Model Optimization

Model Optimization

In this session we shall cover:

- Bias and Variance
- Overfitting and underfitting
- Model Validation
- Gradient Descent
- Regularization
    - Lasso
    - Ridge
- Grid Search
- RandomizedSearch CV

Optimization

Optimization

OPTIMIZATION

- Prediction Evaluation
- Model Validation
- Fine Tuning

- Prediction Evaluation: Process of evaluating how effectively the constructed model performs predictions
- Model Validation:Using test data to validate the model built using train data
- Fine Tuning: Maximizing the performance of a constructed model

Prediction Evaluation

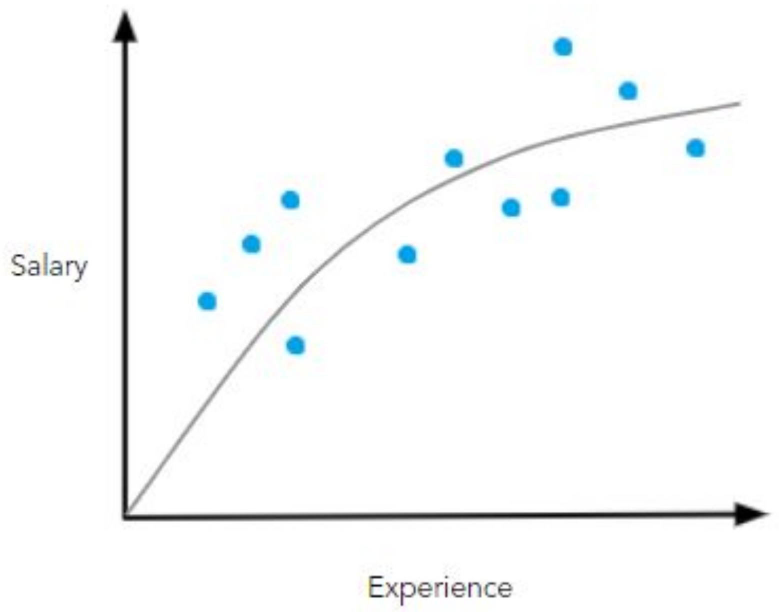
Prediction Evaluation

To construct a model with high prediction efficacy it is important to consider the prediction errors:

- Bias
- Variance

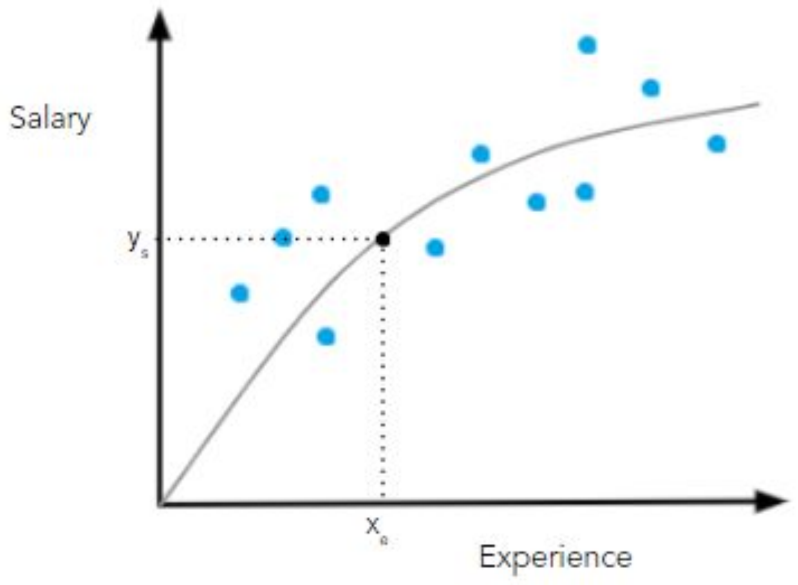
Bias and Variance

Bias and variance



- Consider the example of influence of years of experience on salary
- The plot represents a relationship between salary and experience
- Let us assume a grey curve that captures the true trend for the points in the plot

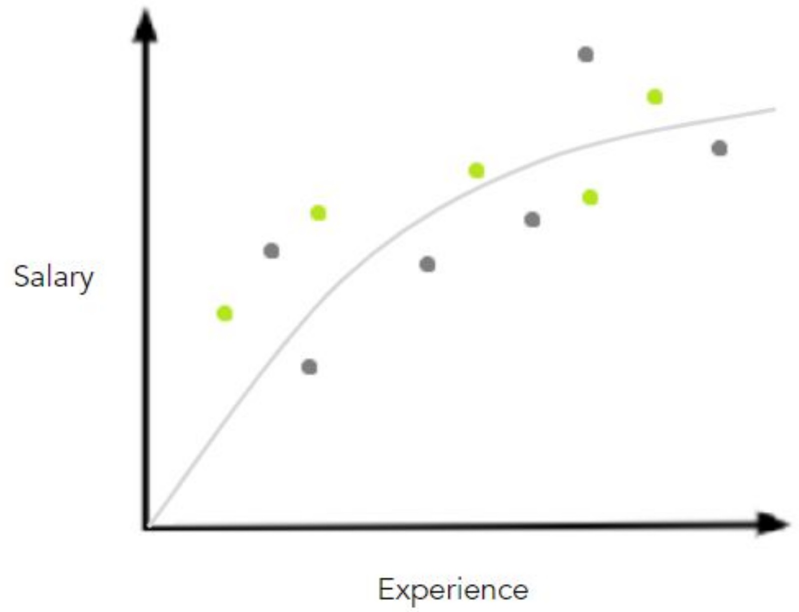
Bias and variance



Given the years of experience information (x_e), we can determine the salary (y_s) using the grey curve

But we do not actually know the grey curve for this plot

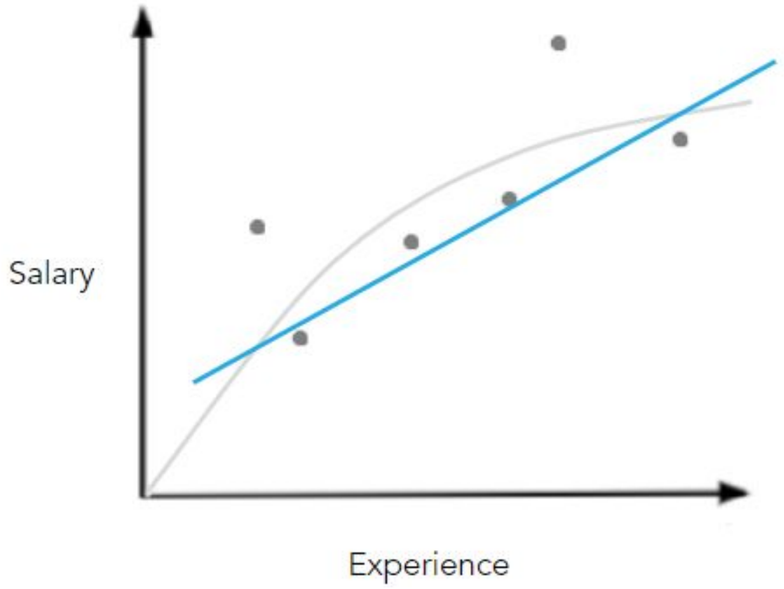
Bias and variance



To find the curve that captures the true trend we divide the data into :

- Train data
- Test data

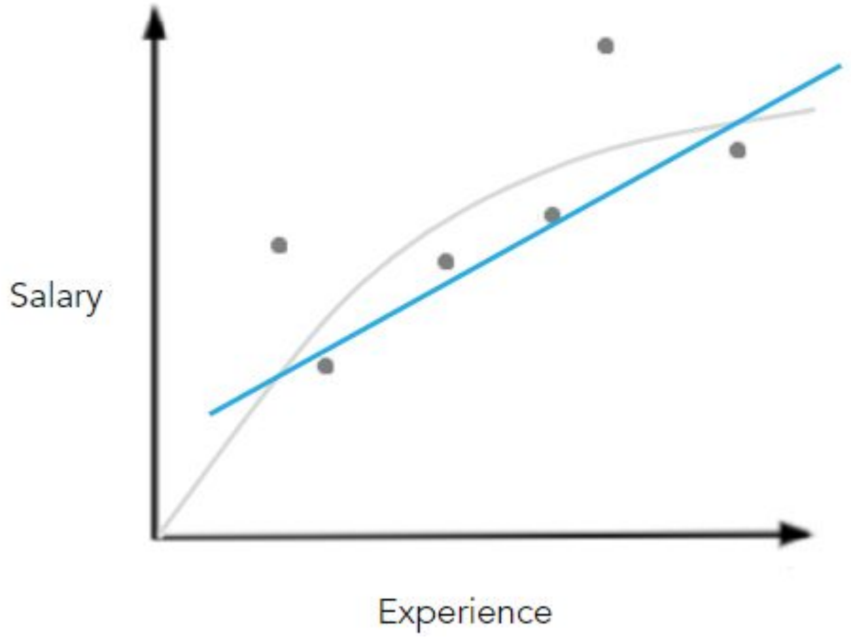
Bias and variance



- We first estimate a regression line to capture the trend in the train data

- But compared to the line, the grey curve seems to better capture the relationship between experience and salary

Bias

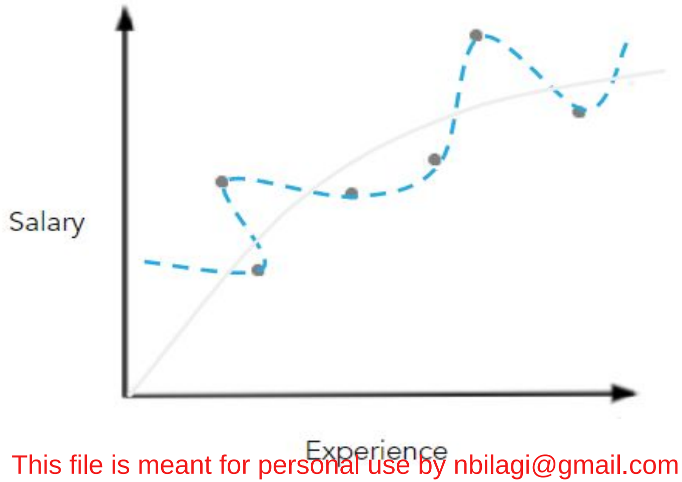


- The linear regression line will never bend and hence will never capture the “true” relationship

- This inability to capture the “true” relationship is called bias

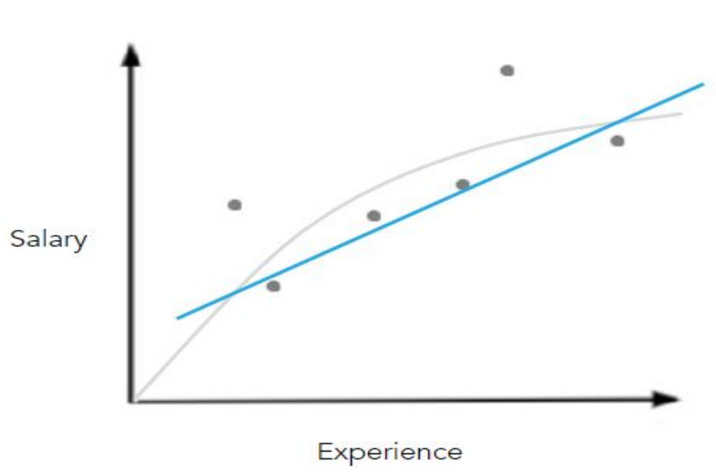
Bias and variance

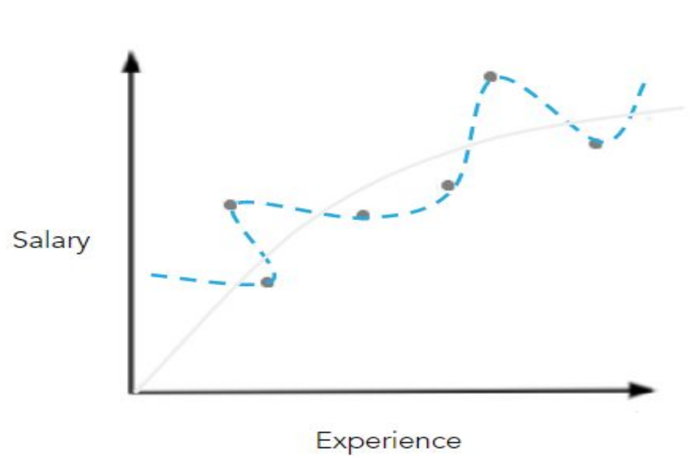
We then estimate a blue curve that captures the trend in train data perfectly, even better than the grey curve



Error calculation

Error is measured by adding the squares of difference between the actual and the fitted values.

