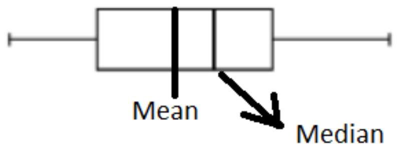
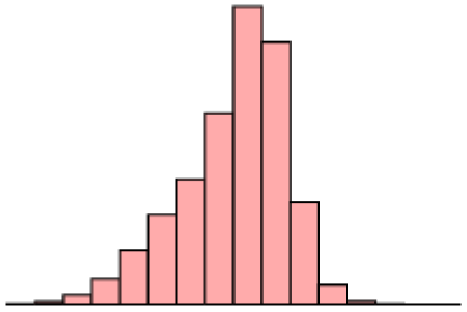
Symmetric and Skewed data

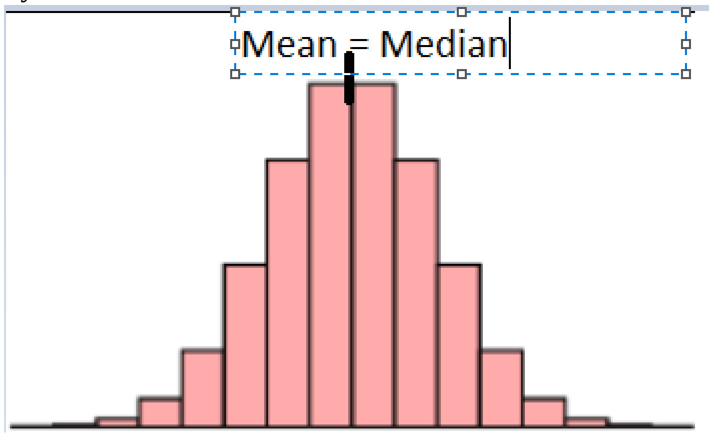
- A distribution is skewed left
  - the mean < the median:



  - the tail of the distribution is longer on the left hand side than on the right hand side. Note: The highest point in skewed left and right is not mean or median, just right for Symmetric skew

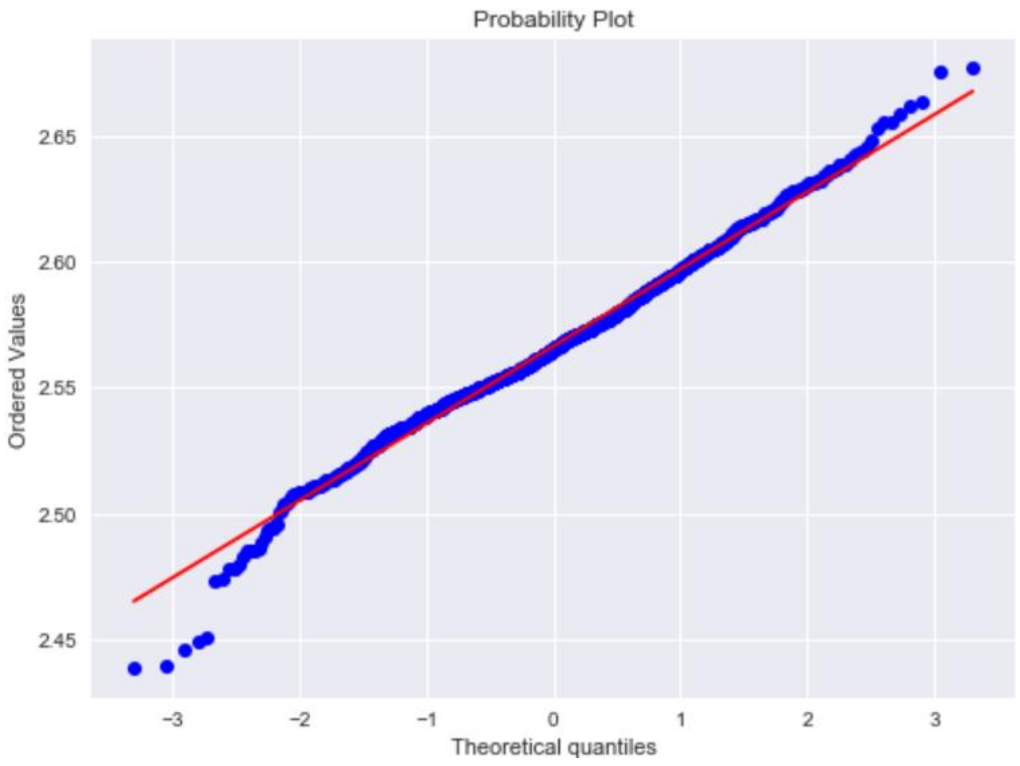


- Symmetric Distribution:

