Regression & Classification

Dr. Anuja Arora, Jaypee Institute of Information Technology

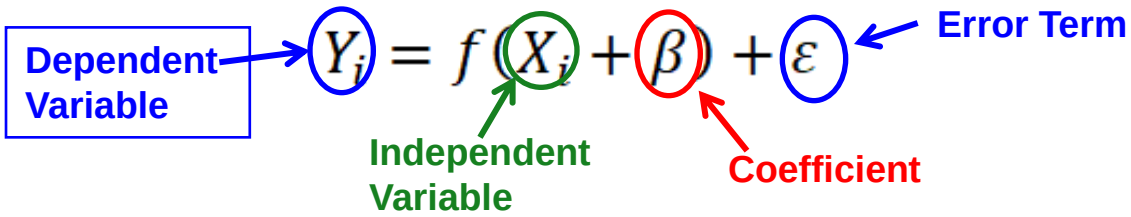
Table of Content

What is Regression

- Regression is a method of modelling a **target value / dependent value** based on **independent predictors**.
- It is a **statistical tool** which is used **to find out the relationship between the outcome variable** also known **as the dependent variable, and one or more variable often called as independent variables.**
- Regression techniques mostly differ based on the **number of independent variables** and the **type of relationship between the independent and dependent variables.**

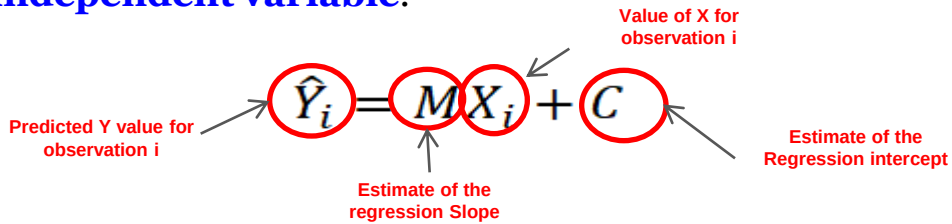
When and Why do we Need Regression?

- When dependent variable is of **Continuous data type.** Independent Variable/predictor could be of **any data type** such as continuous, nominal/categorical, etc.
- Regression method tries **to find the best fit line** which shows the relationship between the dependent variable and predictors with **least error.**
- In regression, the **output/dependent variable** is the **function of an independent variable and the coefficient and the error term.**



Simple Linear Regression

- **Simple Linear Regression:** Establish a best linear relationship to **predict a dependent variable** using **Single independent variable**.



- A linear regression line has the equation

  Y = mx+c,

- The mathematical formula to calculate slope (m) is:

  m= (mean(x) * mean(y) – mean(x*y)) / (mean (x)^2 – mean(x^2))

- The formula to calculate intercept (c) is:

  C= mean(y) – mean(x) * m

Simple Linear Regression Exercise and Python Code

Simple Linear Regression Example

Dataset: Home Area- Price

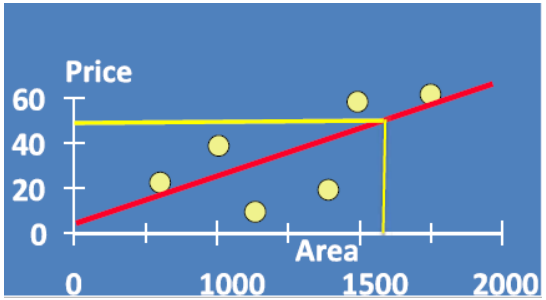
Assume this dataset share linear relationship

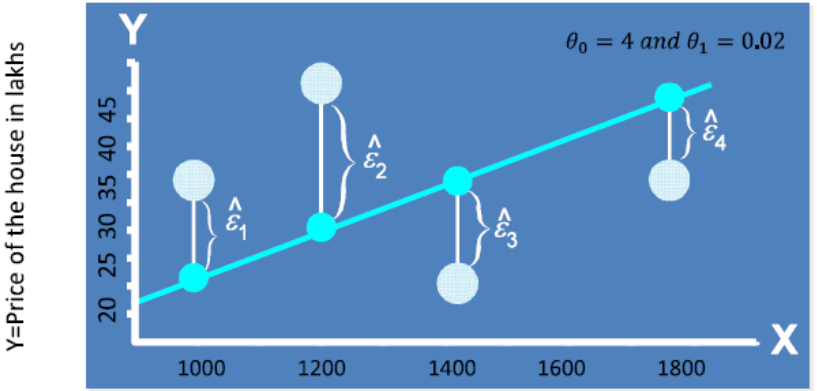
Independent Variable

Area (in sqft)	1000	1200	1500	1800	2000	2100	2200	2300
Price (in Lacs)	30	35	42	55	59	62	67	70