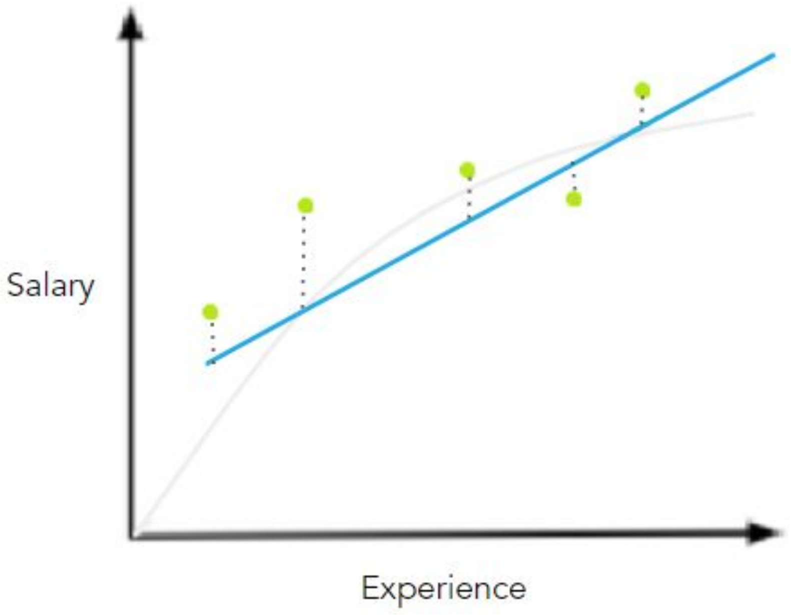


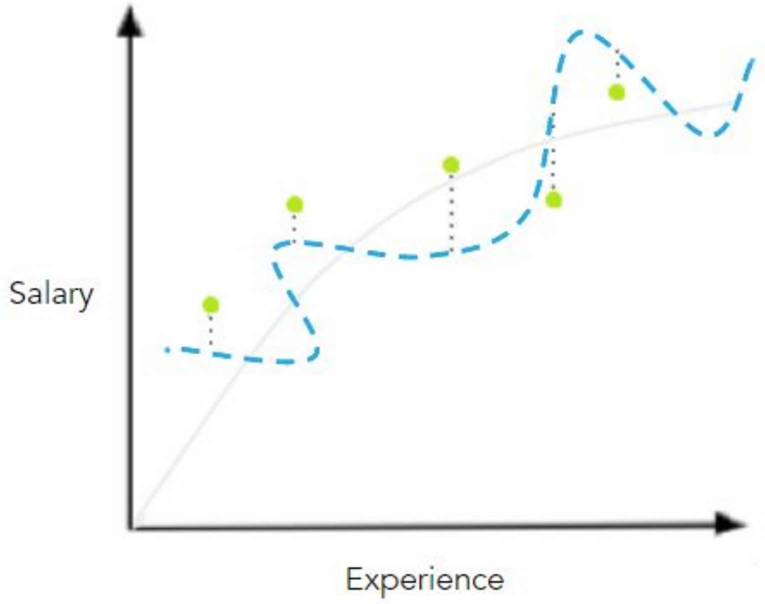


The curve fits the data points so perfectly, the difference between the actual and fitted values is actually zero.

Bias and variance

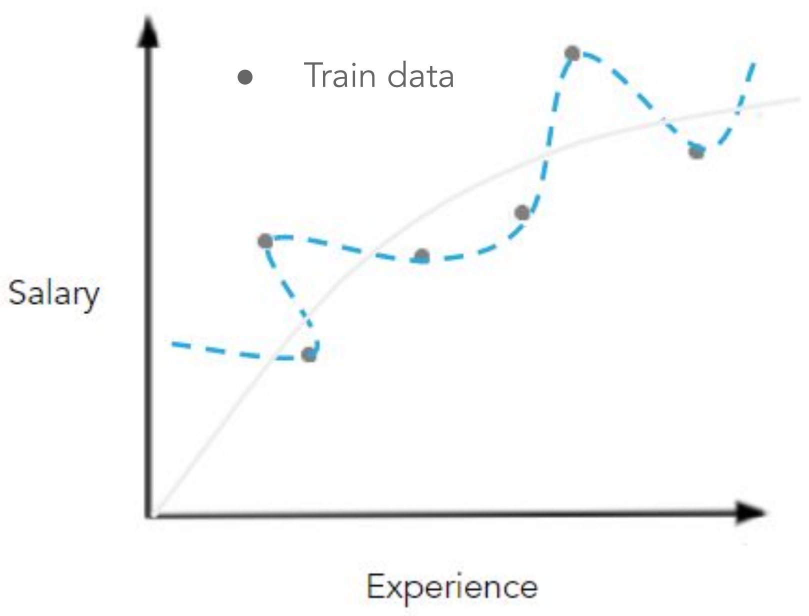
We use the same blue line and blue curve to estimate trends in test data

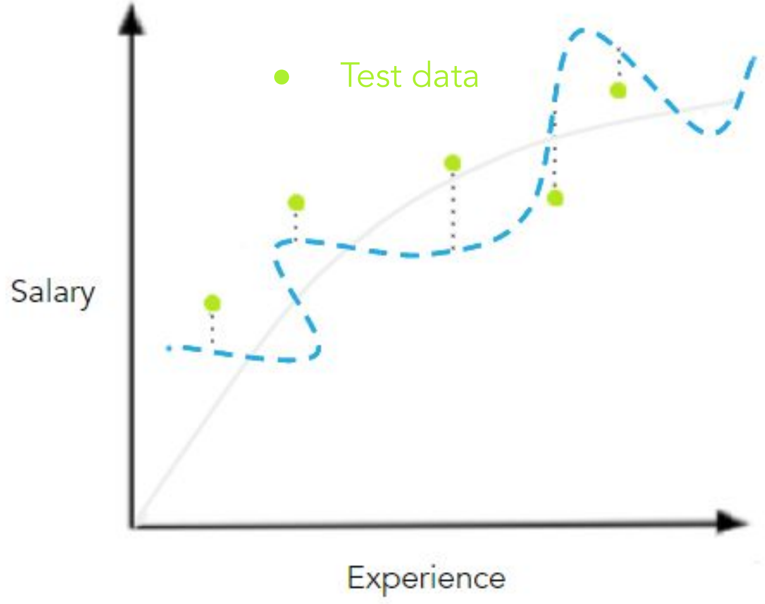




Bias and variance

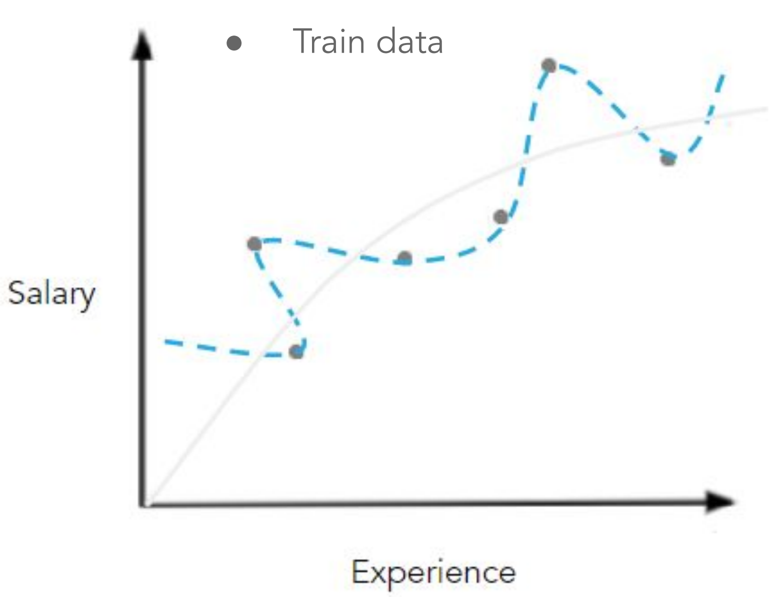
Even though the blue curve fits the train data with zero error, it does not predict well on test data

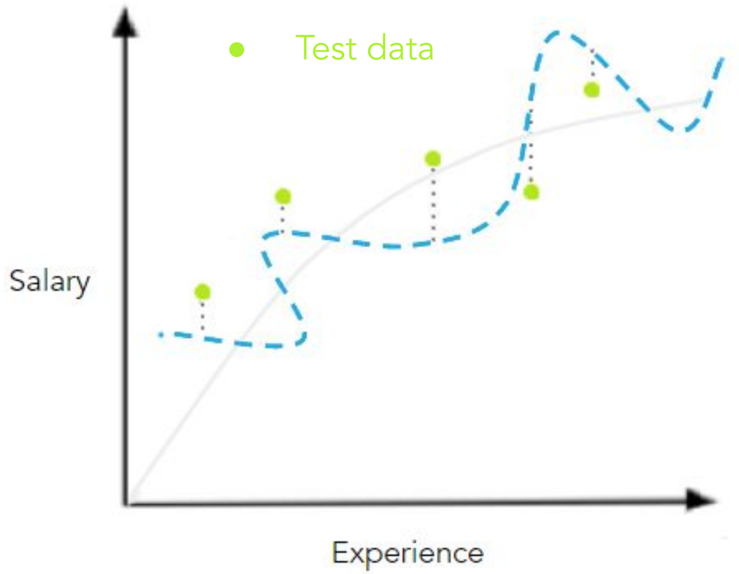




Variance

This difference in fits is called variance





Bias and variance

- Bias is the difference between a model’s predicted values and the observed values

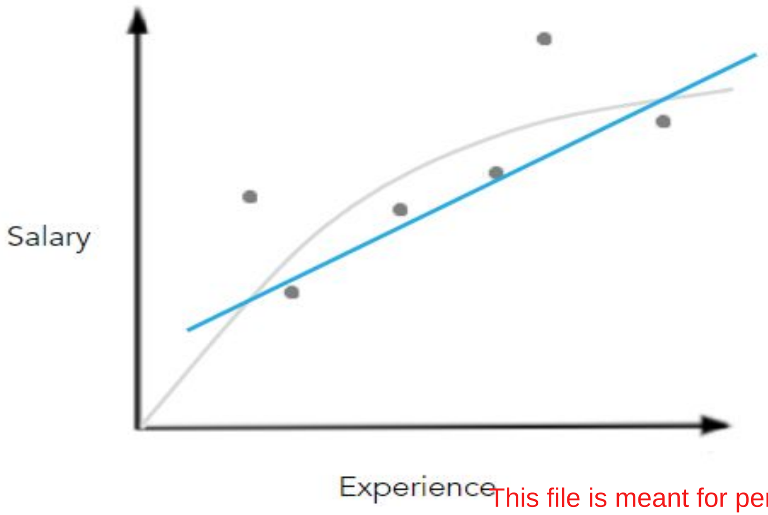
- Variance of a model is the difference between predictions if the model is fit to different datasets

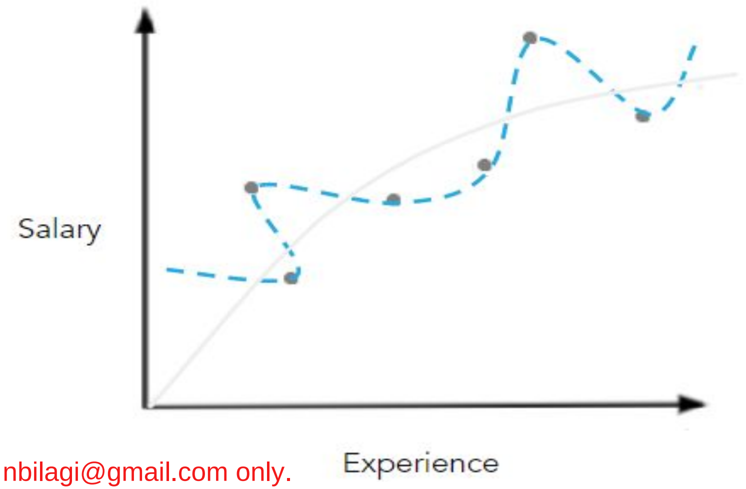
Bias-Variance for a simple model

- If the model is too simple it will have a high bias and low variance
- Such a model will give not perfectly accurate predictions, but the predictions will be consistent
- The model will not be flexible enough to learn from majority of given data, this is termed as underfitting

Example for bias

As we can see compared to the blue line, the blue curve captured the trend in train data perfectly. Hence we can say that the blue line has a high bias.





