Exploratory Project

CSM 291

on

Linear Regression - Predicting Insurance Prices

Submitted By

Mr. Shrey Gupta

B.Tech. Student of Mathematics & Computing

IIT(BHU), Varanasi

1 Introduction

The most interesting topic in the modern era is Machine Learning. Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience. There are many divisions of Machine Learning. One of the most important topic in Machine Learning is *Linear Regression*. Linear Regression can be used to solve a variety of real-world problems like predicting the housing prices, predicting the sales for a particular year of a company among many others. Also, a supervised learning algorithm is an algorithm in which we give the algorithm some training data to learn from and then after training make the predictions. Linear regression is also a supervised learning algorithm in which we have to fit a function f that maps our inputs X to the corresponding function values $f(x)$. One such problem is as follows:
Suppose we are given the data of an insurance company which contains the charges they charge for their insurance for a person using the provided information by the person.

age	sex	bmi	children	smoker	region	charges
19	female	27.9	0	yes	southwest	16884.924
18	male	33.77	1	no	southeast	1725.5523
28	male	33	3	no	southeast	4449.462
33	male	22.705	0	no	northwest	21984.47061
32	male	28.88	0	no	northwest	3866.8552
31	female	25.74	0	no	southeast	3756.6216
46	female	33.44	1	no	southeast	8240.5896
37	female	27.74	3	no	northwest	7281.5056
37	male	29.83	2	no	northeast	6406.4107
60	female	25.84	0	no	northwest	28923.13692
25	male	26.22	0	no	northeast	2721.3208
62	female	26.29	0	yes	southeast	27808.7251
23	male	34.4	0	no	southwest	1826.843

Now, we wish to predict the insurance prices for a new person given his/her data. We can do the above task by:

(i) Data Analysis

(ii) Cleaning the data and preparing the dataset for training

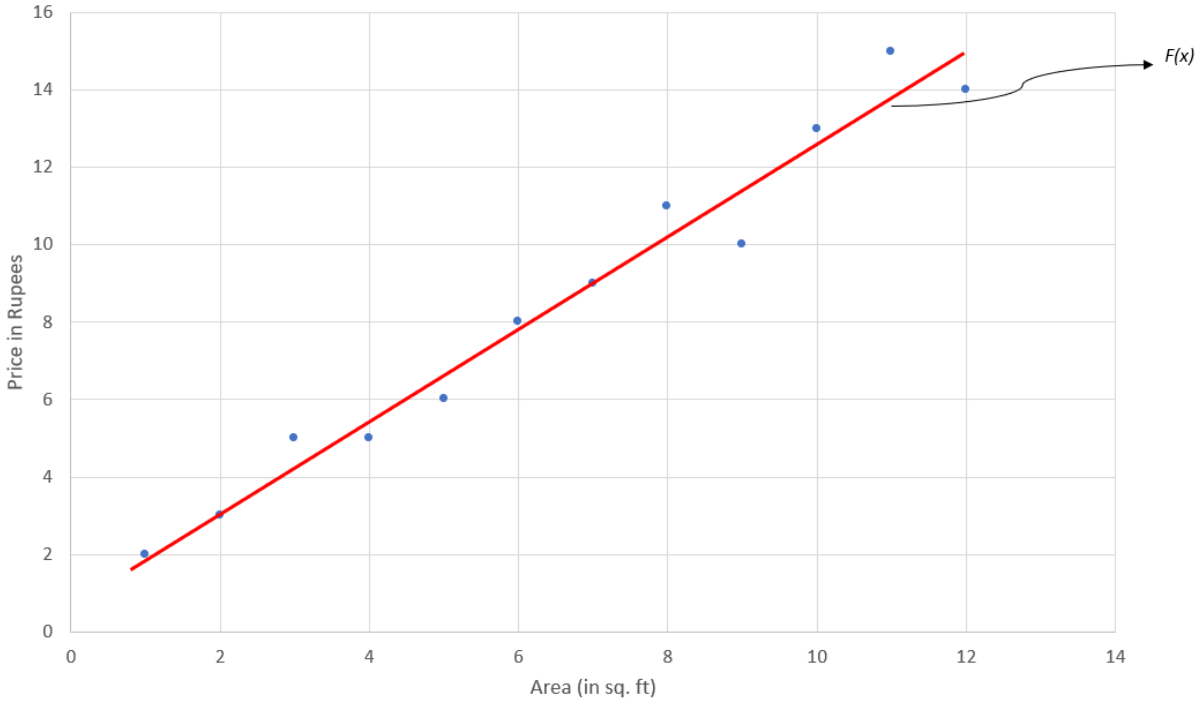
(iii) Make a model that best fits the data: Maximum Likelihood Estimation(MLE), Maximum a Posterior Estimation(MAP) and Bayesian Linear Regression(BLR).