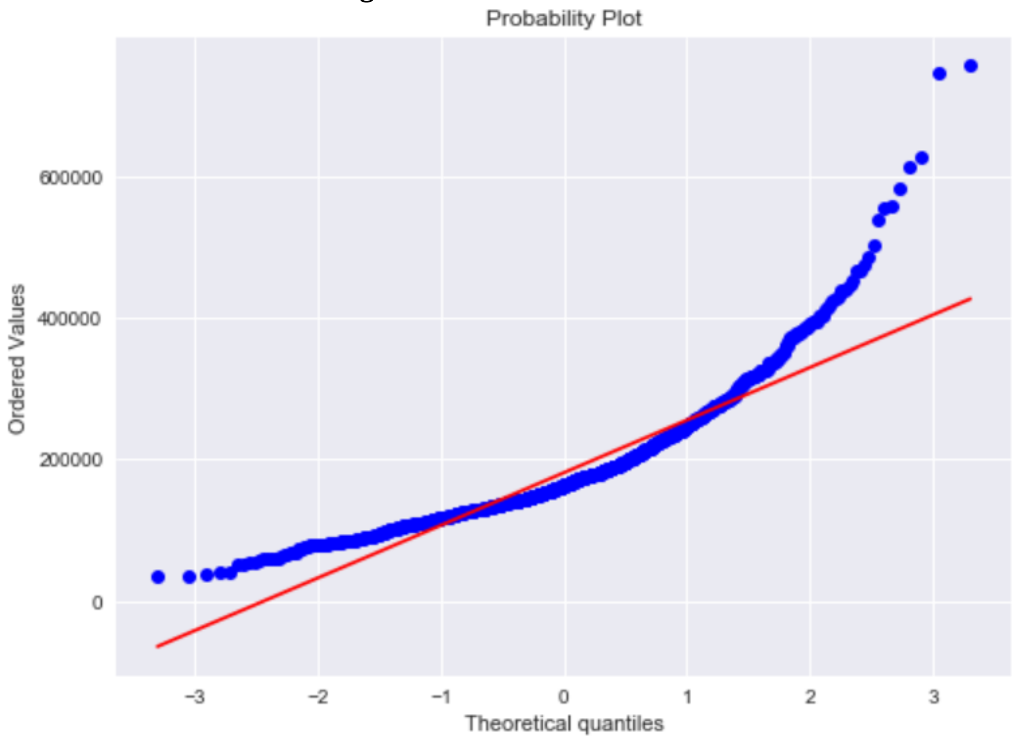


Quantile quantile plot

- The median is a **quantile** where 50% of the data fall below that point and 50% lie above it
- Case 1: A 45 degree angle is plotted on the Q Q plot; if **the two data sets** come from a common distribution, the points will fall on that reference line
- Case 2: A 45 degree angle is plotted on the Q Q plot; if dataset is **normally distributed**



- Case 3: if dataset is **skewed right**



Box Cox Transformation

- This is the way to transform non-normal dependent variables into a normal shape
- Case 1: `*scipy.stats.boxcox*`
    - At the core of the Box Cox transformation is an exponent, lambda **(λ), which varies from -5 to 5**
    - All values λ are considered to get optimal value which results in the best approximation of a normal distribution curve
    - $y(\lambda)=\begin{cases}\dfrac{y^{\lambda}-1}{\lambda} & if\ \lambda\neq 0\\ \log(y) & if\ \lambda=0\end{cases}$
- Case 2: `*scipy.special.boxcox1p*`

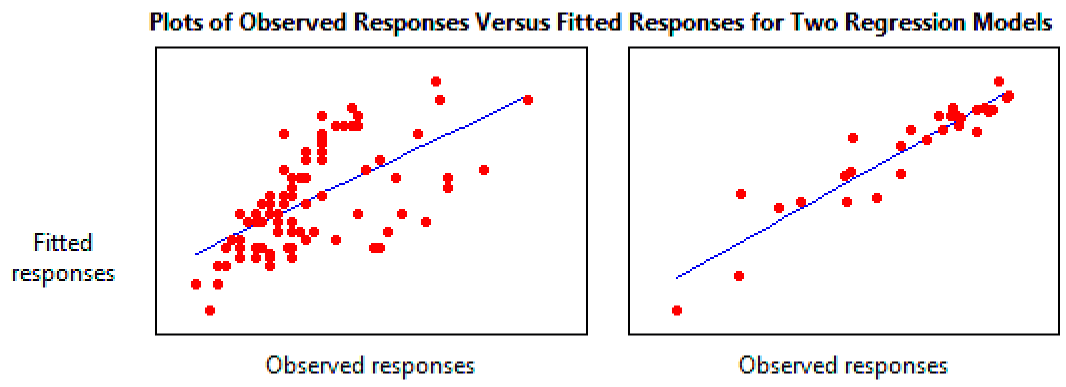
- This test only works for positive data. Another formula is used for negative y-values
- $$y(\lambda) = \begin{cases} \dfrac{(y+\lambda_2)^{\lambda_1} - 1}{\lambda_1} & if\ \lambda_1 \neq 0 \\ \log(y+\lambda_2) & if\ \lambda_1 = 0 \end{cases}$$
- Usually **$\lambda_2 = 1$**

R-Squared

- Statistical measure how close the data are to the fitted regression line
- The percentage of variance can be explained by fitted regression line
- In general, the higher the R-squared, the better the model fits your data

$$\frac{Explained\,Variance}{Total\,Variance}$$

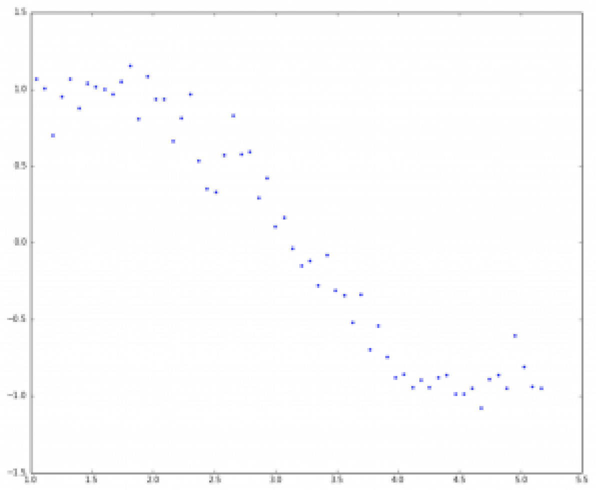
- Explain Variance: Number of observed values equal fitted values
- Graphical Representation of R-squared