Bias-Variance for a complex model

- If the model is too complex it will have a low bias and high variance
- Such a model will give accurate predictions but inconsistently
- The high variance indicates it will have a much better fit on the train data compared to the test data, this is termed as overfitting

Model Validation

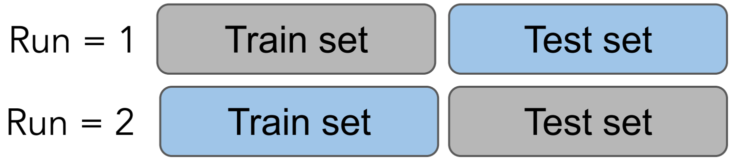
Model validation

- The model validation methods use test data to validate the model built using train data

- The model validation:
    - k - fold cross validation
    - Leave one out cross validation (LOOCV)

Cross validation

- Procedure:
    - Consider a data having ‘2n’ observations
    - Partition the dataset into two subsets: train and test sets of the equal size (n)
    - Measure the model performance
    - Swap the train and test sets
    - Total error is obtained by summing up the errors for both runs



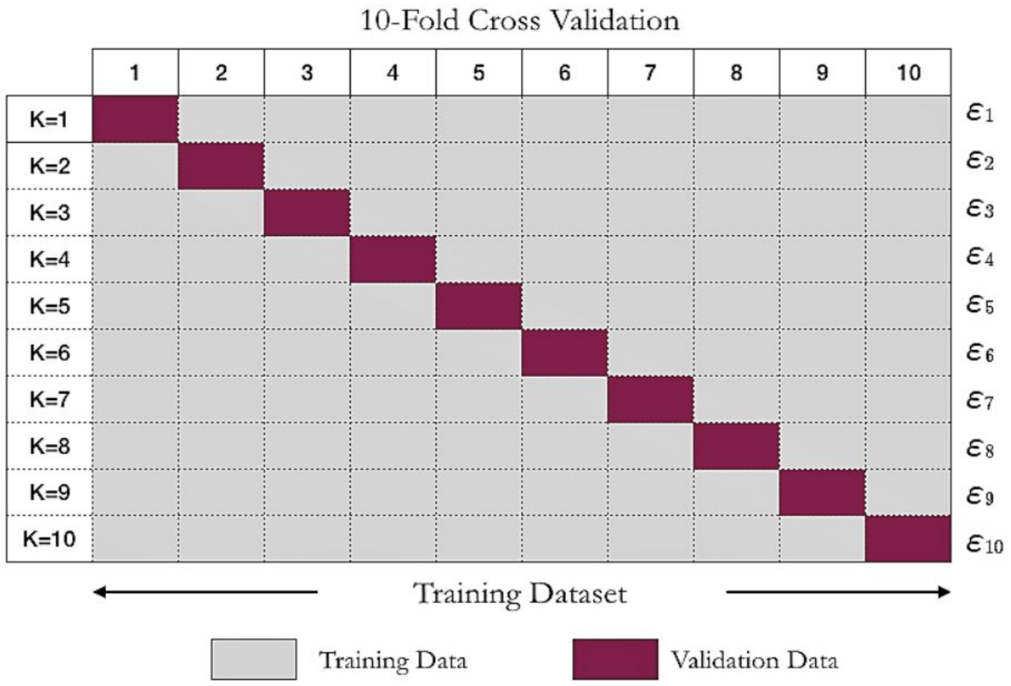
- This method is known as two fold cross validation
- Here, each observation is used exactly once for training and once for testing

The k - fold cross validation

- Procedure:
  - Partition the dataset into ‘k’ subsets
  - Consider one subset as the test set and remaining subsets as train set
  - Measure the model performance
  - Repeat this until all k subsets are considered as test set
  - Total error is obtained by summing up the errors for all the k runs
- This method is known as the k - fold cross validation
- Here, each observation is used exactly k times for training and exactly once for testing

10 - Fold cross validation

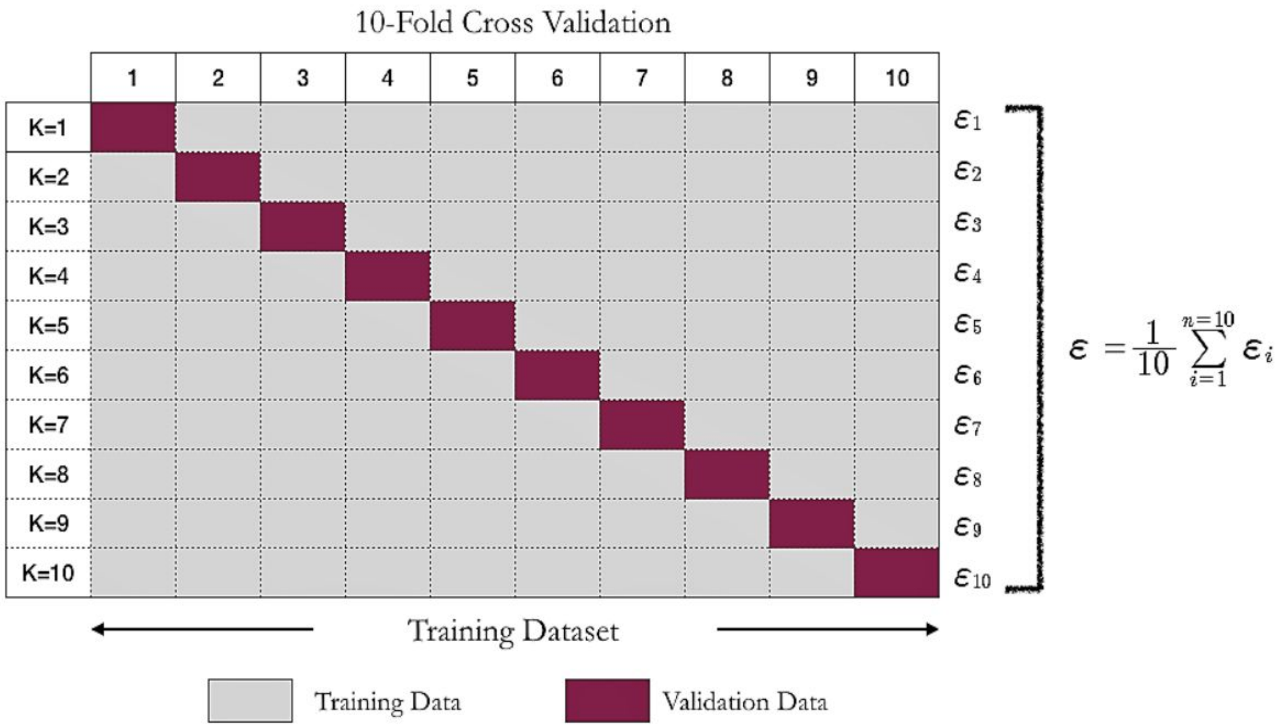
The split of training and test for each run.



And the total error is given by:

$$\varepsilon = \frac{1}{10}\sum_{i=1}^{n=10} \varepsilon_i$$

Cross validation



LOOCV (leave one (record) out cross validation)

- It is a special case of k - fold cross validation method. Instead of subsetting the data, at every run one observation is considered as the test set
- For n observations, there are n runs
- The total error is the sum of errors for n runs
- In LOOCV, the estimates from each fold are highly correlated and their average can have a high level of variance

Gradient Descent

$$\underbrace{\text{Gradient}}_{\text{Slope}}\ \underbrace{\text{Descent}}_{\text{Moving Downwards}}$$

What is a cost function?

- A cost function tells how good the model performs at making predictions for a given set of parameters

- Cost function = Loss function = Error function

- For linear regression, the cost function is given by the sum of squares of residuals, i.e.

$$Error = \sum_{i=1}^{n} (y_{act} - y_{pred})^2$$

where y_{act} is the actual value and y_{pred} is the predicted value

The gradient descent

- The gradient descent is an optimization technique which finds the parameters such that the error term is minimum

- It is an iterative method which converges to the optimum solution

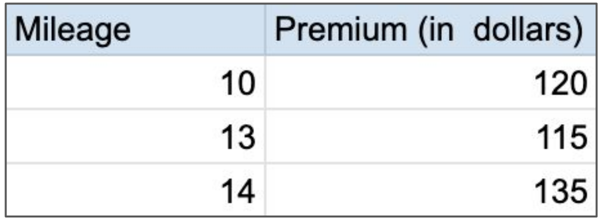
- It takes large steps when it is away from the solution and takes smaller steps closer to the optimal solution

- The estimates of the parameter are updated at every iteration

Let us consider an example

- We consider the same example that we had for simple linear regression
- However, we shall consider only three observations
- In context with our example,

$$\text{Premium} = \beta_0 + \beta_1 \text{ Mileage} + \varepsilon$$



Mileage	Premium (in dollars)
10	120
13	115
14	135

Let us consider an example

- We shall first estimate for the intercept β_0 and assume a value for β_1

- Let $\beta_1 = 2.5$. This value will be constant in all the iterations

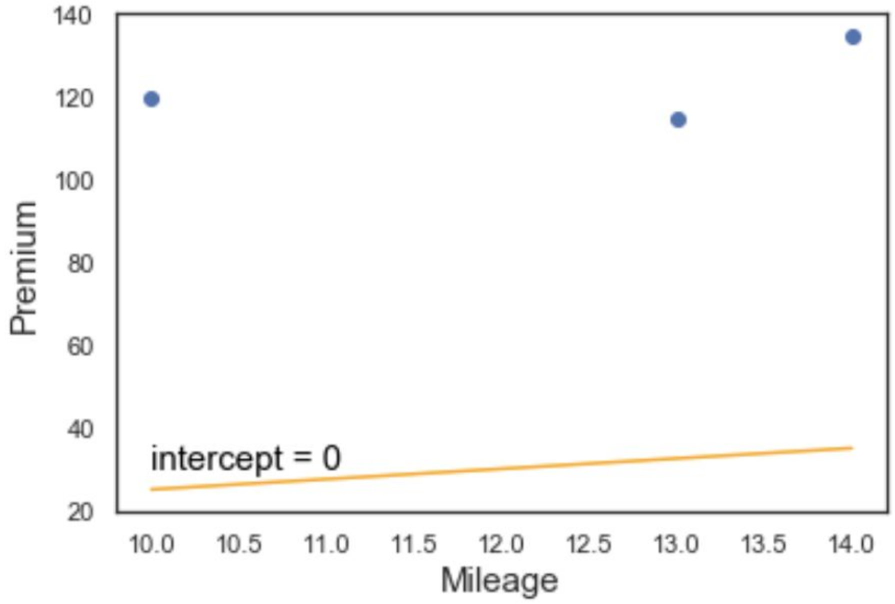
- In context with our example,

$$\text{Premium} = \beta_0 + 2.5\ \text{Mileage} + \varepsilon$$

Mileage	Premium (in dollars)
10	120
13	115
14	135

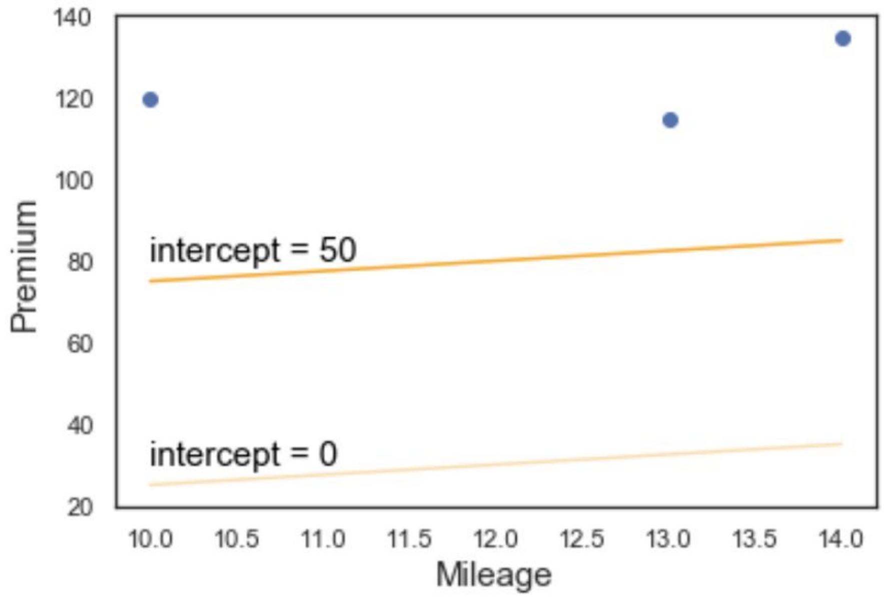
Example: gradient descent

- To estimate β_0, start with some initial value. Let $\beta_0 = 0$
- The line with intercept = 0 is obtained as shown
- We see it is far away from the data points, naturally the cost function value will be high
- The value cost function is 25831.25