(iv) Make Predictions

In this project, we create a suitable model that will fit the training data well and make descent predictions based on the understandings from the literature survey.

2 Literature Survey

Regression is a supervised learning algorithm. In a regression problem, we have to fit a function f that maps our inputs X to the corresponding function values $f(x)$. For example, suppose we are trying to predict the housing prices on the basis of the area (in sq. ft.) and we are given some training data for the same. We plot the graph for these training examples :-



We can clearly see that as the $area$ increases the $price$ also increases linearly. So, we can fit

$$f(x) = \theta_0 + \theta_1 x$$

We have to determine these model parameters (θ_0, θ_1) that fit this graph very well. Now, using this $f(x)$ we can predict house price given its area.

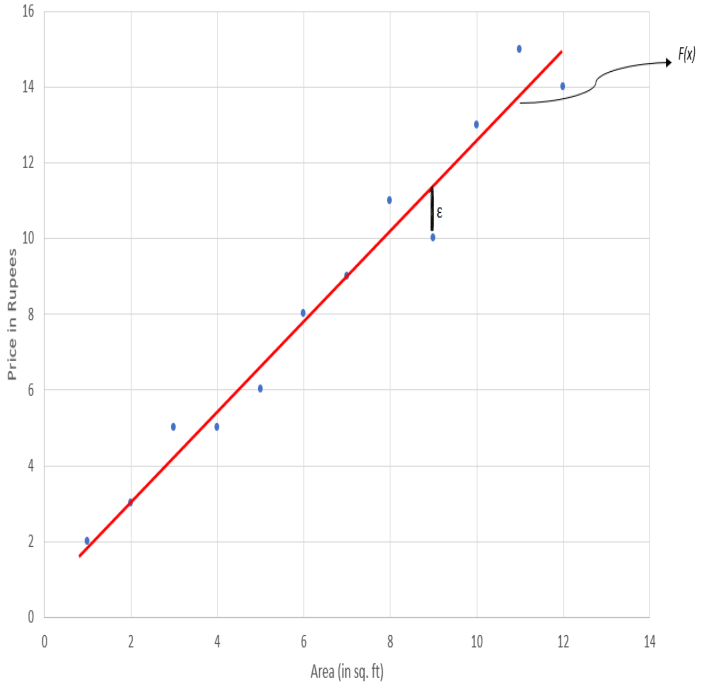
The word"linear" in linear regression means that the models that are "linear in parameters", that is, models that describe a function by a linear combination of input features, X.

In the above example, let us assume that we are given set of training inputs, x_n and corresponding noisy observations

$$y_n = f(x_n) + \epsilon,$$

where ϵ is the measurement noise. We assume ϵ is an independent and identically distributed random variable. It has a Normal (Gaussian) distribution with mean 0 and variance σ^2.

