Outliers and Anomalies

Ahmet Balcioglu

03/01/2021

Outline

In this presentation, we will go over the basics of what an outlier is in linear regression and then apply what we have learned there in developing a modern algorithm for a basic problem. A list of topics, I aim to go over:

- Simple linear regression model and Gaussian least squares
- What is an anomaly in simple linear regression and why it is important?
- Logistic regression model and logits, 'odds', function and how it effects outliers
- Talk about a regression model whose aim is to predict itself and why it is useful in understanding anomalies.

What is an Anomaly?

Before we begin, it is important to clear up what we mean by an anomaly. The definition may be a little involved, but simply anomalies are points that do not conform to a well-defined notion of normal behaviour.(Chandola et al.,2009)

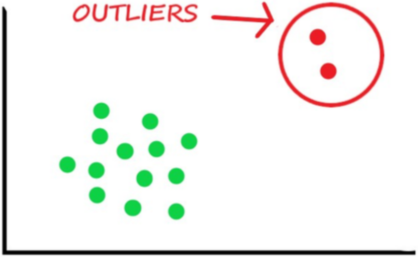


Figure 1: (Tripathi, 2020)

Simple Linear Regression

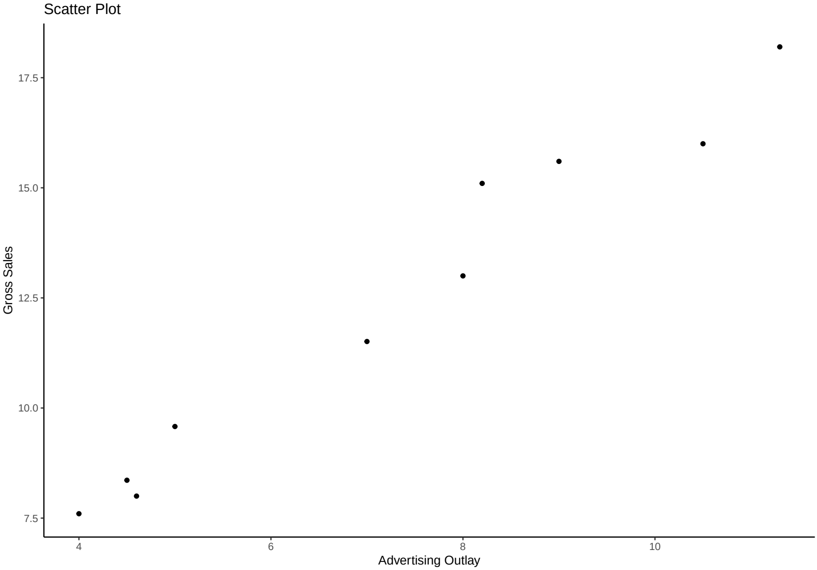
We can summarize the regression model as:

$$\mathbb{E}(Y|X=x) = \beta_0 + \beta_1 x$$
$$var(Y|X=x) = \sigma^2$$

and under some assumptions about the difference between actual and expected values of y. We will begin with an example to help clarify.
Let's suppose we are given the task ascertaining the nature of the relationship between the advertising spending and sales of a company. Our model is:

$$Gross\ sales(Y) = \beta_0 + \beta_1 \times Advertising\ outlay(X) + \epsilon$$

Sales Data


Scatter Plot
Gross Sales
17.5
15.0
12.5
10.0
7.5
4
6
8
10
Advertising Outlay

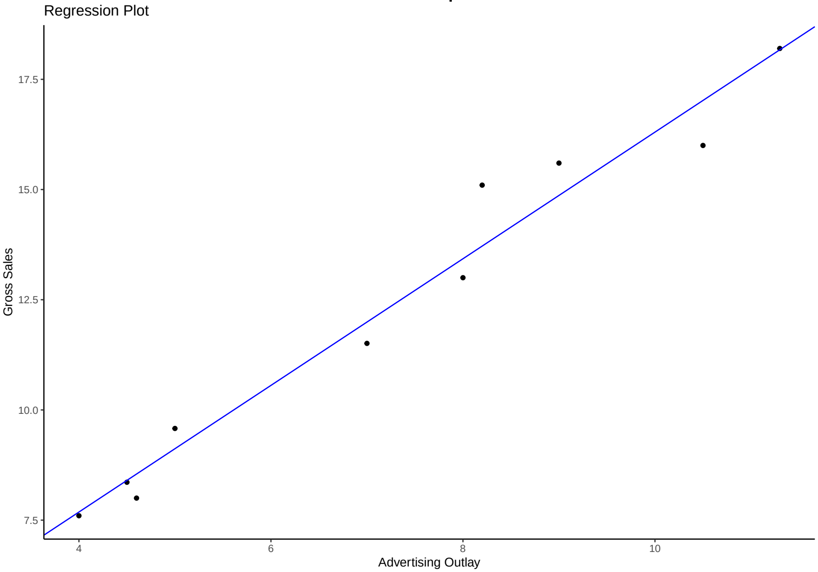
Regression Continued

So far we have our data, and we have a model but we do not yet have a way to calculate the 'best' β_0 and β_1 values. For that end we will try to minimise the distance between an observed value Y_i and the predicted $\mathbb{E}(Y|X = x_i)$, sometimes shown as $\hat{Y}$.
Thus, minimize $(Y - \mathbb{E}(Y|X = x))^2$:

$$
\begin{aligned}
\sum_{i=1}^{n} \epsilon^2 = \sum_{i=1}^{n} (Y_i - \mathbb{E}(Y|X = x_i))^2 \\
= \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 \\
= \sum_{i=1}^{n} (Y_i - \hat{\beta}_0 - \hat{\beta}_1 x)^2
\end{aligned}
$$

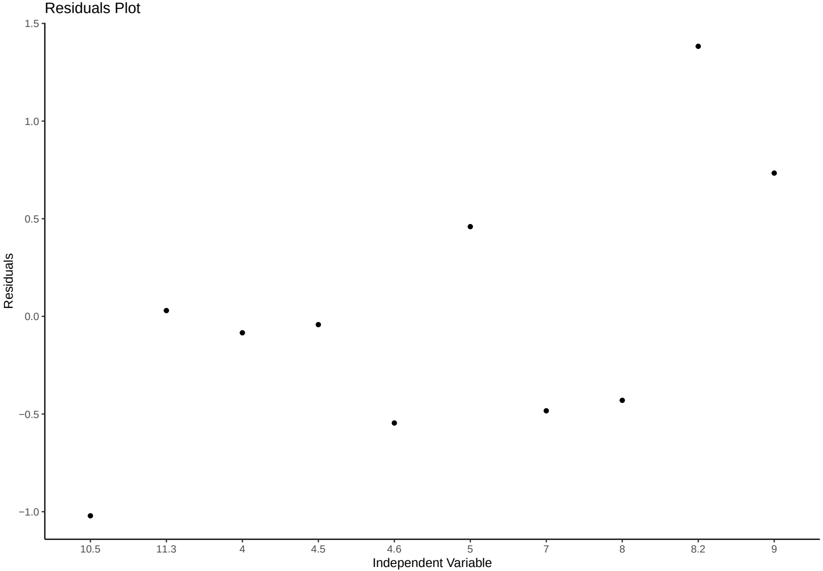
Regression Continued

Let's see how this works in our example:



Regression Errors

We may learn a lot about the success of our model by looking at the errors, this will be a key when dealing with anomalies.



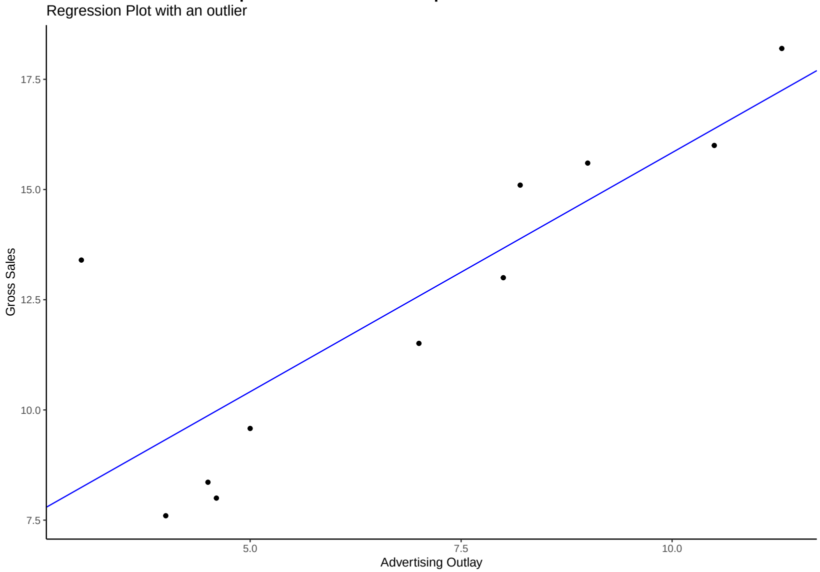
Outliers

Let's add an extra point to our data, which is intended to be an outlier:

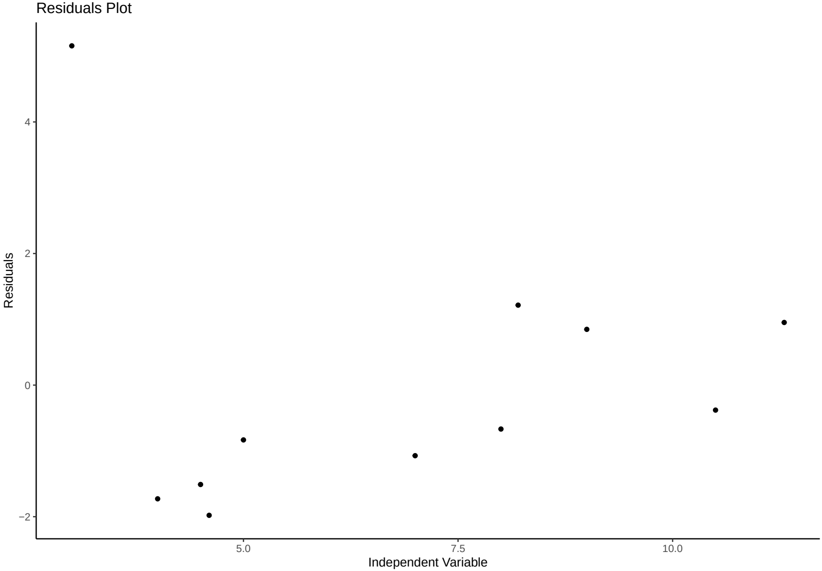
```
Outl_Data<-Data_1%>%
  add_row(.,Year=1996,
          Advertising_outlay=3,
          Gross_sales=13.4,
          .before=1)
Outl_Model<-Outl_Data%>%
  glm(Gross_sales~Advertising_outlay,
      family=gaussian,.,)
```

Regression Plot ..again

Looks like our new point does not quite fit in.

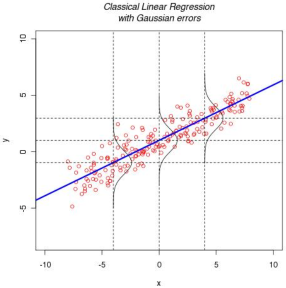


Residual Plot with the outlier



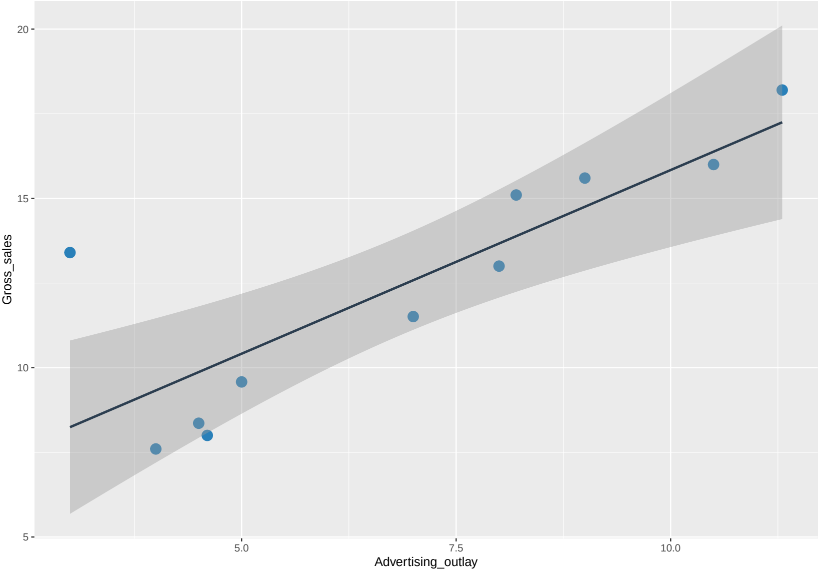
Outlier Detection

How can we qualify our new point as an outlier, or can we give a measure of how much of an outlier a given point is? In a linear regression model, it depends on the errors which are expected to be normally distributed with variance σ^2. Using this information we may create confidence intervals around each $\mathbb{E}(Y|X = x_i)$ with variance σ^2, which may help us quantify how much of an anomaly a point is based on the smallest confidence interval it fits. Note that this approach **assumes** that our model is a reasonable representation of our data.