Dependent Variable

Project the area for which user wants to know price. For exam: 1600 Square feet





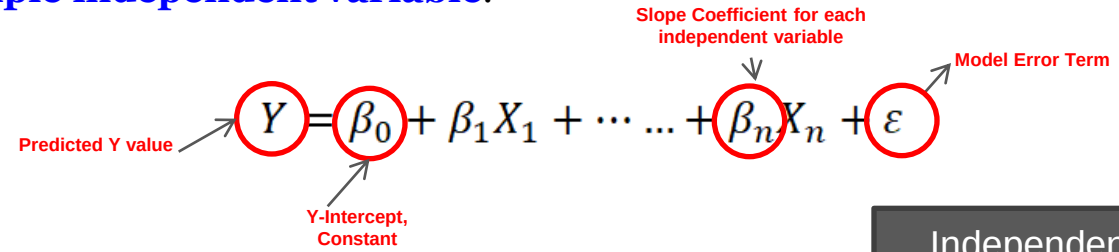
Assumed $Price = \theta_0 + \theta_1 * Area$ and selected θ_0 and θ_1

Assumptions: Simple Linear Regression

- **Linearity:** The relationship between independent variables and the mean of the dependent variable is linear.
- **Homoscedasticity**: If Variance of errors are constant across independent variables, then it is called Homoscedasticity. So, the variance of residuals (error) should be equal.
- **Independence:** Observations are independent of each other.
- **Normality:** For any fixed value of an independent variable, the dependent variable is normally distributed.

Multiple Linear Regression

- **Multiple Linear Regression:** Establish a best linear relationship to **predict a dependent variable** using **Multiple independent variable**.



Independent Variables

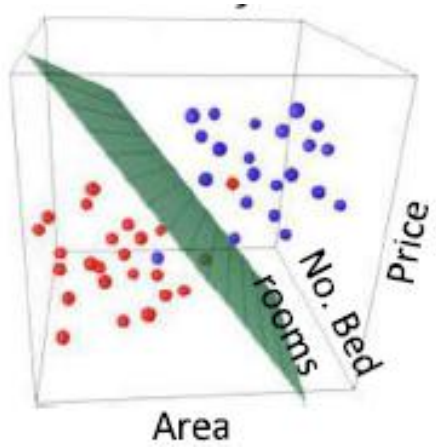
Dataset: Home Area, No of bed rooms, Price

Area (in sq ft)	1000	1200	1500	1800	2000	2100	2200	2300
No of Bedrooms	2	2	3	2	3	4	3	4
Price (in Lacs)	30	35	42	55	59	62	67	70

Dependent

$$Price = \theta_0 + \theta_1 * Area + \theta_2 * Number\ of\ bed\ rooms$$

Multiple Linear Regression



$\theta_0 = 4$
$\theta_1 = 0.02$
$\theta_2 = 5$

Other independent factors: Distance from city centre in kms

Multiple Linear Regression
Python Code

Assumptions: Multiple Linear Regression

- **Linearity:** The relationship between independent variables and the mean of the dependent variable is linear.
- **Multicollinearity:** There should not be high correlation between two or more independent variables. Multicollinearity can be checked using correlation matrix, Tolerance and Variance Influencing Factor (VIF).
- **Homoscedasticity**: The variance of residuals (error) should be equal. The residuals should be homoscedastic. Standardized residuals versus predicted values is used to check homoscedasticity.
- **Independence:** Observations are independent of each other.
- **Multivariate Normality:** Residual should be normally distributed.