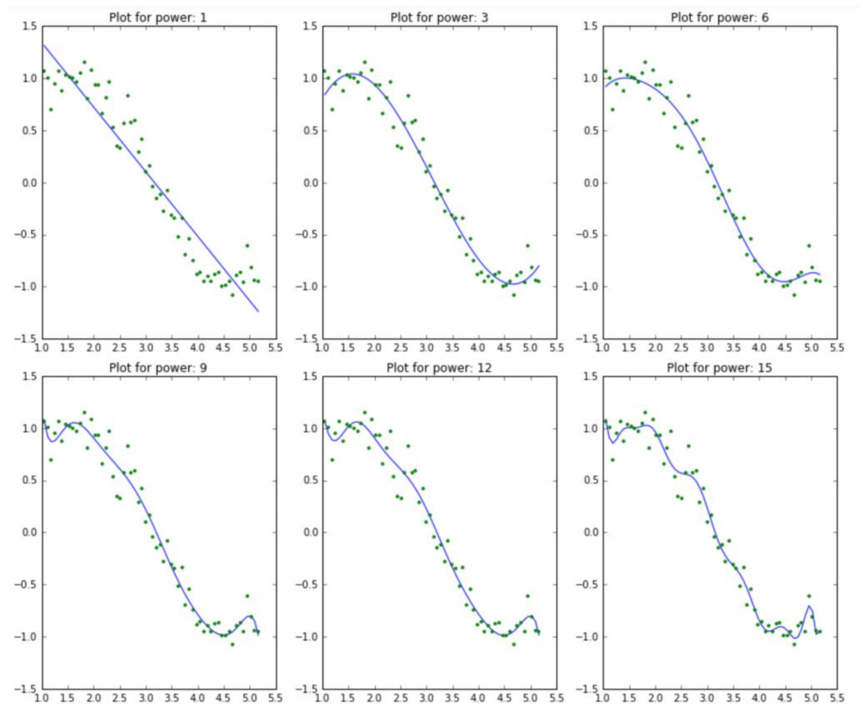
- Easily see that R-squared of first plot is smaller than that of second one

Regularization

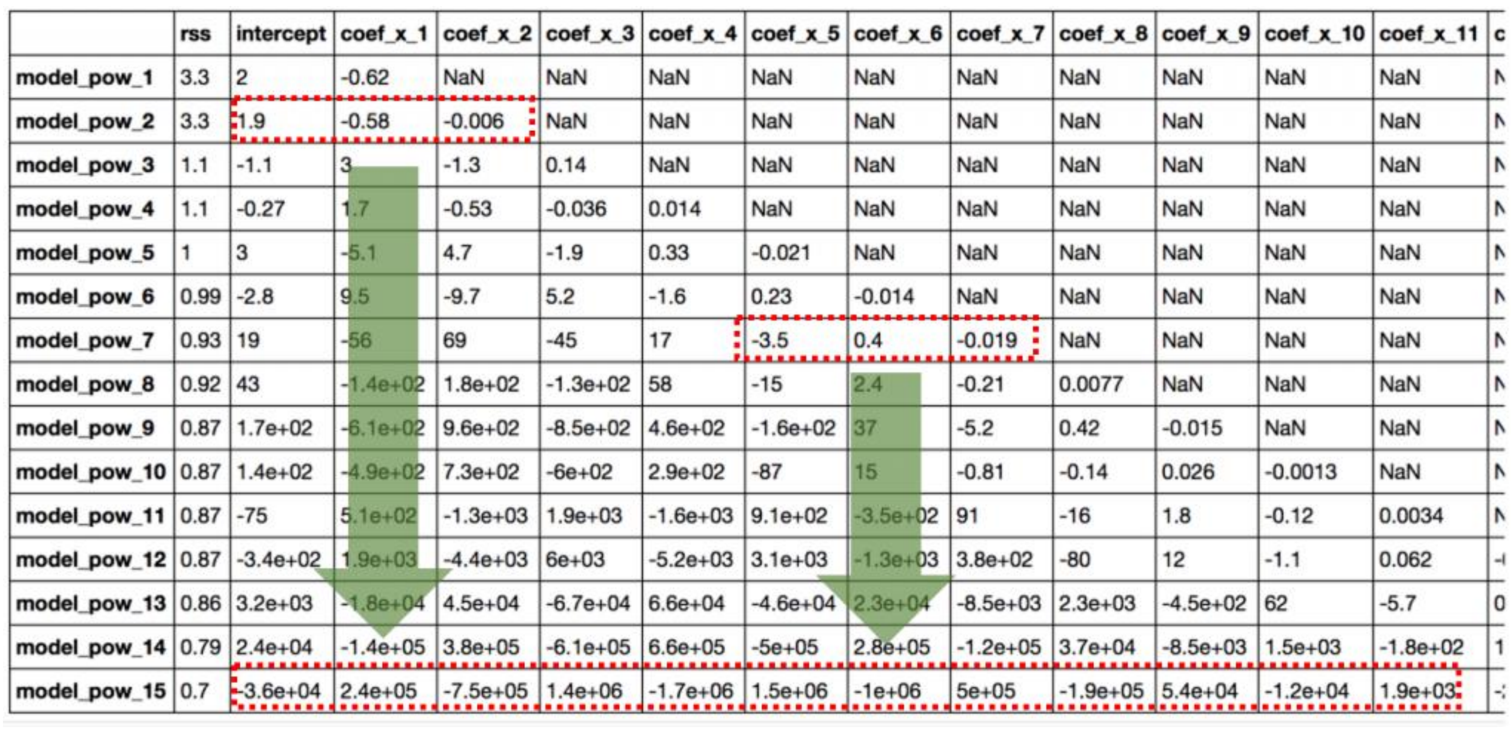
- *Motivation*
  - Let's take some points from sine function



  - Let's try to estimate the sine function using **polynomial regression** with the powers of x from 1 to 15
  - And this is the result