Confidence Intervals

Here is our 95% confidence intervals plot:



So far..

In this past example we have looked into our data, fitted with an appropriate model and claimed that the minority of points, in our case it was just the one, that disagree greatly with our model are outliers. Bear in mind that this does **not** always suggest that these points are unimportant, or that they should be removed. We have the make our decisions on a case-by-case basis, for example the outlier point we have added corresponded to a greater number in sales by less spending on advertising, which may be of great interest!

Now we will move on to a classification task in which our dependant variable is binary.

$$Y = \begin{cases} 1 & \text{if the outcome is a success} \\ 0 & \text{if the outcome is a failure} \end{cases}$$

, with $\mathbb{P}(Y = 1) = \pi$ and $\mathbb{P}(Y = 0) = 1 - \pi$.

Logistic Regression

In order to adapt to a classification task, we need to change our model a little bit. In particular we need to make sure our predictions lie within $(0, 1)$ to correspond to a probability of success/failure.

$$Y = g(\beta_0 + \beta_1 X)$$

where g would *link* our X's to Y's.

What should our link function be?

It is plausible to try the model the *odds* of success, i.e. $\frac{\mathbb{P}(Y=1)}{\mathbb{P}(Y=0)} = \frac{\pi}{1-\pi}$ this way we can guess a failure when $odds < 1$ and a success when $odds > 1$. However, this goes against our initial instinct of making our prediction lie within $(0, 1)$. One easy solution is to take logarithmic scale:

$$\log(\frac{\pi}{1-\pi}) = \beta_0 + \beta_1 X.$$

This gives us the *logit* link function. Taking an inverse of the logit function gives us:

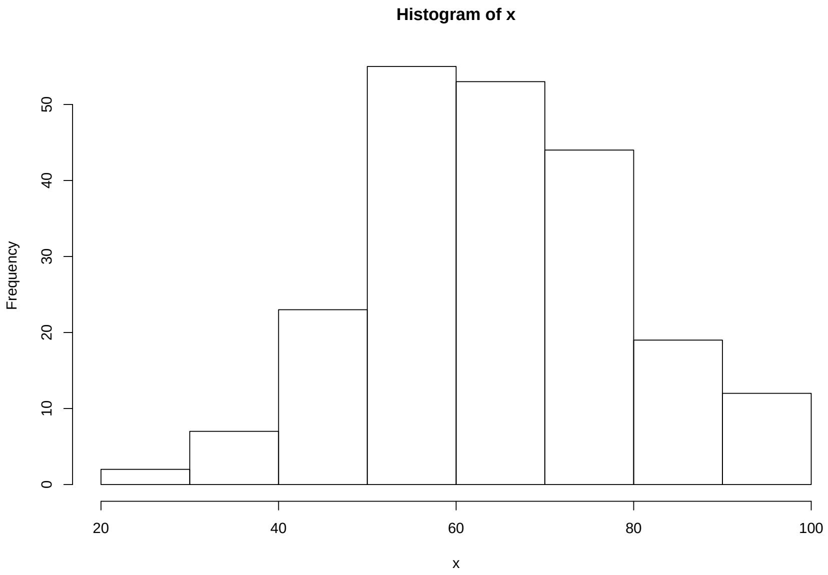
$$g(x) = \frac{e^x}{1+e^x}$$

which is known as the **sigmoid function** $(\sigma(x))$.