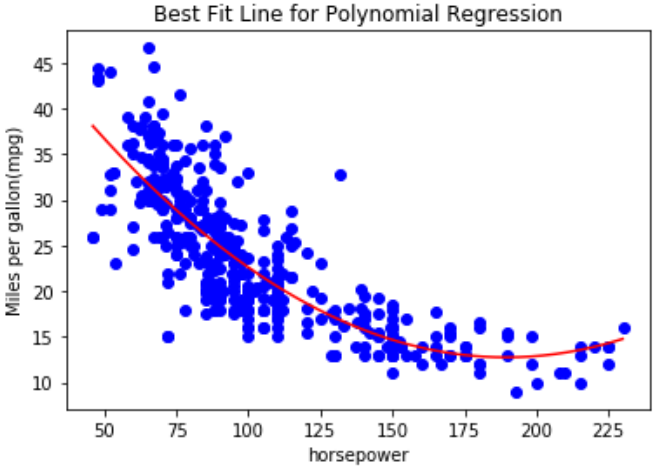
Polynomial Linear Regression

- Polynomial regression is a **non-linear regression**. In Polynomial regression, the relationship of the dependent variable is fitted to **the nth degree of the independent variable.**

Equation of Polynomial Regression

$$Y = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \beta_3 X_i^3 + \cdots \ldots + \beta_n X_i^n + \varepsilon_i (i = 1,2, \ldots, n)$$

Polynomial Linear Regression
Python Code

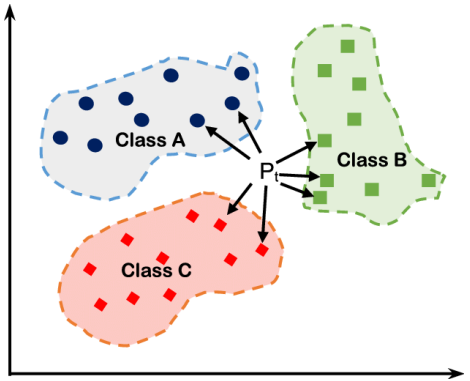


KNN Classifier

- **KNN Classifier:**
  K Nearest Neighbour is simple algorithm and **classifies** new cases based on **a similarity measure**. KNN falls under **lazy learning classifier** means no explicit training phase before classification.

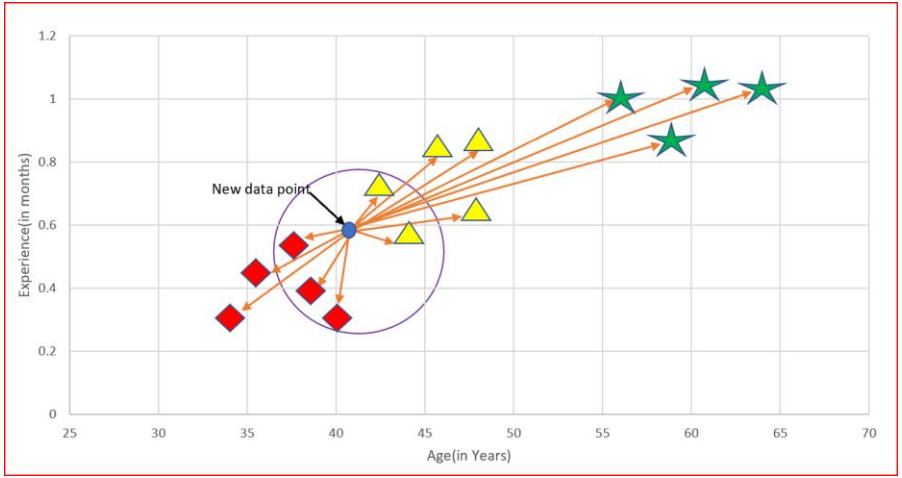
- **KNN Important Measures:** Value of **K, Distance Metric.**

- **Distance Metric:** Euclidean Distance, Cosine Distance, manhattan distance, minkowski distance, etc.