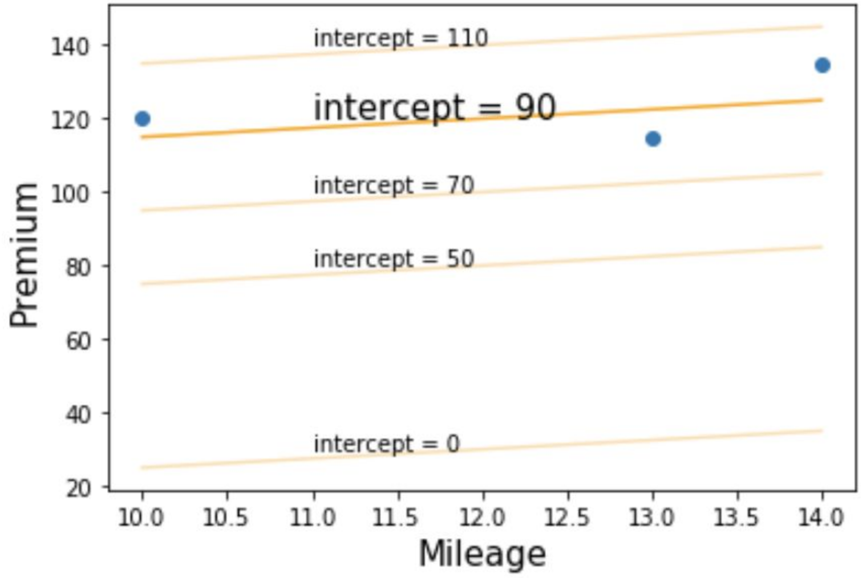
Example: gradient descent

- Now we increase the value of β_0 to 50
- The line with intercept = 50 obtained as shown
- The line has moved towards the data points as a result will have a lower cost function value
- The value cost function drops to 5581.25

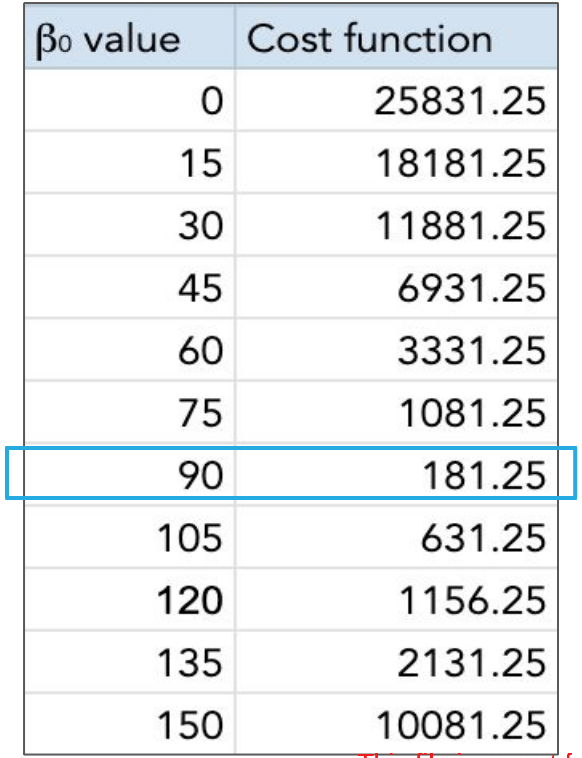


Example: gradient descent

- We continue to increase until we obtain the lowest cost function
- The line with intercepts as 0, 50, 70, 80, 90, 110 and 150 are shown
- Note the line with intercept 90 is the closest to points
- For intercept = 90, the value cost function to 181.25



The optimal solution



β_0 value	Cost function
0	25831.25
15	18181.25
30	11881.25
45	6931.25
60	3331.25
75	1081.25
90	181.25
105	631.25
120	1156.25
135	2131.25
150	10081.25

Is β_0= 90 the optimal solution?

The optimal solution

β_0 value	Cost function
0	25831.25
15	18181.25
30	11881.25
45	6931.25
60	3331.25
75	1081.25
90	181.25
105	631.25
120	1156.25
135	2131.25
150	10081.25

Is $\beta_0 = 90$ the optimal solution?

Answer: Perhaps.

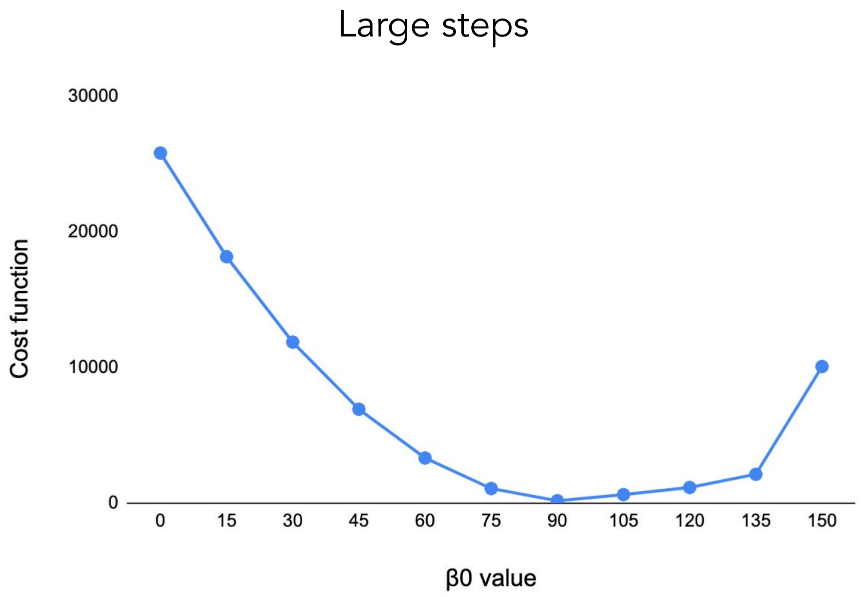
The optimal solution

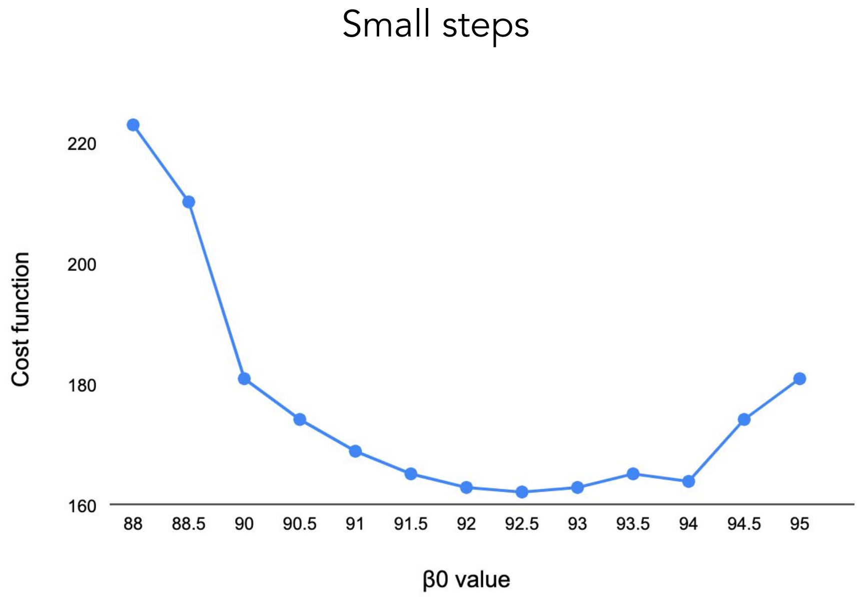
β_0 value	Cost function
88	223.25
88.5	210.5
90	181.25
90.5	174.5
91	169.25
91.5	165.5
92	163.25
92.5	162.5
93	163.25
93.5	165.5
94	164.25
94.5	174.5
95	181.25

- We now take small steps, around 90 to look for values which have minimum cost
- We see for β_0 = 92.5, the cost function is minimum
- In context with our example,

$$\text{Premium} = 92.5 + 2.5\ \text{Mileage}$$

The step size





Gradient descent

Now we have the following questions:

- What should be the step size?
- How did we know the value of the intercept is to be increased?

Gradient descent

Now we have the following questions:

- $\boxed{\text{What should be the step size?}}$

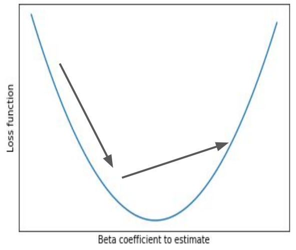
  The step size is the learning rate.

- How did we know the value of the intercept is to be increased?

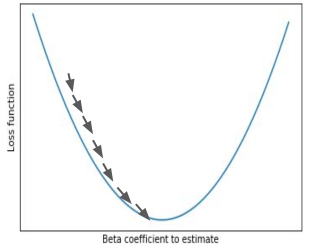
What is a learning rate?

- The gradient descent technique has a hyperparameter called learning rate, α
- It specifies the jumps the algorithm takes to move towards the optimal solution
- For very large α, the algorithm may skip the optimal solution and converges to suboptimal solution
- For very small α, the algorithm is more precise, however computationally expensive
- Thus, it is important to choose an appropriate learning rate

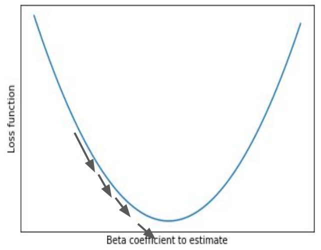
High learning rate



Low learning rate



Adequate learning rate



Gradient descent

Now we have the following questions:

- What should be the step size?

  The step size is the learning rate.

- How did we know the value of the intercept was to be increased?

  It is determined by the derivative of the cost function

Increase/decrease in the parameters

- The parameters are updated as

  New parameter = old parameters - (learning rate x derivative)

$$\theta_{new} = \theta_{old} - (\alpha . \delta)$$

- The learning rate is always a positive number

- The gradient descent computes the derivative of the cost function at each iteration. This derivative value determines the increase/decrease in the parameter

Gradient descent procedure

- Start with some initial set of parameters
- Compute the cost function
- The derivative of the cost function (delta; δ) is calculated
- Update the parameters based on learning rate α and derivative δ
- Repeat the procedure until the derivative of cost function is zero

Types of gradient descent

- Batch gradient descent
- Stochastic gradient descent
- Mini batch gradient descent

Batch Gradient Descent

$$\underbrace{\text{Batch}}_{\text{Complete Set}}\ \underbrace{\text{Gradient}}_{\text{Slope}}\ \underbrace{\text{Descent}}_{\text{Moving Downwards}}$$

Batch gradient descent

- The batch gradient descent computes the cost function with respect to the parmeter for the entire data
- Also known as vanilla gradient descent
- In spite of being computationally expensive, it is efficient and gradually converges to the optimal solution
- In the premium example (slides 9-14), we used the batch gradient descent

Stochastic Gradient Descent

$$\underbrace{\text{Stochastic}}_{\text{Random}}\ \underbrace{\text{Gradient}}_{\text{Slope}}\ \underbrace{\text{Descent}}_{\text{Moving Downwards}}$$

Stochastic gradient descent (SGD)

- For a data with many samples and many features the batch gradient descent is slow
- The SGD works efficiently for large data as it works with only a single observation at each iteration, i.e. this one sample is used to calculate the derivative
- Advantage of SGD is that we can add more data to the train set. The estimated new parameters is based on the recent estimates and previous
- Specially useful in presence of clustered data

Mini Batch Gradient Descent

$$\underbrace{\text{Mini}}_{\text{Small}}\ \underbrace{\text{Batch}}_{\text{Set}}\ \underbrace{\text{Gradient}}_{\text{Slope}}\ \underbrace{\text{Descent}}_{\text{Moving Downwards}}$$

Mini batch gradient descent

- It is a combination of both Batch gradient descent and SGD
- Like in SGD where one sample is considered, mini batch uses a group of samples and as in batch GD, all the sample are considered to obtain the cost function
- Hence it works faster than batch gradient descent and SGD
- Also known as vanilla mini batch gradient descent

To sum up...