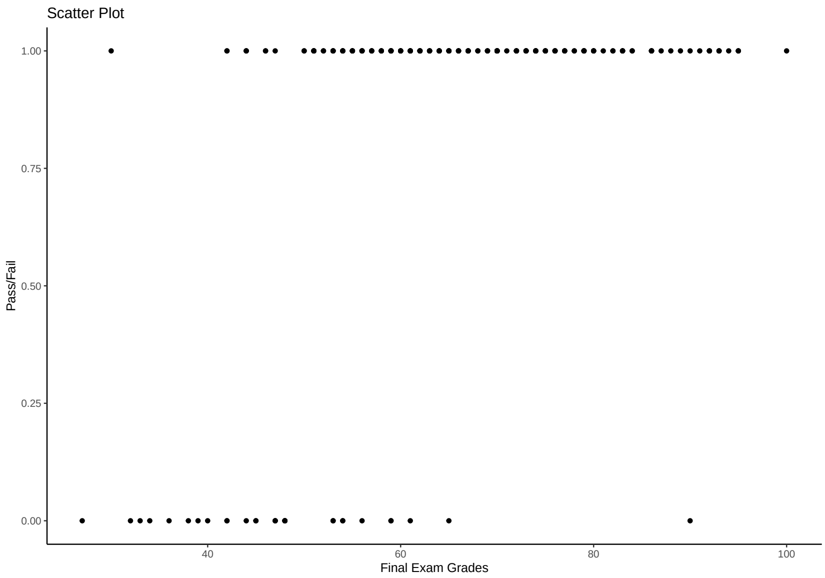
Let us see an example

```r
sample_grades <- function(n){
  x = rnorm(100000, 65, 15)
  x = as.integer(x + 0.5)
  x = replace(x, x > 100, 100)
  return(sample(x, n, replace=TRUE))
}
sample_pass_fail <- function(x){
  y = c()
  for(i in x){
    if(i>=70){y = c(y, 1)}
    if(i<70 & i>=60){y = c(y, runif(1) > 0.05)}
    if(i<60 & i>=50){y = c(y, runif(1) > 0.1)}
    if(i<50 & i>40){y = c(y, runif(1) > .7)}
    if(i<=40){y = c(y,0)}}
  return(y)}
```

Data Histogram



Data Scatter Plot



Let's fit our model

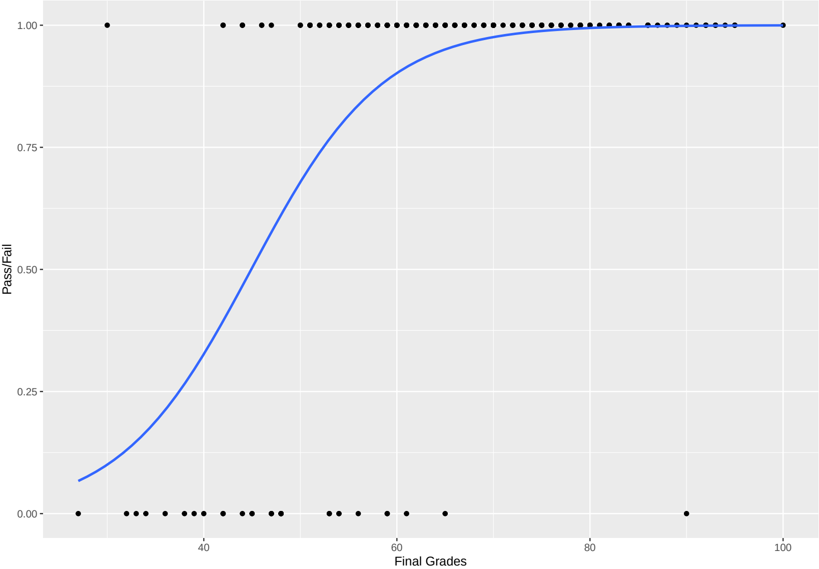
We will use R's glm function:

```r
logit_model = glm(y ~ x,
                  data = student_data,
                  family = binomial(link="logit"))
coef(logit_model)
```

```
## (Intercept)           x
##  -6.6106912   0.1471761
```

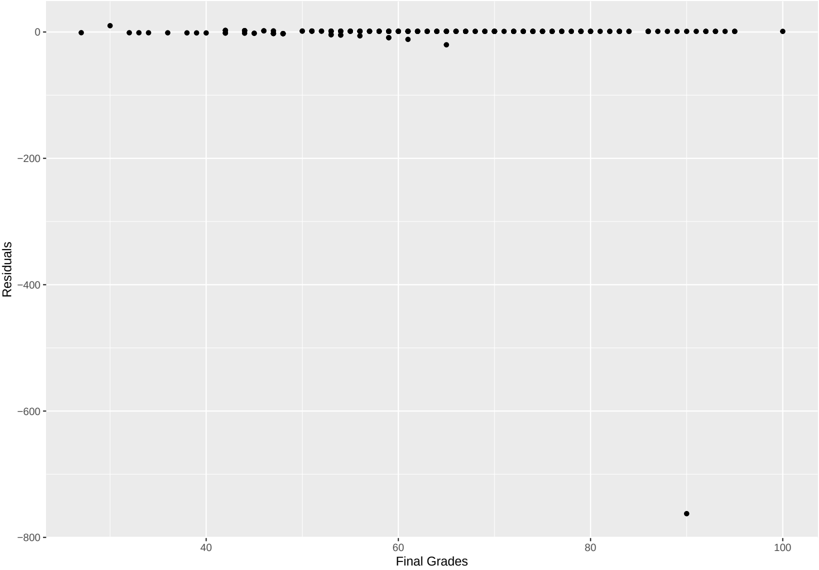
Regression Plot

What do we notice, are there any outliers?