KNN Classifier

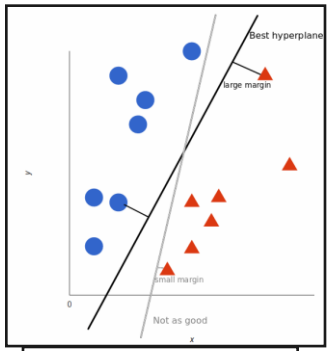
- **Step1:** Choose a value for K.
- **Step2:** Find the distance of the new point to each of the training data.
- **Step3:** Find the K nearest neighbors to the new data point.
- **Step4:** For classification, count the number of data points in each category among the k neighbors. New data point will belong to class that has the most neighbors. For regression, value for the **new data point will be the average of the k neighbors**.



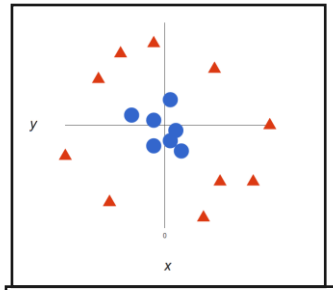
KNN Classifier Exercise and Python Code

SVM Classifier/ Regression (SVR)

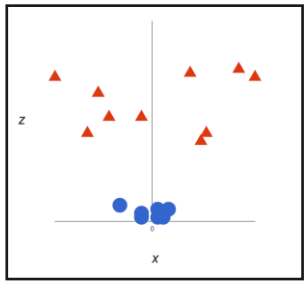
- **Support Vector Machine:** This is a supervised ML model formally defined by separating hyperplane. It finds out a line/ hyperplane in multidimensional space to separate out classes.



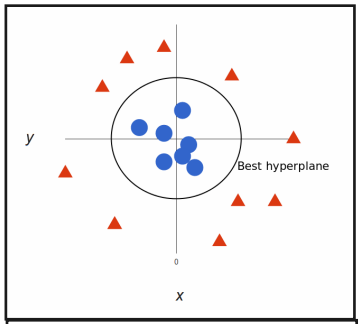
Simple
classify in 2 classes



Little complex
Not possible separating
line in 2D Plane



Plot in zy axis,
easily to separate by line



Transform back to XY and
line changes to circle

This transformations are called ‘Kernels’

SVM Classifier

- **SVM Tuning Parameters: Kernel, Regularization, Gamma, and Margin**

regularization

```
# Modeling Different Kernel Svm classifier using Iris Petal features
iris = datasets.load_iris()
X = iris.data[:, 2:]  # we only take the last two features.
y = iris.target
C = 1.0  # SVM regularization parameter

# SVC with linear kernel
svc = svm.SVC(kernel='linear', C=C).fit(X, y)
# LinearSVC (linear kernel)
lin_svc = svm.LinearSVC(C=C).fit(X, y)
# SVC with RBF kernel
rbf_svc = svm.SVC(kernel='rbf', gamma=0.7, C=C).fit(X, y)
# SVC with polynomial (degree 3) kernel
poly_svc = svm.SVC(kernel='poly', degree=3, C=C).fit(X, y)
```

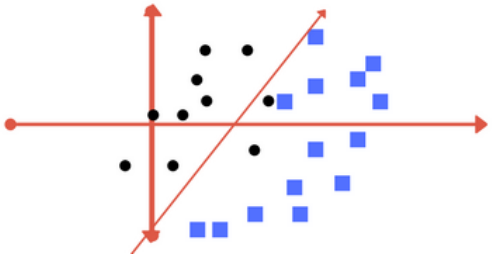
kernel

gamma

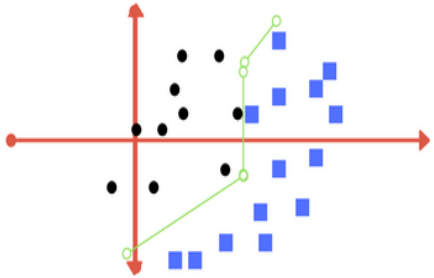
SVM Classifier

- **Kernels:**
  - **Linear Kernel:** linear SVM is done by transforming the problem using some linear algebra.
  - **RDF Kernel:** Radial Basis kernel is a kernel is a distance-based kernel.
  - **Polynomial Kernel:** calculate separation line in higher dimensions