	Batch GD	Mini Batch GD	Stochastic GD
Accuracy	High	Moderate	Low
Time required	High	Moderate	Low

Why gradient descent?

The number of updates required for the algorithm to converge will increase with the increase in the training data.

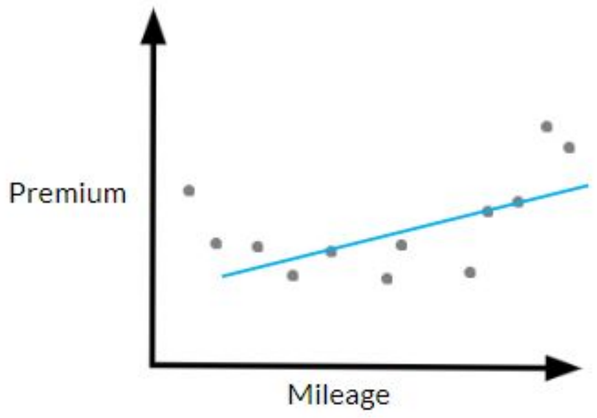
However, as the training data gets larger and larger, it is quite possible for the algorithm to converge much before every instance in the training data is learnt.

In other words, the increase in the training data size need not increase the training time needed to train the best possible model where the test error is at its least.

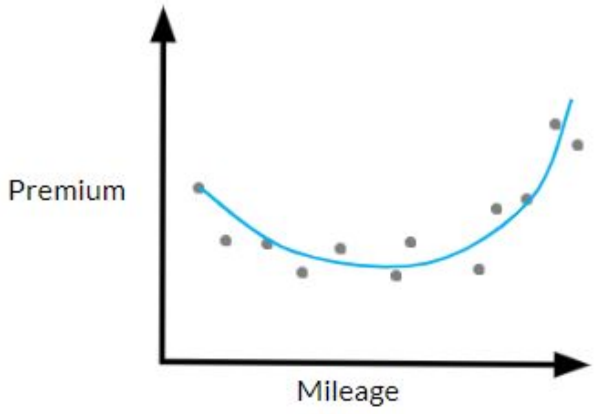
Revisiting Underfitting and Overfitting

Underfitting and overfitting

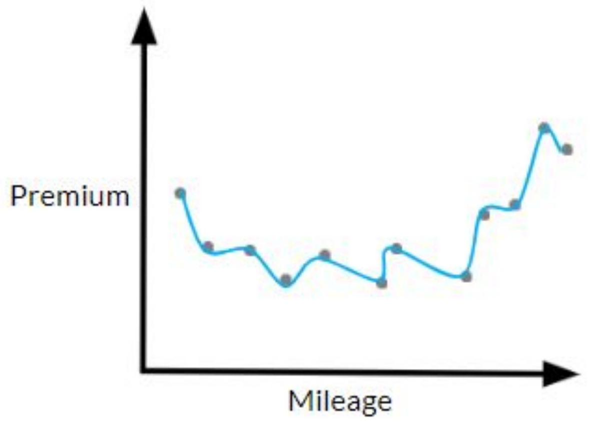
The plot that represents relationship between Premium and Mileage



Underfitted



Good fit/Robust



Overfitted

Overfitting

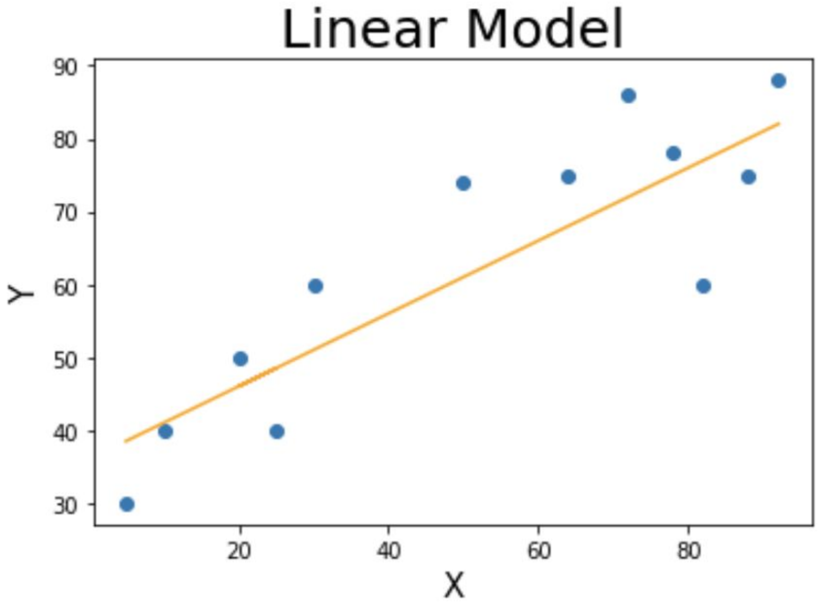
Generalization Error & Overfitting

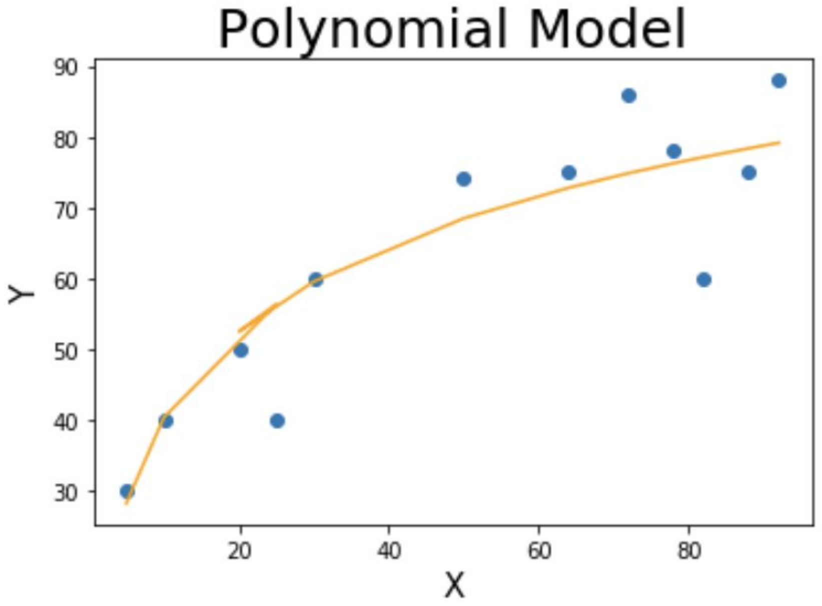
- If a model performs very well on the training data but does not perform well on the testing data, it is said to have high generalization error

- High generalization error implies overfitting

- Generalization error can be reduced by avoiding overfitting in the model

Polynomial Model





Prevention of overfitting for Linear regression

To make our model more robust and fix the problem of overfitting, we need to:

- Shrink the coefficients or weights of features in model
- Eliminate high degree polynomial feature from a polynomial model

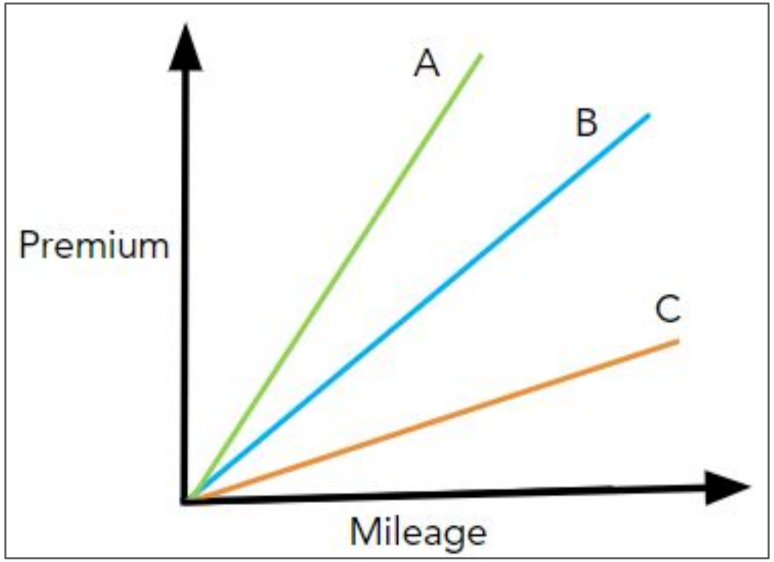
This can be achieved by using regularization.

Prevention of overfitting for Linear regression

To make our model more robust and fix the problem of overfitting, we need to shrink the coefficients or weights of features in model.

This can be achieved by using regularization.

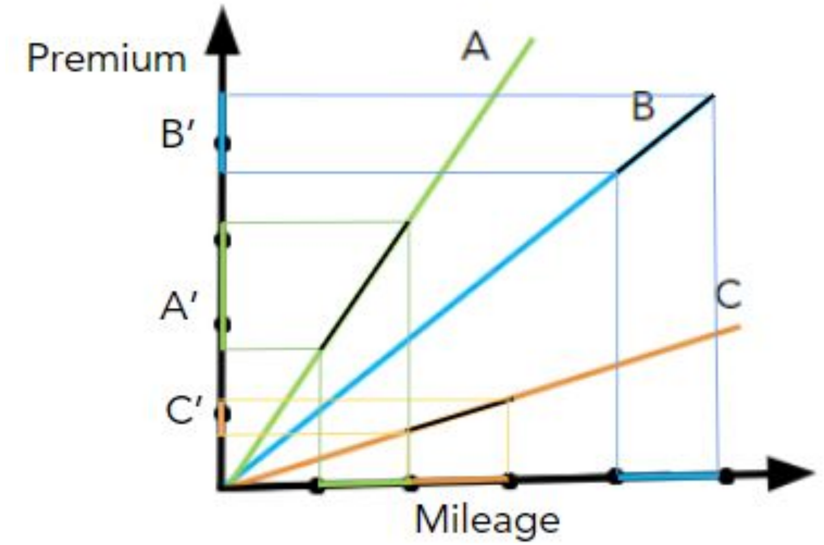
Example: Shrinking the β coefficients



Line A, B and C represent the relationships between Mileage and Premium for different cars.

We use them to study the influence of the regression coefficient on the target variable.

Example: Shrinking the β coefficients



$A' > B' > C'$

- A’, B’ and C’ represent the change in Premium per unit change in Mileage for the three different lines A, B and C respectively
- As we can see that as the slope decreases, the Premium become less sensitive to change in Mileage
- Thus, by reducing the sensitivity of the target variable with respect to the predictor variables, the bias increases

Why do we shrink the β coefficients?

- As we have seen from the Premium and Mileage example with decrease in slope, the dependent variable becomes less sensitive to change in independent variable
- Shrinking the slope introduces bias to the regression model
- With increase in bias, the variance decreases and the problem of overfitting can be fixed

Regularization

What is regularization?