Again how de we quantify outliers?

Quantifying outliers is a more difficult task in logistic regression, but in our example we can still use residuals.



Quantifying outliers?

This time we will formulate a method of our own:

- First we will remove all suspected outliers,
- We will then fit another logistic regression model using this new data,
- We will make predictions in this new model using x values of our suspected points.
- Finally we will decide this point to be an outlier if our prediction and the real value is greatly different.

```r
outlier_removed <-student_data[1:213,]
logit_no_outlier = glm(y ~ x,
                  data = outlier_removed,
                  family = binomial(link="logit"))
```

Quantifying outliers?

As we can observe, there is great difference (> 0.95) between the response and the observed value:

```
c(predict(logit_no_outlier, tibble(x=90), type="response"),
  student_data[214,]$y)
```

```
##         1
## 0.9998324 0.0000000
```

```
c(predict(logit_no_outlier, tibble(x=30), type="response"),
  student_data[215,]$y)
```

```
##          1
## 0.03252968 1.00000000
```

Final task: Anomalies

The problem of fitting a deep neural network can be viewed as a regression problem:

$$Y = f(X) + \epsilon,$$

where ϵ is a random noise variable, (just like before!) and function $f(x)$ has the form:

$$f(x) = W_L \sigma(W_{L_1} \sigma(W_{L-2} \dots \sigma(W_1 x))),$$

where $\sigma(x)$ can be non-linear functions (including our own sigmoid), and W's are matrices.

Our interest is in a particular type of deep neural network, called the **auto-encoder network**, which has the form:

$$X = f(X) + \epsilon.$$

Auto-encoder Networks

Of course, we may just pick $f(x) = x$ as our function and have a trivial result. In fact, if model is not designed carefully this result will be a pitfall for our model.

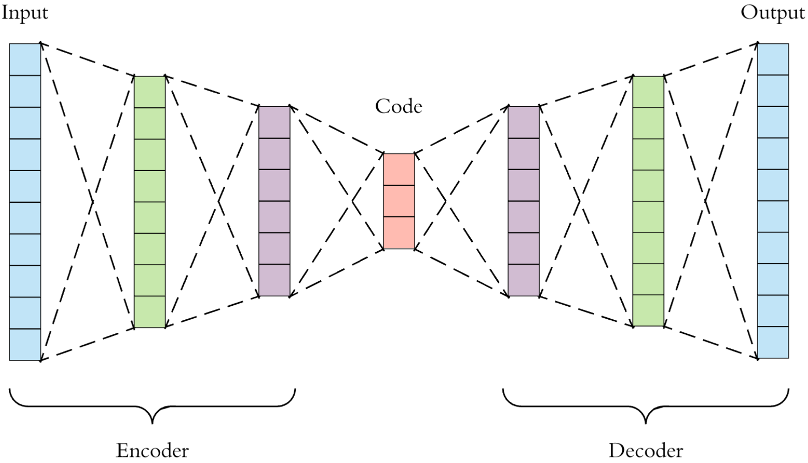


Figure 3: (Dertat, 2017)

Auto-encoder Networks

The way to go around it is to use select an encoder layer that will lose us some information, by using matrices with smaller dimensions than our number of variables, and then try to regain the lost information in the decoder layer.
From this point forward we will continue from python code.

References

- Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection. ACM Computing Surveys, 41(3), 1–58. https://doi.org/10.1145/1541880.1541882
- Dertat, A. (2017, October 3). Applied Deep Learning - Part 3: Autoencoders. Towardsdatascience.Com. https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798
- Tripathi, M. (2020, June 16). Knowing all about Outliers in Machine Learning. Datascience.Foundation. https://datascience.foundation/sciencewhitepaper/knowing-all-about-outliers-in-machine-learning
- Vijay, P. (2020, May 31). Timeseries anomaly detection using an Autoencoder. Keras.io.

References

- Nguyen, V., & (2009, May 12). Linear Regression plot with normal curves for error (sideways). Super Nerdy Cool. http://blog.nguyenvq.com/blog/2009/05/12/linear-regression-plot-with-normal-curves-for-error-sideways