- **Regularization:**



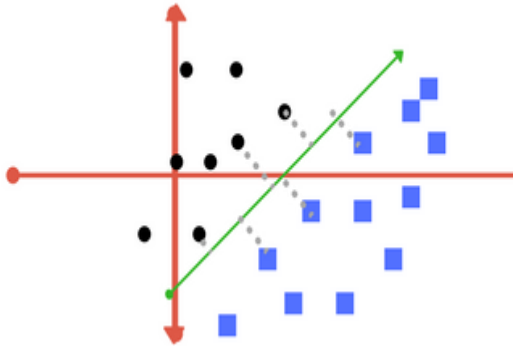
Low Regularization- optimizer will look for a larger margin separating plane



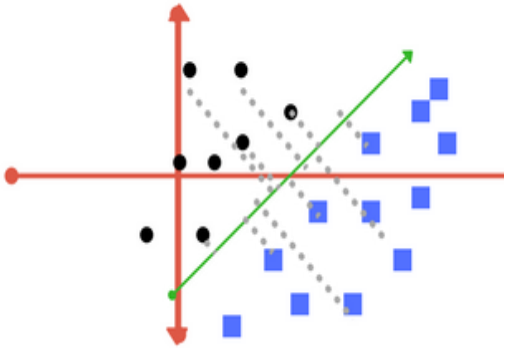
High Regularization- optimizer will look for a smaller margin separating plane, better working to classify training point

SVM Classifier

- **Gamma:** defines the reach of influence of single training example. calculations of separation line is based near-by point (High Gamma) or far away points (Low Gamma).



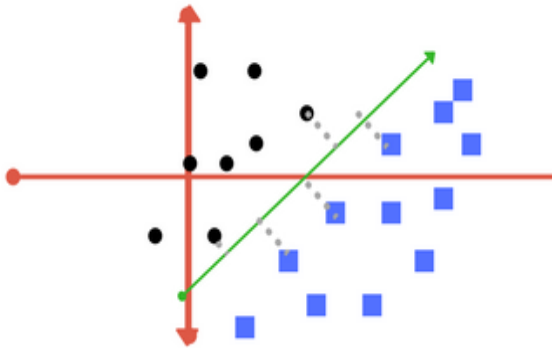
High Gamma
Nearby points are considered



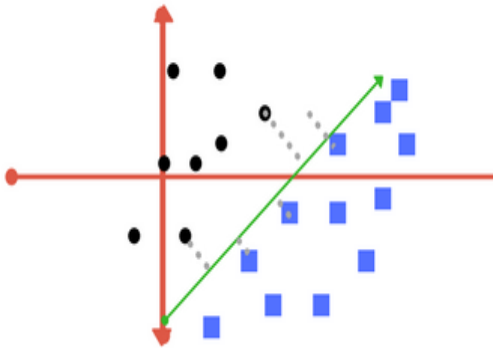
Low Gamma
Far away point are also considered

SVM Classifier

- **Margin:** A margin is a separation of line to the closest class points.



Good Margin
Equidistant from
both classes



Bad Margin
Close to Blue Class

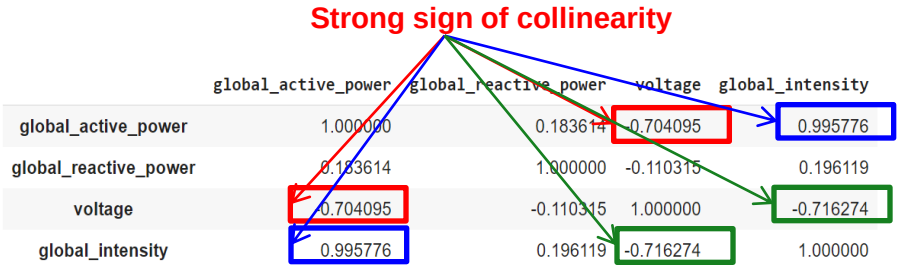
SVM Classifier Exercise and Python Code

Useful Concepts: Regression & Classification

Dr. Anuja Arora, Jaypee Institute of Information Technology

Multicollinearity

- **Multicollinearity:** There should not be high correlation between two or more independent variables. Multicollinearity can be checked **using correlation matrix, Tolerance and Variance Influencing Factor (VIF).**
- **Multicollinearity** refers to a situation in which more than two explanatory variables in a model are highly linearly related.
- We have perfect **multicollinearity** if, for example as in the equation above, the correlation between two independent variables is equal to **1 or −1 in correlation**.
- **Variance inflation factors (VIF)** measures how much the variance of the estimated regression coefficients are inflated as compared to when the predictor variables are not linearly related.