- Regularization refers to the modifications we make to a learning algorithm, that help in reducing its generalization error but not its training error

- Regularization adds a penalty term to the cost function such that the model with higher variance receives a larger penalty

- It chooses a model with smaller parameter values (i.e. shrinked coefficients) that has less error

Regularization for linear regression

- On regularization, for linear regression there are two terms in the loss function
    - The OLS loss function
    - The penalty term

$$\text{Loss function}_{\text{regularization}} = \text{Loss function}_{\text{ols}} + \text{Penalty term}$$

- Regularization tries to balance the error from the loss function of OLS and the penalty term

Regularization for linear regression

- Regularization converges the beta coefficients of the linear regression model towards zero. This is known as shrinkage

- For linear regression the goal is to minimize,

$$\sum_{i=1}^{n} \left(y_{act} - y_{pred}\right)^2 + \text{penalty}$$

penalty = λ * w

λ = Regularization parameter

w = weight associated with the variables; generally considered to be the L-p norms

Regularization parameter

- Regularization parameter (λ) controls the strength of the penalty term
- If λ = 0, then there is no difference between a model with regularization and without regularization
- λ can take any values from 0 to infinity
- The best value for λ is determined by trying different values; the value that leads to least cross validation error is chosen

Types of regularization

- Ridge regression: Here, the w is the L2 norm
- Lasso regression: Here, the w is the L1 norm
- Elastic net regression: It is a combination of ridge and lasso regression

For more on L1, L2 norms refer the addendum provided

Ridge Regression

Ridge regression

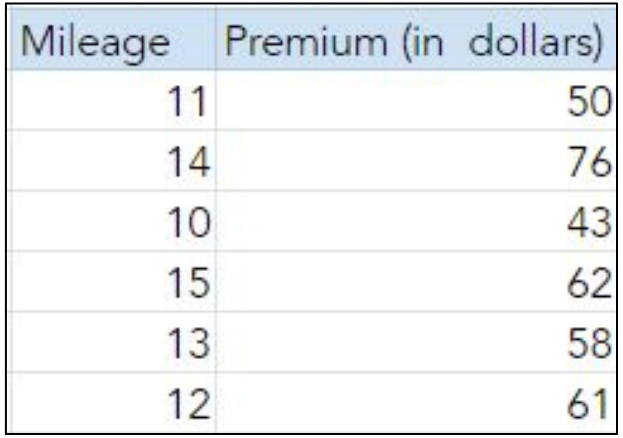
- Ridge regression uses squared L-2 norm regularization i.e it adds a squared L-2 penalty
- Also known as L-2 regularization
- Squared L-2 penalty is equal to squares of magnitudes of β coefficients
- It diminishes the insignificant predictors but does not completely eliminate them

$$\text{cost function} = \underbrace{\sum_{i=1}^{n}\left(y_{act} - y_{pred}\right)^2}_{\text{least squares regression error}} + \underbrace{\lambda \cdot ||w||_2^2}_{\text{ridge regression penalty}}$$

Let us consider an example

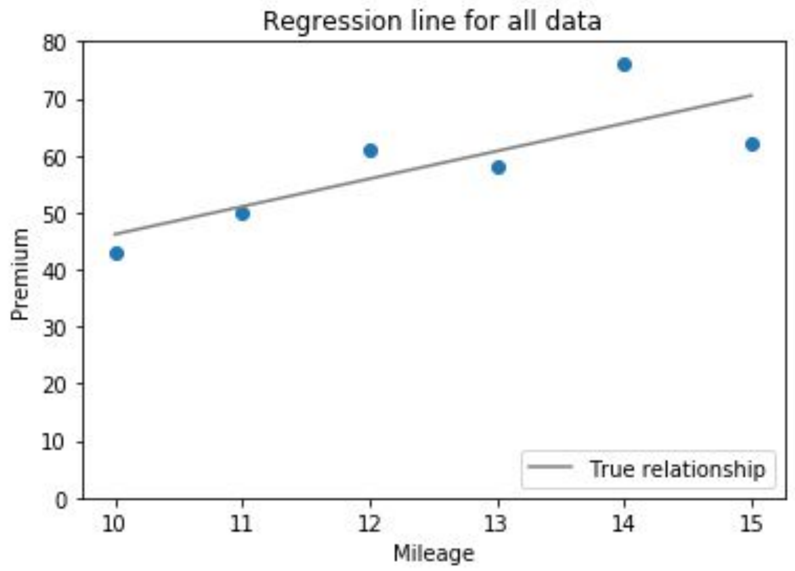
- We consider the same example as earlier
- However, we shall consider the six given observations
- In context with our example,

$$\text{Premium} = \beta_0 + \beta_1 \text{ Mileage} + \varepsilon$$



Mileage	Premium (in dollars)
11	50
14	76
10	43
15	62
13	58
12	61

Simple linear regression - OLS

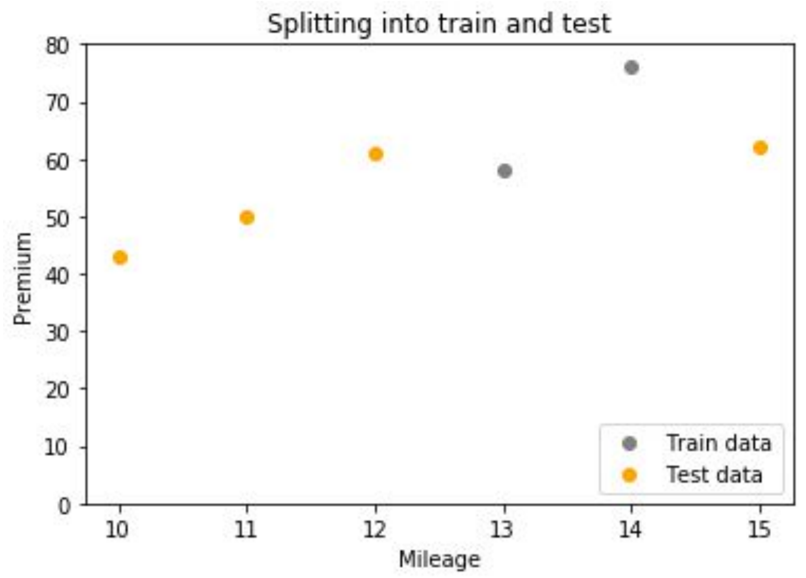


- We plot the graph of Premium against Mileage and estimate a regression line using all the data points, this captures the true relationship

- The equation is given by:

  Premium = -2.3809 + 4.8571 Mileage

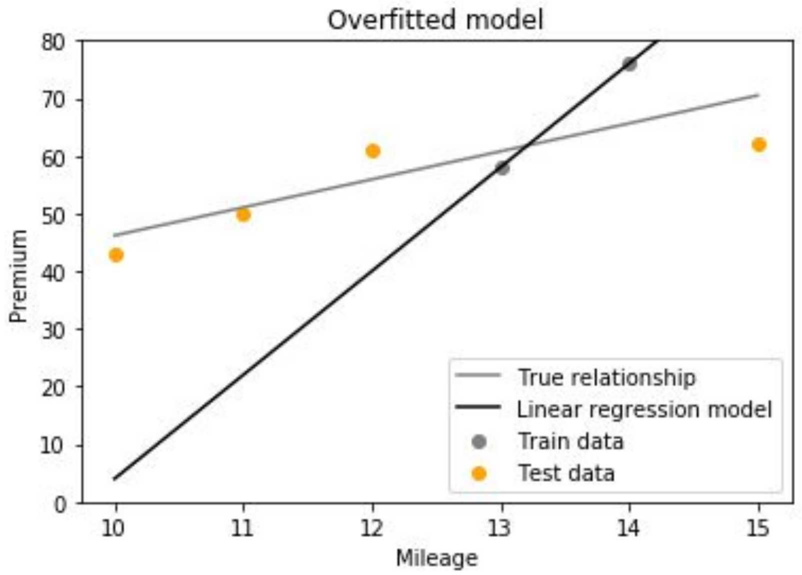
The train-test split



The entire data is split into:

- Train data
- Test data

Overfitted model

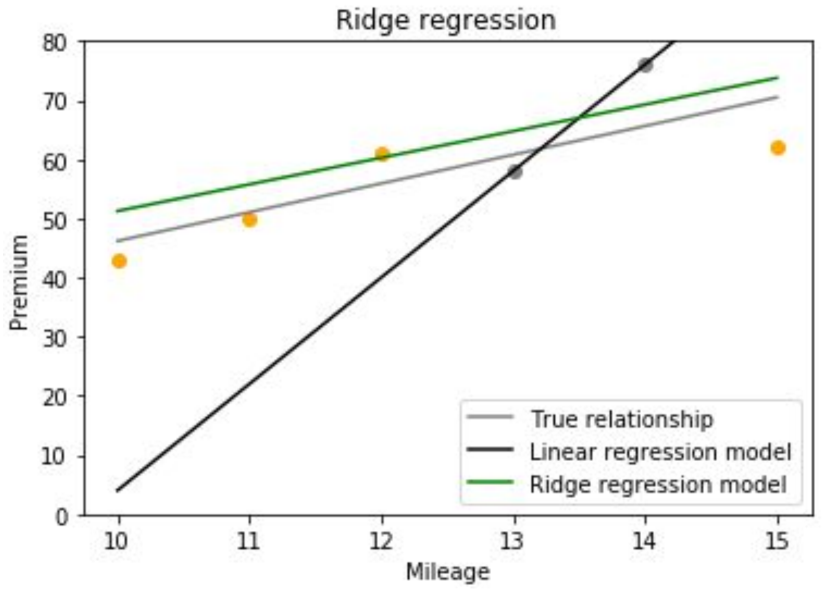


- Compared to the true relationship the black line performs very poorly on test data

- But it fits perfectly for train data, hence it is an overfitted model

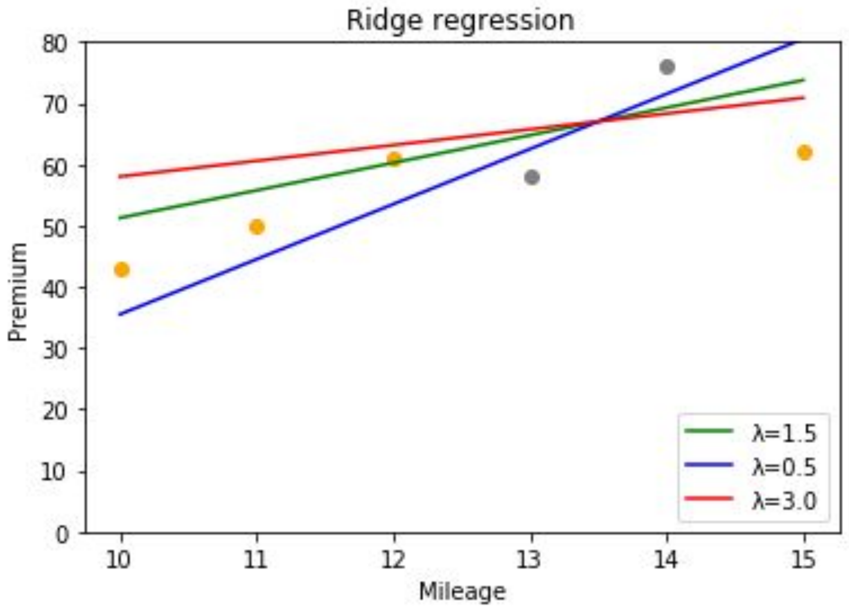
- The black line has very high variance

Ridge regression



- Using ridge regression we get green line
- Compared to the black line, the green line is much closer to the line that captures the true relationship
- Hence, the green line performs much better on the test data than the black line