$$\epsilon \sim \mathcal{N}(0, \sigma^2)$$



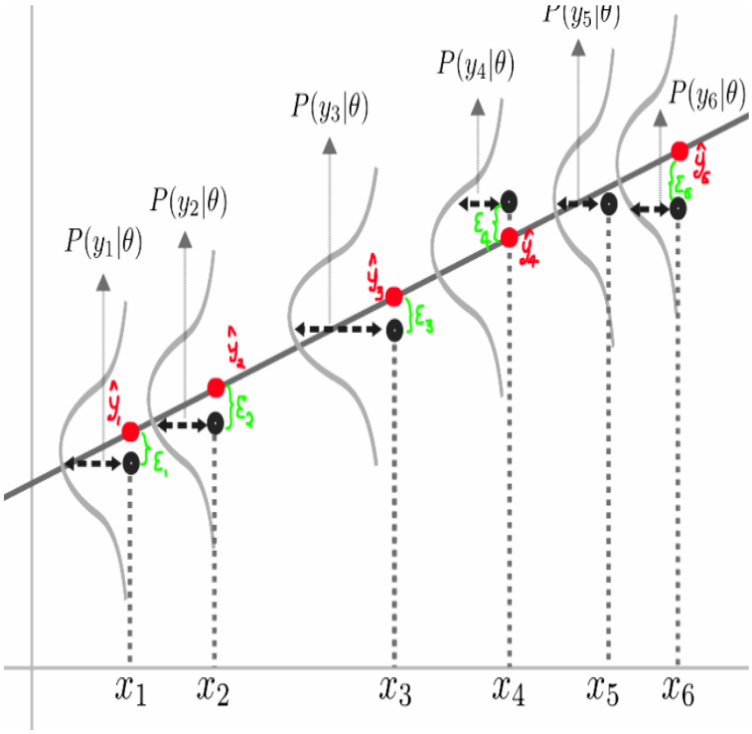
So, finally our task in linear regression is to find a function that not only models the training data, but generalizes well to predicting function values at input locations that are not a part of the training set.

2.1 Problem Formulation

Let us consider the likelihood function

$$p(y|x) = \mathcal{N}(y|f(x), \sigma^2)$$

where $x \in \mathbb{R}^D$ are the inputs and $y \in \mathbb{R}$ are the targets. Here, D refers to the number of features. Also, $y = f(x) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is an independent and identically distibuted random variable that represents the noise. Suppose the graph for the training examples is as follows:



For y_n the mean of the distribution is $f(x_n)$ and variance is σ^2. Now, for linear regression, we consider

$$f(x) = x^T\theta$$

where x is the feature vector and θ is the model parameters vector. For example, consider that we are having 4 features, then

$$x = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ x^{(3)} \\ x^{(4)} \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix}$$

$$f(x) = x^T\theta$$

$$f(x) = \begin{bmatrix} x^{(1)} & x^{(2)} & x^{(3)} & x^{(4)} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix}$$

$$f(x) = x^{(1)}\theta_1 + x^{(2)}\theta_2 + x^{(3)}\theta_3 + x^{(4)}\theta_4$$

For the time being we assume σ^2 to be known. Hence, $p(y|x) = \mathcal{N}(y|x^T\theta, \sigma^2) \Leftrightarrow y = x^T\theta + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2)$, where $\theta \in \mathbb{R}^D$ are the model parameters which we need to determine. Here, the

likelihood is the probability density function of y evaluated at $x^T\theta$.

2.2 Parameter Estimitation

Consider the training set,

$$Z = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

consisting of N inputs $x_n \in \mathbb{R}^D$ and targets $y_n \in \mathbb{R}$, $n = 1, 2, \ldots, N$. Now,

$$p(Y|X, \theta) = p(y_1, y_2, \ldots, y_n|x_1, x_2, \ldots, x_n, \theta).$$

As we know that y_i and y_j are conditionally independent given their respective inputs x_i and x_j. Therefore,

$$p(Y|X, \theta) = \prod_{n=1}^{N} p(y_n|x_n, \theta) = \prod_{n=1}^{N} \mathcal{N}(y_n|x_n, \theta)$$

where $X = \{x_1, x_2, \ldots, x_n\}$ is the input set and $y = \{y_1, y_2, \ldots, y_n\}$ is the target set. Now, we have to find optimal parameters $\theta^* \in \mathbb{R}^D$, so that they fit the data very well. Once we are able to find θ^* then we can predict function values using $y = x^T\theta + \epsilon$, so that at an arbitrary test input x_* the distribution of the corresponding target y_* is