	global_active_power	global_reactive_power	voltage	global_intensity
global_active_power	1.000000	0.183614	-0.704095	0.995776
global_reactive_power	0.183614	1.000000	-0.110315	0.196119
voltage	-0.704095	-0.110315	1.000000	-0.716274
global_intensity	0.995776	0.196119	-0.716274	1.000000

Using correlation Matrix

```
[ ]  r_squared = r2_score_from_scratch(y, y_pred)
     print(r_squared)

     0.49574957962036914

[ ]  VIF=1/(1-r_squared)
     print(VIF)

     1.9831416288104198
```

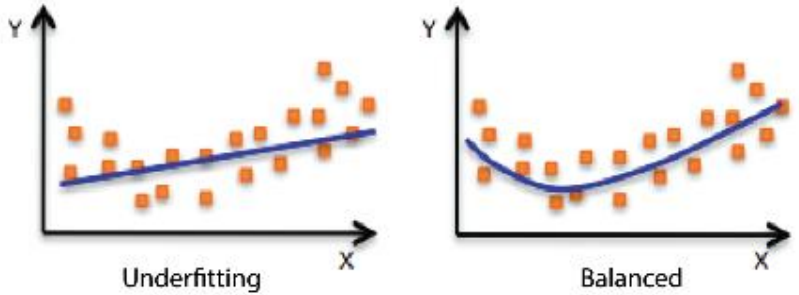
Moderately Correlated

Using VIF

Multicollinearity Python Code

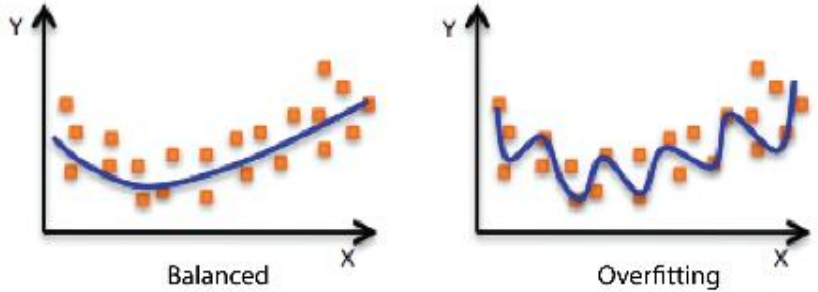
Underfitting and Overfitting

- **Optimised and best-fit line** describes the impact of the change in the independent variable on the change in the dependent variable by keeping the error term minimum.
- Bad performance while fitting model can be due to **Underfitting or Overfitting**.
- **Underfitting:**
  - The under-fitted model leads to low accuracy of the model and model is unable to capture the relationship, trend or pattern in the training data.
  - Underfitting of the model could be avoided by using more data, or by optimising the parameters of the model.



Underfitting and Overfitting

- **OverFittting:**
  - Overfitting is the opposite case of underfitting, i.e., when the model predicts very well on training data and is not able to predict well on test data or validation data.
  - The main reason for overfitting could be that the model is memorising the training data and is unable to generalise it on test/unseen dataset.
  - Overfitting can be reduced by doing feature selection or by using regularisation techniques.



Regularization

- Overfitting of model can be avoided by various process such as cross-validation sampling, reducing number of features, regularization etc.
- Regularization basically adds the penalty as model complexity increases.
- Regularization parameter (lambda) penalizes all the parameters except intercept so that model generalizes the data and won't overfit.
- **L1 Regularization:** A regression model that uses L1 regularization technique is called ***Lasso Regression.*** Uses **"absolute value of magnitude"** of coefficient as penalty term to the loss function.
- **L2 Regularization:** A regression model that uses L1 regularization technique is called **Ridge *Regression.*** Uses "**Squared magnitude**" of coefficient as penalty term to the loss function.