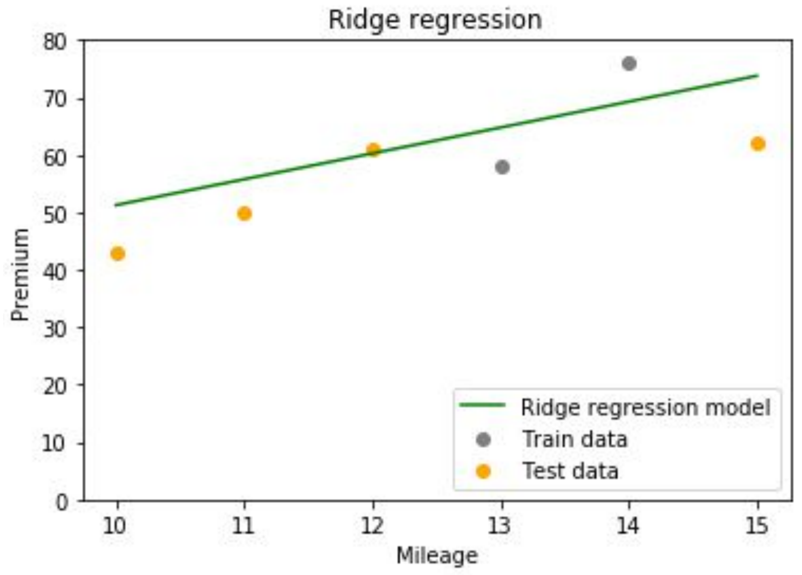
Regularization parameter



λ for model	1.5	0.5	3.0
RMSE	7.7419	11.0114	10.2444

When λ = 1.5, the RMSE value is least for test data, we choose this value

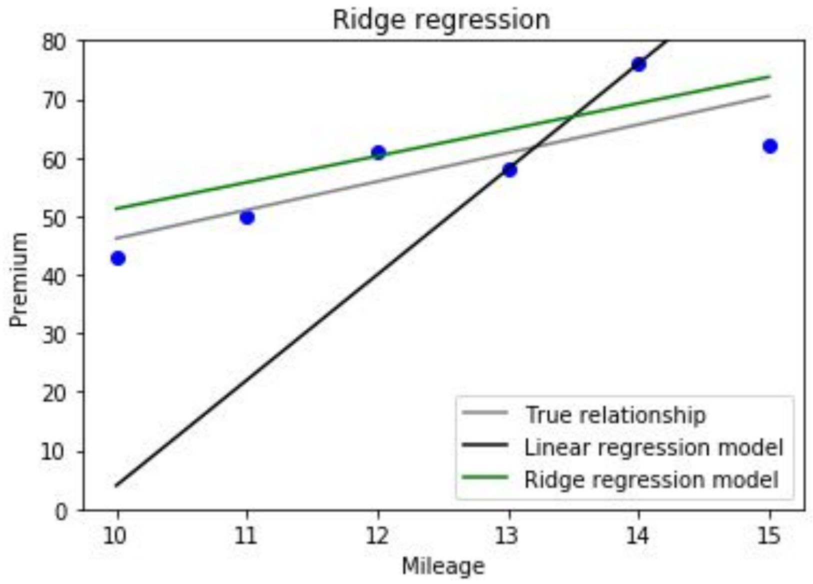
Ridge regression model



- This line introduces some bias to the model but decreases the variance
- The ridge regression line shrinks the β coefficients
- Equation of the green line is given by:

$$\text{Premium} = 6.25 + 4.6\ \text{Mileage}$$

Ridge regression penalty

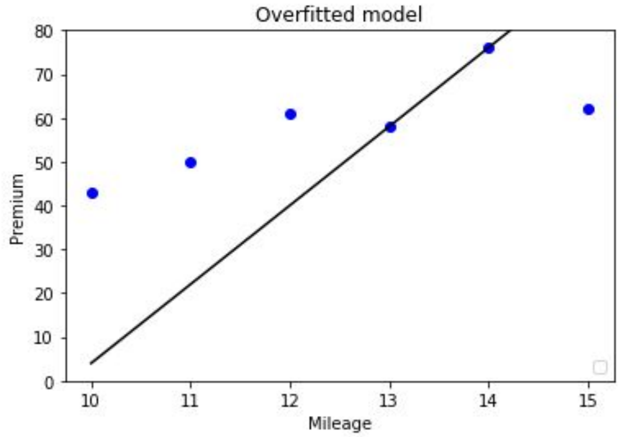


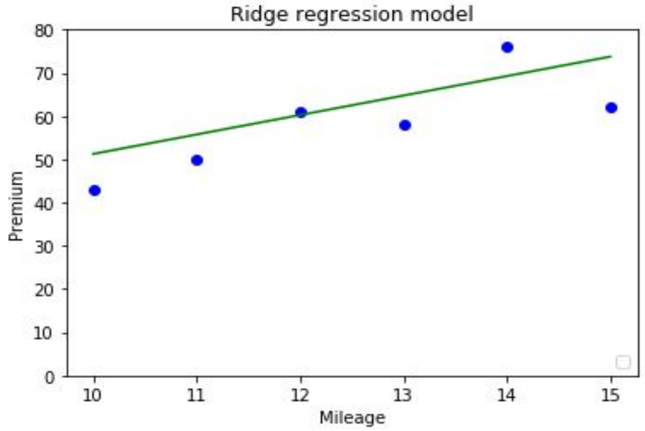
Ridge regression assigns a higher penalty to the model with higher variance

$\lambda * (\text{slope})^2$	$\lambda * (\text{slope})^2$
486	31.74

Here, $\lambda = 1.5$

Comparison of models





	β_0	$\beta_{mileage}$
β Coefficients	-176	18
Cost function	$\sum(\text{Actual Premium} - \text{Predicted Premium})^2$	

	β_0	$\beta_{mileage}$
β Coefficients	6.25	4.6
Cost function	$\sum(\text{Actual Premium} - \text{Predicted Premium})^2 + \lambda^* (\beta_{mileage})^2$	

Lasso Regression

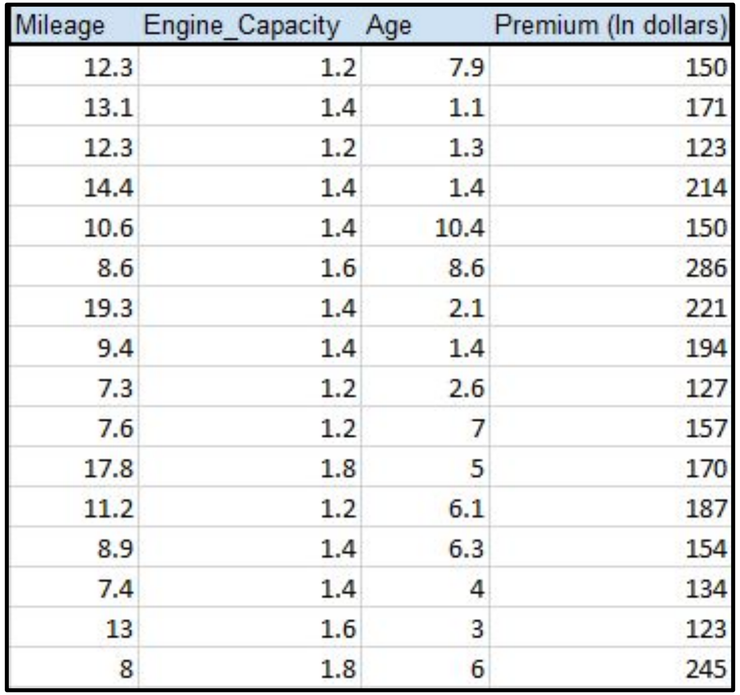
Lasso regression

- Lasso regression uses L-1 norm regularization i.e it adds a L-1 penalty
- Also known as L-1 regularization
- L-1 penalty is equal to absolute value of β coefficients
- It extinguishes the insignificant predictors

$$\text{cost function} = \underbrace{\sum_{i=1}^{n}\left(y_{act} - y_{pred}\right)^2}_{\text{least squares regression error}} + \underbrace{\lambda \cdot ||w||_1}_{\text{lasso regression penalty}}$$

Data

Let us consider the following data.

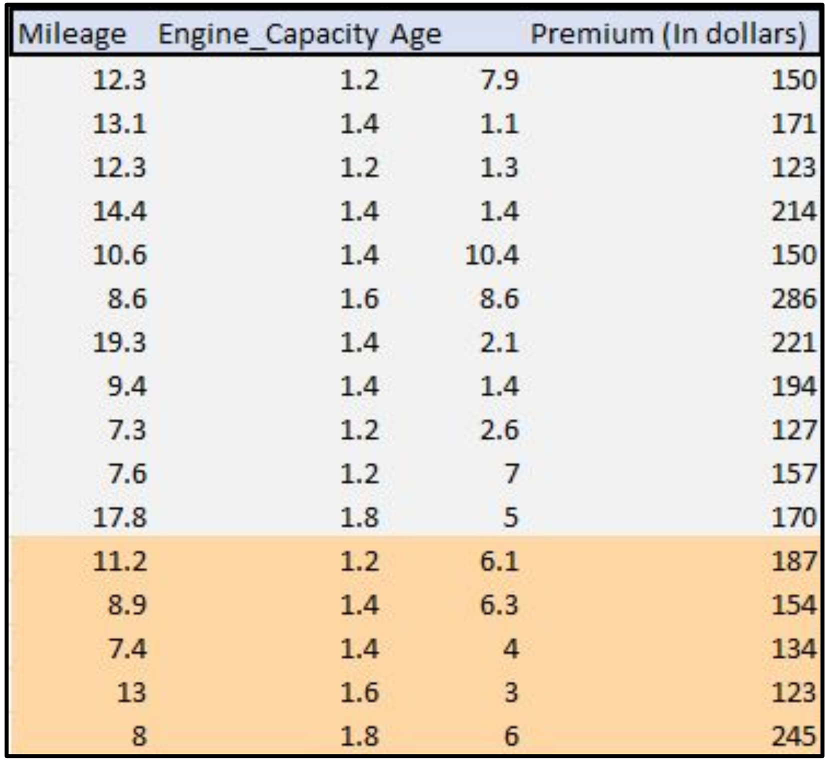


Mileage	Engine_Capacity	Age	Premium (In dollars)
12.3	1.2	7.9	150
13.1	1.4	1.1	171
12.3	1.2	1.3	123
14.4	1.4	1.4	214
10.6	1.4	10.4	150
8.6	1.6	8.6	286
19.3	1.4	2.1	221
9.4	1.4	1.4	194
7.3	1.2	2.6	127
7.6	1.2	7	157
17.8	1.8	5	170
11.2	1.2	6.1	187
8.9	1.4	6.3	154
7.4	1.4	4	134
13	1.6	3	123
8	1.8	6	245

The train-test split

We divide the data into :

- Train data
- Test data



Mileage	Engine_Capacity	Age	Premium (In dollars)
12.3	1.2	7.9	150
13.1	1.4	1.1	171
12.3	1.2	1.3	123
14.4	1.4	1.4	214
10.6	1.4	10.4	150
8.6	1.6	8.6	286
19.3	1.4	2.1	221
9.4	1.4	1.4	194
7.3	1.2	2.6	127
7.6	1.2	7	157
17.8	1.8	5	170
11.2	1.2	6.1	187
8.9	1.4	6.3	154
7.4	1.4	4	134
13	1.6	3	123
8	1.8	6	245

Feature scaling before regularization

The regularization parameter (λ) imposes a higher penalty on the variable with higher magnitude values

To avoid this, we scale all the variables within same range of values

For the considered example, since all the variables in the data do not follow a gaussian distribution we have used min-max normalization

Regression line - OLS method

- We consider the training data and construct a linear regression model to see how the features influence the Premium value

- The equation of estimated line is given by :

Premium = 0.2211 - 0.1177 Mileage + 0.5694 Engine_Capacity - 0.0187 Age

Comparison of RMSE values

- We check the train accuracy and test accuracy on train and test data respectively

	Train data	Test data
RMSE	0.2315	0.3139

- A good model that generalizes well needs to have very similar errors on train and test sets

- Here, the difference between errors for train and test sets is significant, hence we can conclude that the model is overfitting the train data

Lasso regression

- We construct a lasso regression model on train data
- The equation of estimated line is given by :

  Premium = 0.3368 - 0 Mileage + 0.0112 Engine_Capacity + 0 Age

- The lasso regression shrinks the β coefficients of variables Mileage and Age to 0, thus eliminating them from the final model
- This is an instance of how lasso regression performs feature selection

Comparison of RMSE values

- We check the train accuracy and test accuracy on train and test data respectively

	Train data	Test data
RMSE	0.2776	0.2754

- The difference between errors for train and test sets is very insignificant
- We can conclude that the lasso regression model performs better at generalization compared to least squares model

Lasso regression penalty

- Lasso regression assigns a higher penalty to the model with higher variance

Penalty	Least squares regression model	Lasso regression model
$\lambda*(\lvert\beta_0\rvert+\lvert\beta_{Mileage}\rvert+\lvert\beta_{Engine\_Capacity}\rvert+\lvert\beta_{Age}\rvert)$	0.0417	0.0157

- Here λ = 0.045, is calculated by trying different values

Comparison of models

Regression Model	β coefficients				Cost Function	RMSE	
	β_0	$\beta_{Mileage}$	$\beta_{Engine\_Capacity}$	β_{Age}		Train	Test
Least squares	0.2211	-0.1177	0.5694	-0.0187	$\Sigma(y_{act}-y_{pred})^2$	0.2315	0.3139
Lasso	0.3368	0	0.0112	0	$\Sigma(y_{act}-y_{pred})^2 + \lambda^*\Sigma\lvert\beta\rvert$	0.2776	0.2754