$$p(y_*|x_{*,\theta^*}) = \mathcal{N}(y_*|x_*^T\theta^*, \sigma^2).$$

2.3 Maximum Likelihood Estimation

Maximizing the likelihood means maximizing the predictive distribution of the training data given the model parameters. Mathematically, we have to evaluate

$$\theta_{ML} = \arg\max_{\theta} P(Y|X, \theta).$$

We can find θ_{ML} by using an iterative approach called gradient descend on the negative likelihood. But there is a closed form solution for this, so gradient descend becomes unnecessary. In practice, instead of maximizing the likelihood directly, we apply the log-transformation to the likelihood function and then minimize the negative log-likelihood.

Advantages of the log Likelihood:

1. It does not suffer from numerical underflow. As likelihood is the product of N Gaussian distributions. Hence, multiplying N probabilities can lead to very small number which can cause underflow.

2. Differentiation becomes easier. Applying repeated product rule is very cumbersome. Rather in log-transformation we have sum of individual derivatives.

Now,

$$-\log p(Y|X,\theta) = -\log \prod_{n=1}^{N} p(y_n|x_n,\theta) = -\sum_{n=1}^{N} \log p(y_n|x_n,\theta)$$

We have to minimize this negative log-likelihood.

$$p(y_n|x_n,\theta) = \mathcal{N}(y_n|x_n^T\theta,\sigma^2) = \frac{e^{\frac{-(y_n-x_n^T\theta)^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma}$$

This implies

$$\log p(y_n|x_n,\theta) = \log(\frac{1}{\sqrt{2\pi}\sigma}) - \frac{(y_n - x_n^T\theta)^2}{2\sigma^2}$$

Ignoring the $\log(\frac{1}{\sqrt{2\pi}\sigma})$ (independent term of θ), we have

$$-\log p(y_n|x_n,\theta) = \frac{(y_n - x_n^T\theta)^2}{2\sigma^2}$$

Now,

$$-\log p(Y|X,\theta) = \sum_{n=1}^{N} \frac{(y_n - x_n^T\theta)^2}{2\sigma^2}. \tag{2.1}$$

The negative log likelihood is also called *Loss function*.

$$\mathcal{L}(\theta) = \frac{\sum_{n=1}^{N}(y_n - x_n^T\theta)^2}{2\sigma^2} = \frac{(y - X\theta)^T(y - X\theta)}{2\sigma^2}$$

where,

$$X = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(D)} \\ x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(D)} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ x_N^{(1)} & x_N^{(2)} & \dots & x_N^{(D)} \end{bmatrix} \in \mathbb{R}^{NXD}$$

is the feature matrix consisting of N training inputs and D features.

$$y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_n \end{bmatrix} \in \mathbb{R}^N \text{ and } \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ . \\ . \\ . \\ \theta_D \end{bmatrix} \in \mathbb{R}^D.$$

We see that the loss function is the sum of squared errors between y_n and model prediction $x_n^T\theta$.

$$\mathcal{L} = \frac{(y - X\theta)^T(y - X\theta)}{2\sigma^2} = \frac{\|y - X\theta\|^2}{2\sigma^2}$$

where $\|y - X\theta\| = (y - X\theta)^T(y - X\theta)$ and is called 2-norm of the matrix. Now, we need to find the global optimum of this function.

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \theta} &= \frac{\partial}{\partial \theta}(\frac{(y - X\theta)^T(y - X\theta)}{2\sigma^2}) \\ &= \frac{1}{2\sigma^2}\frac{\partial}{\partial \theta}((y^T - \theta^T X^T)(y - X\theta)) \\ &= \frac{1}{2\sigma^2}\frac{\partial}{\partial \theta}(y^T y - y^T X\theta - \theta^T X^T y + \theta^T X^T X\theta) \end{aligned}$$

