performs on data points it hasn't seen before and wasn't trained on. This is a good estimation of how accurately our model will perform in the real world.

Overfitting

Overfitting is when your model has been optimised to fit your training data perfectly - this means your training error will be very low, your model will be very complex, but it won't generalise well.





Discussion:

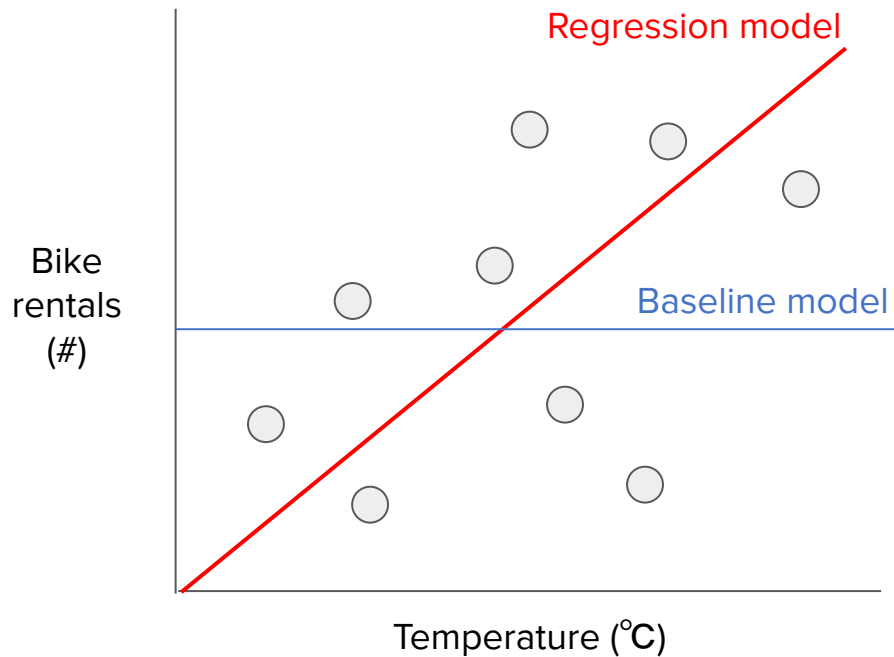
Baseline models

Imagine your linear regression model has an RMSE of 200 on the testing set.

Is this good?

Is there a baseline we can compare this to?

Baseline models



A baseline model is a very simple model that always predicts the **same value** of bike rentals, no matter what the input is!

We can calculate the RMSE of the baseline model on the testing data, and use this as a comparison.

If our regression model performance isn't much better than the baseline model, it's not a good sign.

Simple Linear Regression

- Linear Regression works best when:
 - The data is normally distributed (but =doesn't have to be)
 - Xs significantly explain y (have low p-values)
 - Xs are independent of each other (low multicollinearity)
 - Resulting values pass linear assumption
- If data is not normally distributed, we could introduce bias