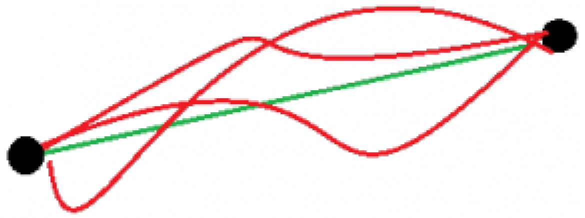
- Clearly see that more power, more fit the regression line is, but this is the coefficients



	rss	intercept	coef_x_1	coef_x_2	coef_x_3	coef_x_4	coef_x_5	coef_x_6	coef_x_7	coef_x_8	coef_x_9	coef_x_10	coef_x_11	c
model_pow_1	3.3	2	-0.62	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
model_pow_2	3.3	1.9	-0.58	-0.006	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
model_pow_3	1.1	-1.1	3	-1.3	0.14	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
model_pow_4	1.1	-0.27	1.7	-0.53	-0.036	0.014	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
model_pow_5	1	3	-5.1	4.7	-1.9	0.33	-0.021	NaN	NaN	NaN	NaN	NaN	NaN	N
model_pow_6	0.99	-2.8	9.5	-9.7	5.2	-1.6	0.23	-0.014	NaN	NaN	NaN	NaN	NaN	N
model_pow_7	0.93	19	-56	69	-45	17	-3.5	0.4	-0.019	NaN	NaN	NaN	NaN	N
model_pow_8	0.92	43	-1.4e+02	1.8e+02	-1.3e+02	58	-15	2.4	-0.21	0.0077	NaN	NaN	NaN	N
model_pow_9	0.87	1.7e+02	-6.1e+02	9.6e+02	-8.5e+02	4.6e+02	-1.6e+02	37	-5.2	0.42	-0.015	NaN	NaN	N
model_pow_10	0.87	1.4e+02	-4.9e+02	7.3e+02	-6e+02	2.9e+02	-87	15	-0.81	-0.14	0.026	-0.0013	NaN	N
model_pow_11	0.87	-75	5.1e+02	-1.3e+03	1.9e+03	-1.6e+03	9.1e+02	-3.5e+02	91	-16	1.8	-0.12	0.0034	N
model_pow_12	0.87	-3.4e+02	1.9e+03	-4.4e+03	6e+03	-5.2e+03	3.1e+03	-1.3e+03	3.8e+02	-80	12	-1.1	0.062	-
model_pow_13	0.86	3.2e+03	-1.8e+04	4.5e+04	-6.7e+04	6.6e+04	-4.6e+04	2.3e+04	-8.5e+03	2.3e+03	-4.5e+02	62	-5.7	0
model_pow_14	0.79	2.4e+04	-1.4e+05	3.8e+05	-6.1e+05	6.6e+05	-5e+05	2.8e+05	-1.2e+05	3.7e+04	-8.5e+03	1.5e+03	-1.8e+02	1
model_pow_15	0.7	-3.6e+04	2.4e+05	-7.5e+05	1.4e+06	-1.7e+06	1.5e+06	-1e+06	5e+05	-1.9e+05	5.4e+04	-1.2e+04	1.9e+03	-

- Clearly see that more fit the regression line, the greater the coefficients are

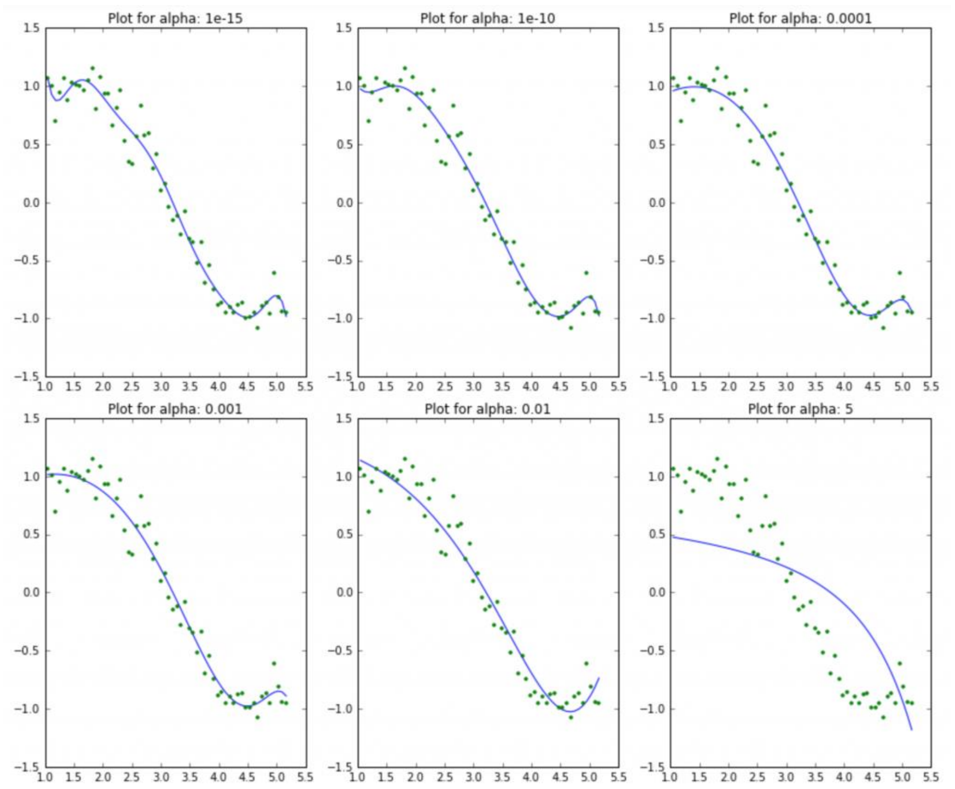
- *What is Regularization*
  - Regularization is a way to avoid **overfitting** by penalizing high-valued regression coefficients
  - Regularization add penalties to more complex models and then sorts potential models from least overfitting to greatest
- *Why is Regularization*
  - Least squares regression can be very unstable because this model just wants to get the minimized residual sum of squares. It results in the ridiculous complex regression model which very fits the training set
  - For example, take a simple dataset of 2 points. The simplest model is a straight line, but there are infinite number of other model: 2nd degree or 3rd degree regression line. In such cases, regularization will add penalties to reduce the overfitting