We see that $y^T X\theta$ is a $1X1$ matrix and hence symmetric. Therefore, $y^T X\theta = (y^T X\theta)^T = \theta^T X^T y$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \theta} &= \frac{1}{2\sigma^2}\frac{\partial}{\partial \theta}(y^T y - 2y^T X\theta + \theta^T X^T X\theta) \\ &= \frac{1}{2\sigma^2}(0 - 2y^T X + 2\theta^T X^T X) \\ &= \frac{1}{\sigma^2}(\theta^T X^T X - y^T X) \end{aligned} \tag{2.2}$$

Setting gradient to 0^T is a necessary and sufficient condition, and we will obtain the global optimum as Hessian $\nabla^2_\theta \mathcal{L}(\theta) = X^T X \in \mathbb{R}^{DXD}$.(Till $X^T X$ is positive and definite). Therefore,

$$\therefore \frac{\partial \mathcal{L}}{\partial \theta} = 0^T$$

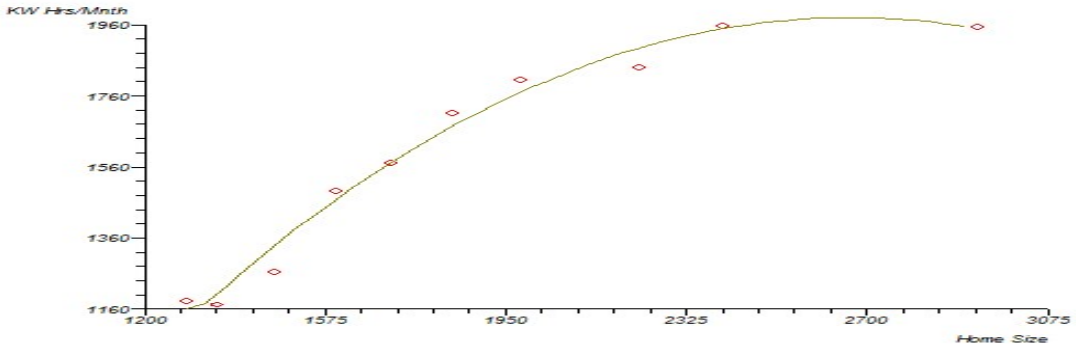
This implies

$$\begin{aligned} \theta_{ML}^T X^T X &= y^T X \\ \theta_{ML}^T &= y^T X(X^T X)^{-1} \\ \theta_{ML} &= (X^T X)^{-1} X^T y. \end{aligned}$$

For all this to happen rank(X) has to be equal to D. Because if rank(X)=D.
X^TX will be positive definite and invertible. Hence, Hessian will be positive and definite and will have global optimum.

2.4 Maximum Likelihood with Features

Assume that we are given a dataset with the following graph



We can clearly see that a straight line will not fit this data. Instead we need a polynomial fit for this data. Since linear regression only refers to linear in parameters, we can perform an arbitrary non-linear transformation $\phi(x)$ of the inputs x and then linearly combine the components of this transformation. Therefore,

$$p(y|x,\theta) = \mathcal{N}(y|\phi^T(x)\theta, \sigma^2) \Leftrightarrow y = \phi^T(x)\theta + \epsilon = \sum_{k=0}^{K-1} \theta_k \phi_k(x) + \epsilon$$

where $\phi : \mathbb{R}^D \to \mathbb{R}^K$ is a non linear transformation of inputs x. $\phi_k : \mathbb{R}^D \to \mathbb{R}$ is the kth component of the vector ϕ. For example,

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ . \\ . \\ . \\ x^{K-1} \end{bmatrix} = \begin{bmatrix} \phi_0(x) \\ \phi_1(x) \\ \phi_2(x) \\ . \\ . \\ . \\ \phi_{K-1}(x) \end{bmatrix} \in \mathbb{R}^K$$

So in simple words we have increased the number of features from 1 to K. Now, generalizing the things. The design(feature) matrix will be:

$$\Phi = \begin{bmatrix} \phi^T(x_1) \\ \phi^T(x_2) \\ . \\ . \\ . \\ \phi^T(x_N) \end{bmatrix} = \begin{bmatrix} \phi_0(x_1) & \phi_1(x_1) & . & . & . & \phi_{K-1}(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & . & . & . & \phi_{K-1}(x_2) \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \phi_0(x_N) & \phi_1(x_N) & . & . & . & \phi_{K-1}(x_N) \end{bmatrix}$$

where $x_n \in \mathbb{R}^D$ are the training inputs and $y_n \in \mathbb{R}$ are the targets, $n = 1, 2, 3, \ldots, N$. The model parameters