Elastic-net Regression

Elastic-net regression

- Elastic-net regression uses both L-1 and L-2 norm regularization

- Elastic-net regression is the combination of lasso and ridge regression

$$\text{cost function} = \underbrace{\sum_{i-1}^{n} \left(y_{act} - y_{pred}\right)^2}_{\text{least square regression error}} + \underbrace{\lambda_{\text{ridge}} \cdot ||w||_2^2}_{\text{ridge penalty}} + \underbrace{\lambda_{\text{lasso}} \cdot ||w||_1}_{\text{lasso penalty}}$$

PLEASE!! NOTE

L1 ratio

While implementing elastic net regression the regularization parameters are expressed in terms of λ and L1 ratio

$$\lambda = \lambda_{ridge} + \lambda_{lasso}$$

$$L1\_ratio = \frac{\lambda_{lasso}}{\lambda_{lasso} + \lambda_{ridge}}$$

	Penalty
L1_ratio = 0	L-2
L1_ratio = 1	L-1
0< L1_ratio <1	Combination of L-1 and L-2

Estimation of least squares regression

- We consider the same example considered for ridge regression

- Constructing a linear regression model on training data we get the following equation :

Premium = 0.2211 - 0.1177 Mileage + 0.5694 Engine_Capacity - 0.0187 Age

Comparison of RMSE values

- We check the train accuracy and test accuracy on train and test data respectively

	Train data	Test data
RMSE	0.2315	0.3139

- A good model that generalizes well needs to have very similar errors on train and test sets
- Here, the difference between errors for train and test sets is significant, hence we can conclude that the model is overfitting the train data

Elastic-net regression

- We construct a elastic-net regression model on train data
- The equation of estimated line is given by :

  Premium = 0.3370 - 0 Mileage + 0.0106 Engine_Capacity + 0 Age

- The elastic-net regression shrinks the β coefficients of variables Mileage and Age to 0, thus eliminating them from the final model

Comparison of RMSE values

- We check performance of the elastic-net regression model trained on training data, on the entire data, (train and test)

	Train data	Test data
RMSE	0.2776	0.2754

- The difference between errors for train and test sets is very insignificant
- We can conclude that the elastic-net regression model performs better at generalization compared to least squares model

Elastic-net regression penalty

- Elastic-net regression assigns a higher penalty to the model with higher variance

Penalty	Least squares regression model	Elastic-net regression model
$\lambda_{lasso}*(\lvert\beta_0\rvert+\lvert\beta_{Mileage}\rvert+\lvert\beta_{Engine\_Capacity}\rvert+\lvert\beta_{Age}\rvert)$ + $\lambda_{ridge}*((\beta_0)^2+(\beta_{Mileage})^2+(\beta_{Engine\_Capacity})^2+(\beta_{Age})^2)$	0.4190	0.0162

- Here $\lambda_{lasso} = 0.045$ and $\lambda_{ridge} = 0.05$, are calculated by trying different values

Comparison of models

Regression Model	β coefficients				Cost Function	RMSE	
	β_0	$\beta_{Mileage}$	$\beta_{Engine\_Capacity}$	β_{Age}		Train	Test
Least squares	0.2211	-0.1177	0.5694	-0.0187	$\Sigma(y_{act}-y_{pred})^2$	0.2315	0.3139
Elastic-net	0.3368	0	0.0112	0	$\Sigma(y_{act}-y_{pred})^2 + \lambda_{lasso}*\Sigma\|\beta\| + \lambda_{ridge}*\Sigma(\beta)^2$	0.2776	0.2754

Net regularization

The two stage elastic net regularization

$$\Sigma(y_{act}-y_{pred})^2 +\lambda_{ridge}*\Sigma(\beta) + \lambda_{lasso}*\Sigma|\beta|$$

- First Stage: Ridge regression
- Second Stage: LASSO regression
- Note a double amount of shrinkage resulting in increased bias.
- In turn makes poor predictions. To improve the prediction performance, rescale the coefficients of elastic net by multiplying the estimated coefficients by $(1+\lambda_{ridge})$

When to use which regularization?

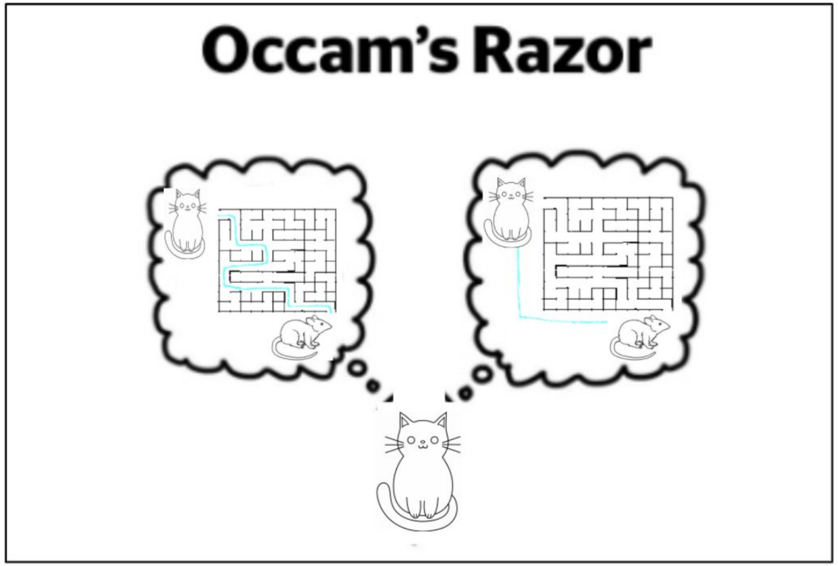
- If there are many interactions present or it is important to consider all the predictors in the model, ridge regression is used

- If the dataset contains some useless independent variables that can be eliminated from the model, lasso regression is used

- If the dataset contains too many variables where it is practically impossible to determine whether to use ridge or lasso regression, elastic-net regression is used

Occam’s razor

Regularization is an application of occam’s razor, since it helps in choosing a simple model rather than an overly complex one



“When faced with two equally good hypotheses, always choose the simpler”

Grid Search

Hyperparameter

- The estimates of parameters are usually estimated from the data
- However, some parameters do not learn from the model; they are preset by the user. Such parameters are called hyperparameters
- As seen in the gradient descent, the learning rate (α) is a hyperparameter. Also the parameter lambda (λ) in regularization is a hyperparameter

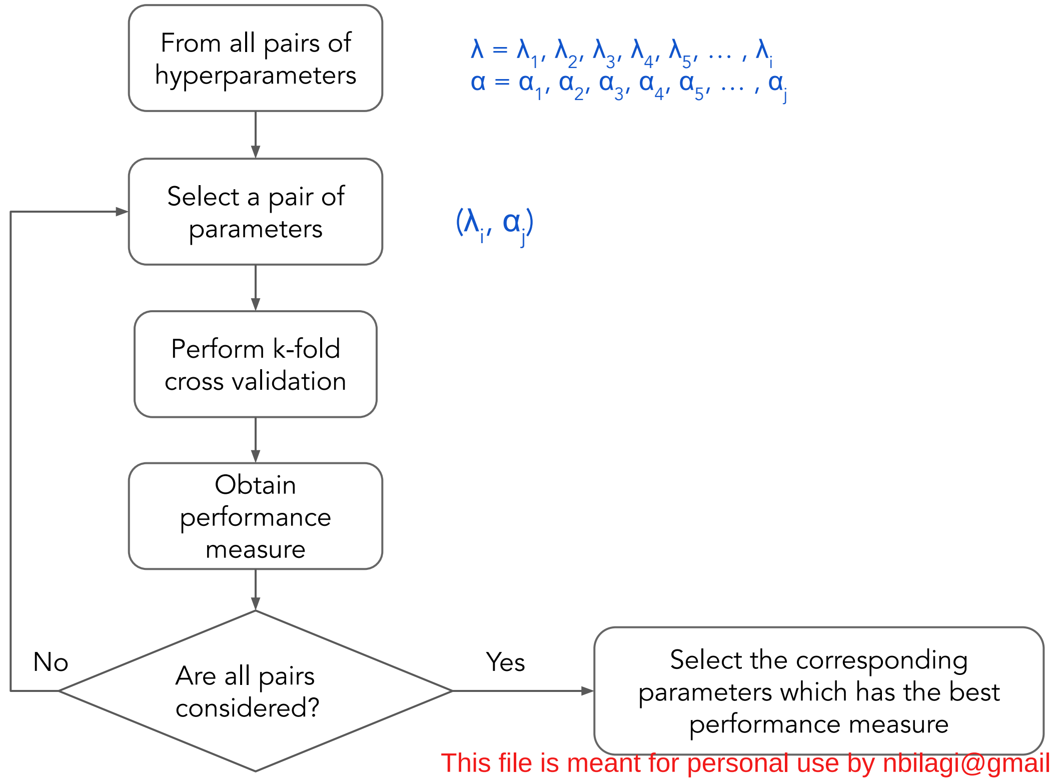
GridSearchCV

- The grid search is the process of tuning the hyperparameters to obtain the optimum values of the hyperparameters

- We use the the ‘GridSearchCV’ method to tune the hyperparameters

- Procedure:
    - Just as the name suggests, a grid of performance measure is obtained
    - Wherein each measure corresponds to a given hyperparameter values
    - Obtain the corresponding hyperparameters whose performance measure is highest

GridSearchCV



From all pairs of hyperparameters
Select a pair of parameters
Perform k-fold cross validation
Obtain performance measure
Are all pairs considered?
No
Yes
Select the corresponding parameters which has the best performance measure

$\lambda = \lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \dots, \lambda_i$
$\alpha = \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \dots, \alpha_j$

(λ_i, α_j)

Hyperparameter λ

Hyperparameter α

	λ_1	λ_2	λ_3	λ_4	λ_5
α_1	.76	.35	.67	.76	.66
α_1	.76	.87	.82	.64	.71
α_1	.45	.56	.85	.72	.79
α_1	.78	.67	.34	.83	.91
α_1	.56	.44	.35	.65	.87

The RMSE values

RandomizedSearch CV

RmizedSearch CV

The hyperparameters tuning using RmizedSearch CV

Procedure:

Step 1: Import the library
Step 2: Setting up the Data
Step 3: Model and its parameters
Step 4: Using Randomized Search CV and Print the results.

RmizedSearch CV

The hyperparameters tuning using RmizedSearch CV

Procedure:

Step 1: Import the library

    from sklearn.model_selection import RandomizedSearchCV

Step 2: Setting up the Data

    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25)

RmizedSearch CV

The hyperparameters tuning using RmizedSearch CV

Procedure:

Step 3: Model and its parameters

create a dictionary with hyperparameters and its values

'alpha' assigns the regularization strength to the model

```
tuned_paramaters = [{'alpha':[1e-15, 1e-10, 1e-8, 1e-4,1e-3, 1e-2,
0.1, 1, 5, 10, 20, 40, 60, 80, 100]}
```

initiate the ridge regression model

```
ridge = Ridge()
```

RmizedSearch CV

Step 4: Using Randomized Search CV and Print the results.

```
ridge_rand = RandomizedSearchCV(estimator = ridge,
                    param_distributions = tuned_paramaters,
                    cv = 10,n_iter = 10, n_jobs=-1)
```

- estimator : In this we have to pass the metric or the model for which we need to optimize the parameters.
- param_distributions : In this we have to pass the dictionary of parameters that we need to optimize.
- cv : In this we have to pass a interger value, as it signifies the number of splits that is needed for cross validation. By default is set as five.
- n_iter : This signifies the number of parameter settings that are sampled. By default it is set as 10.

GridSearchCV Vs RmizedSearchCV

RandomizedSearchCV performs a randomized search over a specified distribution of hyperparameters. Instead of evaluating all possible combinations of hyperparameters like GridSearchCV.

RandomizedSearchCV randomly selects a combination of hyperparameters from the specified distribution for each iteration. The number of iterations is typically specified by the user. The model is trained and evaluated for each combination of hyperparameters, and the best combination is selected based on a performance metric.

Summary

- GridSearchCV is a more exhaustive method but can be computationally expensive.

- RandomizedSearchCV is a more efficient method that may miss some combinations of hyperparameters but is less computationally expensive.

- The choice between the two methods ultimately depends on the specific problem and available computational resources.