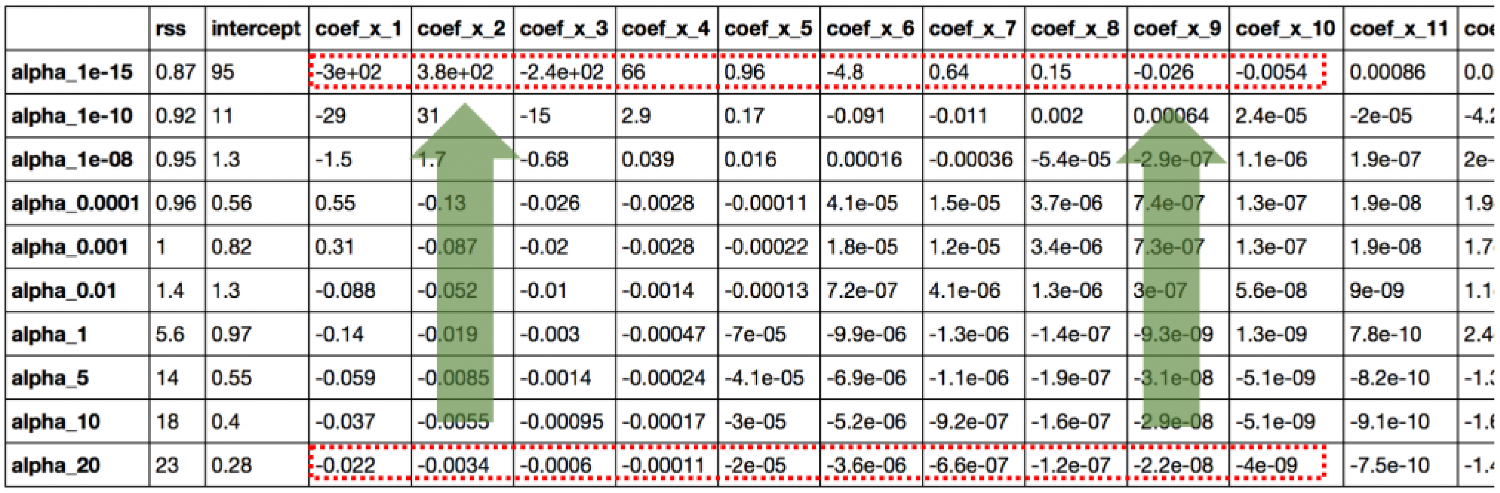
- *Penalty Term*
    - *L2 regularization – Ridge regression*

$$RSS = \frac{1}{2n}\left(\sum_{i=1}^{n}\left(y_i - \sum_{j} x_{ij}\theta_j\right)^2 + \lambda\sum_{j=1}^{p}\theta_j^2\right)$$

- α here is to balance the trade-off between RSS (Residual sum of square) and sum of square of coefficients β
- For example, let's tale above sine function with the power of 15. Now you run Ridge Regression with difference α



- Clearly see that **as the value of alpha increases, the model complexity reduces**
- When alpha of 5, the model got under-fitting. So the alpha should be chosen wisely