Ridge Regression Vs. Lasso Regression Vs. OLS

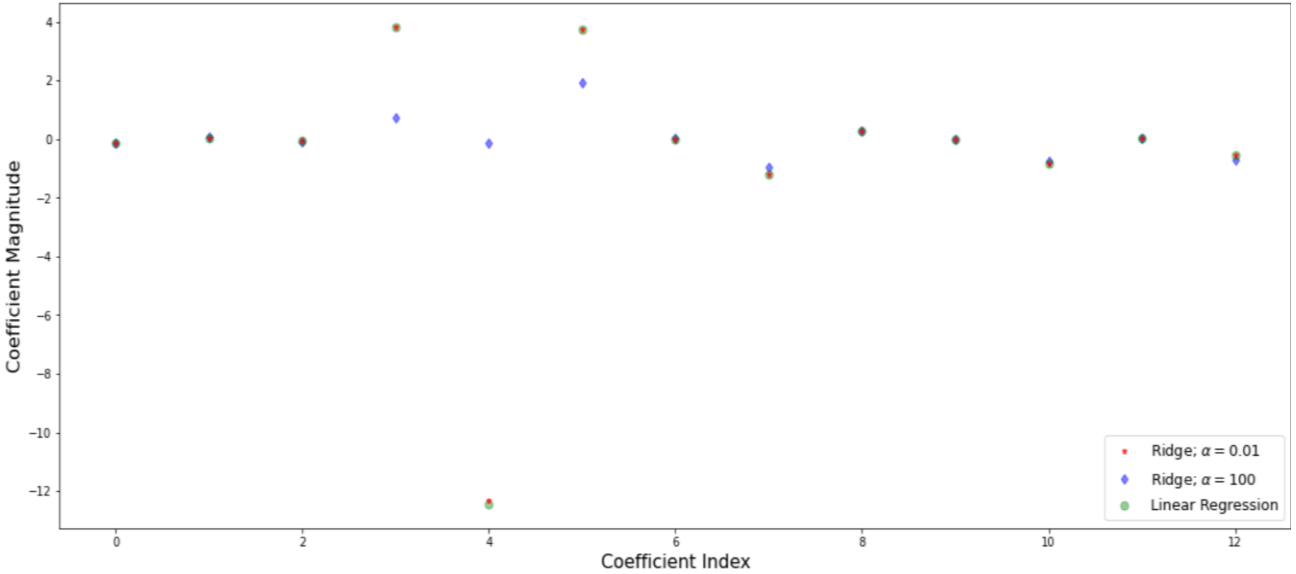
Ridge Regression	Lasso Regression	Ordinary least Square (OLS) Regression
"*squared magnitude*" of coefficient as penalty term to the loss function.	"absolute value of magnitude" of coefficient as penalty term to the loss function.	OLS is the method used to find the simple linear regression of a set of data. Basically it is a type of linear regression
$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$	$\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \lvert\beta_j\rvert$	If λ is Zero, i.e OLS. if λ is very large then it will add too much weight
This technique works very well to avoid over-fitting issue.	*Lasso shrinks the l coefficient to zero thus, removing some feature altogether. So, works well for Feature selection.*	To make error least. It will lead to under-fitting.
L2 Regularization	*L1 Regularization*	

Ridge and Lasso Regression Python Code

Ridge Linear Regression

- Ridge regression is a model tuning method that is used to analyse any data that **suffers from multicollinearity**.
- This method performs **L2 regularization**.
- When the issue of multicollinearity occurs, **least-squares are unbiased**, and **variances are large**, this results in predicted values to be far away from the actual values.

Ridge Vs. Linear Regression



- For low value of Y (0.01), the coefficients are restricted, the magnitudes of the coefficients are almost same as linear regression
- For higher value of Y (100), we see that for coefficient indices 3, 4, 5 the magnitudes are considerably less compared to linear regression case. This is an example of shrinking coefficient magnitudes using Ridge regression

Lasso Vs. Linear