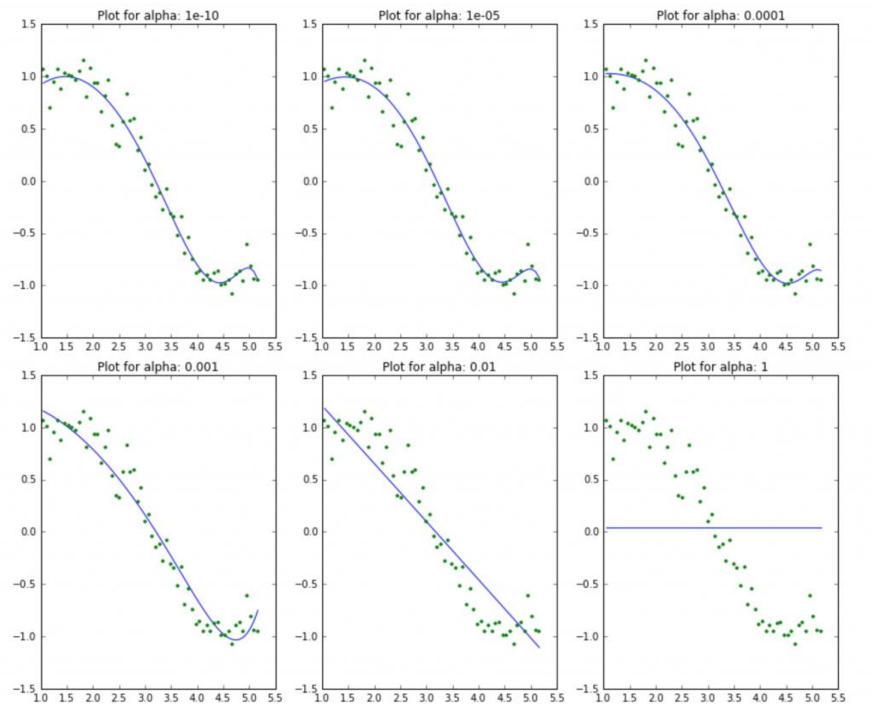


	rss	intercept	coef_x_1	coef_x_2	coef_x_3	coef_x_4	coef_x_5	coef_x_6	coef_x_7	coef_x_8	coef_x_9	coef_x_10	coef_x_11	coe
alpha_1e-15	0.87	95	-3e+02	3.8e+02	-2.4e+02	66	0.96	-4.8	0.64	0.15	-0.026	-0.0054	0.00086	0.0
alpha_1e-10	0.92	11	-29	31	-15	2.9	0.17	-0.091	-0.011	0.002	0.00064	2.4e-05	-2e-05	-4.
alpha_1e-08	0.95	1.3	-1.5	1.7	-0.68	0.039	0.016	0.00016	-0.00036	-5.4e-05	-2.9e-07	1.1e-06	1.9e-07	2e-
alpha_0.0001	0.96	0.56	0.55	-0.13	-0.026	-0.0028	-0.00011	4.1e-05	1.5e-05	3.7e-06	7.4e-07	1.3e-07	1.9e-08	1.9
alpha_0.001	1	0.82	0.31	-0.087	-0.02	-0.0028	-0.00022	1.8e-05	1.2e-05	3.4e-06	7.3e-07	1.3e-07	1.9e-08	1.7
alpha_0.01	1.4	1.3	-0.088	-0.052	-0.01	-0.0014	-0.00013	7.2e-07	4.1e-06	1.3e-06	3e-07	5.6e-08	9e-09	1.1
alpha_1	5.6	0.97	-0.14	-0.019	-0.003	-0.00047	-7e-05	-9.9e-06	-1.3e-06	-1.4e-07	-9.3e-09	1.3e-09	7.8e-10	2.4
alpha_5	14	0.55	-0.059	-0.0085	-0.0014	-0.00024	-4.1e-05	-6.9e-06	-1.1e-06	-1.9e-07	-3.1e-08	-5.1e-09	-8.2e-10	-1.
alpha_10	18	0.4	-0.037	-0.0055	-0.00095	-0.00017	-3e-05	-5.2e-06	-9.2e-07	-1.6e-07	-2.9e-08	-5.1e-09	-9.1e-10	-1.
alpha_20	23	0.28	-0.022	-0.0034	-0.0006	-0.00011	-2e-05	-3.6e-06	-6.6e-07	-1.2e-07	-2.2e-08	-4e-09	-7.5e-10	-1.

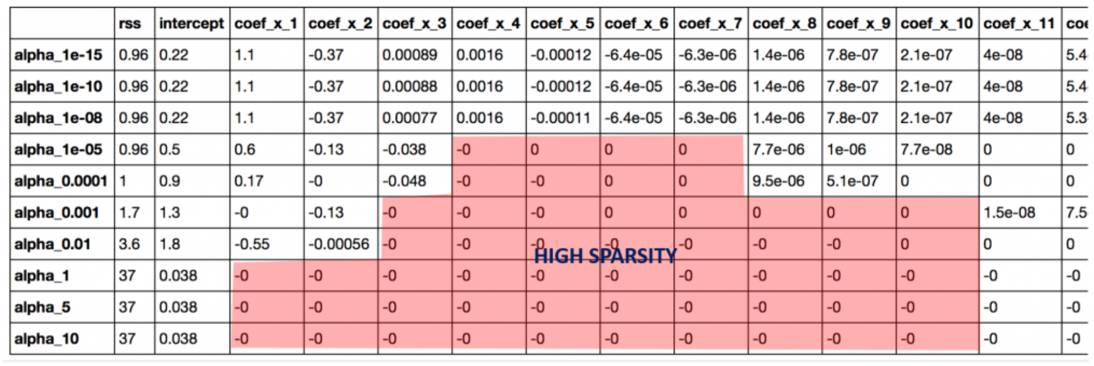
    - Clearly observe that the coefficients are never 0. Remember this point because it's a main difference when compared to Lasso Regression
- *Lasso Regression* – Least **Absolute** Shrinkage and **Selection** Operator

$$RSS = \frac{1}{2n}\left(\underbrace{\sum_{i=1}^{n}\left(y_i - \sum_j x_{ij}\theta_j \right)^2}_{J_1} + \underbrace{\lambda \sum_{j=1}^{p} \left| \theta_j \right|}_{J_2} \right)$$

    - The meaning of α is similar to the ridge regression, but



    - something happens with alpha of 1



	rss	intercept	coef_x_1	coef_x_2	coef_x_3	coef_x_4	coef_x_5	coef_x_6	coef_x_7	coef_x_8	coef_x_9	coef_x_10	coef_x_11	coe
alpha_1e-15	0.96	0.22	1.1	-0.37	0.00089	0.0016	-0.00012	-6.4e-05	-6.3e-06	1.4e-06	7.8e-07	2.1e-07	4e-08	5.4
alpha_1e-10	0.96	0.22	1.1	-0.37	0.00088	0.0016	-0.00012	-6.4e-05	-6.3e-06	1.4e-06	7.8e-07	2.1e-07	4e-08	5.4
alpha_1e-08	0.96	0.22	1.1	-0.37	0.00077	0.0016	-0.00011	-6.4e-05	-6.3e-06	1.4e-06	7.8e-07	2.1e-07	4e-08	5.3
alpha_1e-05	0.96	0.5	0.6	-0.13	-0.038	-0	0	0	0	7.7e-06	1e-06	7.7e-08	0	0
alpha_0.0001	1	0.9	0.17	-0	-0.048	-0	-0	0	0	9.5e-06	5.1e-07	0	0	0
alpha_0.001	1.7	1.3	-0	-0.13	-0	-0	-0	0	0	0	0	0	1.5e-08	7.5
alpha_0.01	3.6	1.8	-0.55	-0.00056	-0	-0	-0	-0	-0	-0	-0	0	0	0
alpha_1	37	0.038	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0
alpha_5	37	0.038	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0
alpha_10	37	0.038	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0

- The coefficient is 0 which means this this feature is eliminated

- *Statistics Proof*
  - Linear Regression

    $$Cost\ function: L(\omega) = \frac{1}{2}\left\| y - \bar{X}\omega \right\|_2^2$$

    $$\nabla_{\omega} L(\omega) = \bar{X}^T\left(\bar{X}\omega - y\right)$$

    $$\omega_{t+1} = \omega_t - \eta\nabla_{\omega} L(\omega) = \omega_t - \eta\bar{X}^T\left(\bar{X}\omega_t - y\right)$$

  - Ridge Regression

    Cost function: $\frac{1}{2}\left\| y - \bar{X}\omega \right\|_2^2\ subject\ to \left\|\omega\right\|_2^2 \le k$

    Apply Lagrange: **Variance + Bias** = Minimize $L(\omega) = \frac{1}{2}\left(\left\| y - \bar{X}\omega \right\|_2^2 + \alpha\left\|\omega\right\|_2^2\right)$

    $$Cost\ function: L(\omega) = \frac{1}{2}\left(\left\| y - \bar{X}\omega \right\|_2^2 + \alpha\left\|\omega\right\|_2^2\right)$$

    $$\nabla_{\omega} L(\omega) = \bar{X}^T\left(\bar{X}\omega - y\right) + \alpha\omega$$

    $$\omega_{t+1} = \omega_t - \eta\nabla_{\omega} L(\omega) = \omega_t - \eta\left[\bar{X}^T\left(\bar{X}\omega_t - y\right) + \alpha\omega_t\right] = (1-\eta\alpha)\omega_t - \eta\bar{X}^T\left(\bar{X}\omega_t - y\right)$$

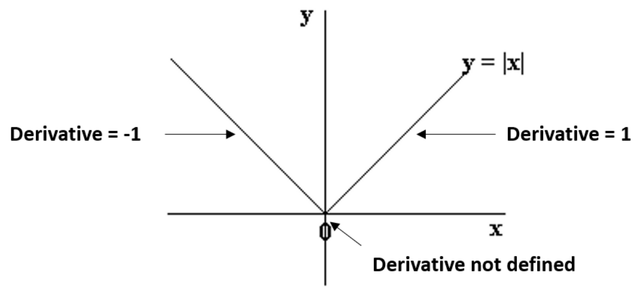
    Ridge regression is equivalent to Linear Regression by reducing $(1-\eta\alpha)\omega_t$, then apply same update rule as simple linear regression. And this is why the coefficients never get 0

  - Lasso Regression

    $$Cost\ function: L(\omega) = \frac{1}{2}\left(\left\| y - \bar{X}\omega \right\|_2^2 + \alpha\left\|\omega\right\|_1\right)$$

    $$\nabla_{\omega} L(\omega) = \bar{X}^T\left(\bar{X}\omega - y\right) + \alpha\omega$$

    In this case, gradient is not differentiable at x = 0



Coordinate descent: $\omega_i = \begin{cases} \rho_i + \frac{\alpha}{2} & if\ \rho_i < -\frac{\alpha}{2} \\ 0 & if\ -\frac{\alpha}{2} < \rho_i < \frac{\alpha}{2} \\ \rho_i - \frac{\alpha}{2} & if\ \rho_i > \frac{\alpha}{2} \end{cases}$

$$\rho_i = \sum_j feature_i\left(output_j - prediction_j + \omega_i\, feature_i\right)$$

- *Conclusion*
  - Correlation between λ *and* θ
    - $\frac{\partial J(\theta)}{\partial \theta} = \frac{\partial J_1(\theta)}{\partial \theta} + \lambda \frac{\partial J_2(\theta)}{\partial \theta}$
    - Mission of lasso or ridge regression is to decrease the weights of coefficient which results in decreasing the complexity of model
    - When you increase lambda, you make the GD concentrate on $\frac{\partial J_2(\theta)}{\partial \theta}$, so θ will decrease

$$\rightarrow \begin{cases} \lambda \nearrow \rightarrow \theta \searrow & \theta = \theta - \lambda\theta \\ & \lambda \swarrow \rightarrow \theta \searrow \end{cases}$$

    - Another way to explain when lambda increase:
      - when $\lambda \rightarrow \infty$, it means **high bias**, compare to the equation in ridge regression:

        $\theta = \left(\mathbf{x}^{\mathbf{T}}\mathbf{x}\right)^{-1}\mathbf{x}^{\mathbf{T}}\mathbf{y} = \left(\mathbf{x}^{\mathbf{T}}\mathbf{x} + I\alpha\right)^{-1}\mathbf{x}^{\mathbf{T}}\mathbf{y} = \frac{1}{\alpha}\mathbf{x}^{\mathbf{T}}\mathbf{y} \rightarrow \mathbf{0} \rightarrow \mathbf{y} \approx \mathbf{0} \rightarrow$ **Underfitting**
      - When $\left\|y - \bar{X}\omega\right\|_2^2$ high, Variance is high. To solve high variance, you need more data rows, remove features or increase lambda
    - The coefficient **C in sklearn model** that means **lambda/ alpha in regularization**

  - Key Difference:
    - ❖ Ridge: It includes all (or none) of the features in the model. Thus, the major advantage of ridge regression is coefficient **shrinkage** and **reducing** model complexity.
    - ❖ Lasso: Along with shrinking coefficients, lasso performs **feature selection** as well
  - Typical Use Cases
    - ❖ Ridge: It is majorly used to prevent overfitting. Since it includes all the features, it is not very useful when having millions of features
    - ❖ Lasso: Since it provides **sparse** solutions, it is generally the model of choice for modelling cases where the #features are in millions or more. In such a case, getting a sparse solution is of great computational advantage as the features with zero coefficients can simply be ignored
  - Presence of Highly Correlated Features
    - ❖ Ridge: It generally works well even in presence of highly correlated features as it will include all of them in the model but the coefficients will be distributed among them depending on the correlation.
    - ❖ Lasso: It arbitrarily selects any one feature among the highly correlated ones and reduced the coefficients of the rest to zero
- *Elastic Net*
  - The Combination of Ridge and Lasso Regression

$$Cost\ \ function: L(\omega) = \frac{1}{2}\left(\left\|y - \bar{X}\omega\right\|_2^2 + \alpha\left\|\omega\right\|_2^2 + \alpha\left\|\omega\right\|_1\right)$$

Evaluation in Classification

- *Confusion Matrix*

Mark	Temperature	y	y	Confusion Matrix
8	23	1	1	TP
6	25	1	1	TP
4	28	1	0	FN
8	30	1	0	FN
5	25	1	1	TP
2	27	0	0	TN

4	28	0	1	
7	21	1	0	
9	20	0	1	
8	11	1	1	
8	36	0	0	
5	12	1	0	
2	4	1	1	
9	7	0	1	
7	12	0	1	
5	33	0	0	
4	35	1	1	
9	37	1	0	
6	38	0	0	
3	12	0	0	

→ Confusion matrix:

	Predicted Value		
		1	0
Actual Value	1	True Positive	False Negative
	0	False Positive	True Negative

	Predicted Value		
		1	0
Actual Value	1	6	5
	0	4	5

- Precision = $\frac{TP}{TP+FP}$
- Recall = $\frac{TP}{TP+FN}$
- Precision and Recall are a tradeoff

- *F1-Score/ Harmonic Mean*
  - Let's take example, you don't care about what model should be used, you just predict all values is 1, so the confusion matrix:

	Predicted Value		
		1	0
Actual Value	1	11	0
	0	9	0

  - You easily got the precision = 11/20 > 0.5 and the recall is 100%, so the precision and recall may not evaluate your model correctly

  - Now you got

$$Harmonic\ Mean = 2\times\frac{\Pr ecision\times \mathrm{Re}call}{\Pr ecision+\mathrm{Re}call} = 2\times\frac{\frac{11}{20}\times 1}{\frac{11}{20}+1} = 0.7$$

which neutralize the Precision and Recall

$$F_\beta = \frac{(1+\beta^2)PR}{\beta^2(P+R)}$$

- In Logistics Regression, Threshold usually is 0.5, at this point Precision and Recall is equal. but Based on the situation, you can $\begin{cases} threshold \nearrow \rightarrow \mathrm{Pr}ecision \nearrow \\ threshold \swarrow \rightarrow \mathrm{Re}call \nearrow \end{cases}$

- *Macro/ Micro/ Weighted Average*
  - Imagine you have a One-vs-All (there is only one correct class output per example) multi-class classification system with four classes and the following numbers when tested:
    - Class A: 1 TP and 1 FP
    - Class B: 10 TP and 90 FP
    - Class C: 1 TP and 1 FP
    - Class D: 1 TP and 1 FP

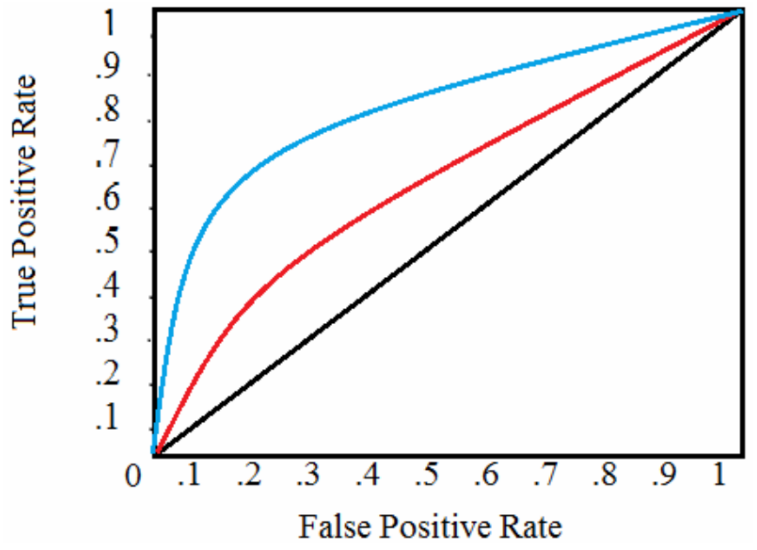
  →Precision$_A$ = Precision$_C$ = Precision$_D$ =0.5 and Precision$_B$ =0.1

  - Macro average: Precision = $\frac{0.5+0.1+0.5+0.5}{4} = 0.4$
  - Micro Average: Precision = $\frac{1+10+1+1}{2+100+2+2} = 0.123$
  - Macro Average got a little bit incorrect because class B takes for 94.3% dataset but in macro average it just takes equal ratio of 25%. Whereas, the micro average will capture this class imbalance and bring average down to 0.123
  - To handle the class imbalance, aside from micro average, Weighted macro average can handle it
  - Weighted macro average will weigh the precision of each class:

  $$\mathrm{Pr}ecision_{weight-macro} = 0.0189 \bullet 0.5 + 0.943 \bullet 0.1 + 0.0189 \bullet 0.5 + 0.0189 \bullet 0.5 = 0.123$$

  - 0.0189 or 0.943 is the weight you choose

- *Receiver Operating Characteristic (AUC Score): Used for binary classification*



  - A ROC plot shows:
    - Test accuracy: the closer the graph is to the top and left-hand borders, the more accurate the test. Likewise, the closer the graph to the diagonal, the less accurate the test.
    - A perfect test would go straight from zero up the the top-left corner and then straight across the horizontal.
    - The likelihood ratio; given by the derivative at any particular cutpoint.
  - How to calculate:

- Recall (True Positive Rate) and Fallout(False Positive Rate) = $\frac{FP}{FP+TN}$
- Calculate Area between (Blue or red line) and Diagonal Line