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ . \\ . \\ . \\ \theta_{K-1} \end{bmatrix} \in \mathbb{R}^K.$$

For example. For third order polynomials. Assume $x_n \in \mathbb{R}, n = 1, 2, \ldots, N$ be the inputs and $y_n \in \mathbb{R}$ be the targets. Then,

$$\Phi = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ 1 & x_N & x_N^2 & x_N^3 \end{bmatrix} \in \mathbb{R}^{NX4} \qquad \text{and} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} \in \mathbb{R}^4$$

Now combining it with our normal likelihood we see that we have only changed X with Φ. Hence,

$$\theta_{ML} = (\Phi^T\Phi)^{-1}\Phi^T y.$$

Here, $\Phi^T\Phi$ is invertible if rank(Φ) is K.

2.5 Estimating Noise Variance

Till now we have assumed that σ^2 was known to us. But we can calculate σ^2_{ML} for noise variance with the help of likelihood function. Now

$$\begin{aligned}\log p(Y|X,\theta,\sigma^2) &= \sum_{n=1}^{N} \log \mathcal{N}(y_n|\phi^T(x_n)\theta,\sigma^2) \\ &= \sum_{n=1}^{N} (\frac{-\log(2\pi)}{2} - \frac{\log\sigma^2}{2} - \frac{(y_n-\phi^T(x)\theta)^2}{2\sigma^2}) \\ &= -\frac{N}{2}\log\sigma^2 - \frac{\sum_{n=1}^{N}(y_n-\phi^T(x_n)\theta)^2}{2\sigma^2} + const\end{aligned}$$

Assuming $\sigma^2 = z$ and $\sum_{n=1}^{N}(y_n-\phi^T(x_n)\theta)^2 = S$

$$\log p(Y|X,\theta,\sigma^2) = -\frac{N}{2}\log z - \frac{S}{2z} + \text{constant.}$$

Now partial derivative $w.r.t.$ z.

$$\frac{\partial}{\partial z}\log p(Y|X,\theta,\sigma^2) = -\frac{N}{2z} + \frac{S}{2z^2}$$

Now putting this derivative to 0.

$$-\frac{N}{2z} + \frac{S}{2z^2} = 0 \implies z = \frac{S}{N}$$

Therefore,

$$\sigma^2_{ML} = \frac{\sum_{n=1}^{N}(y_n-\phi^T(x_n)\theta)^2}{N}.$$

2.6 Overfitting in Linear Regression

We can see that even after fitting the best possible curve for the dataset we still have some error in the model. We can evaluate the quality of the model by:

1. The Negative Log-Likelihood

$$\mathcal{L}(\theta_{ML}) = \frac{(y-X\theta_{ML})^T(y-X\theta_{ML})}{2\sigma^2}$$

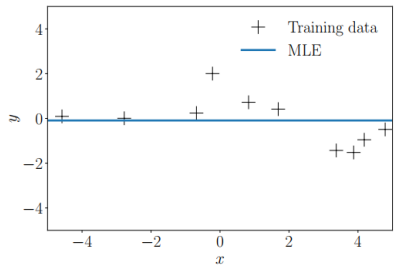
   Less the loss better the model.

2. Using RMSE

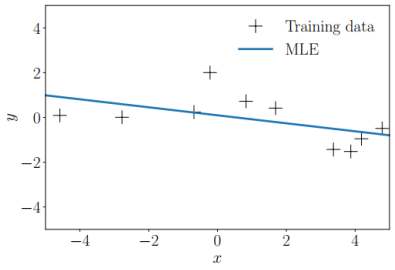
$$RMSE = \sqrt{\frac{\sum_{n=1}^{N}(y_n - \phi^T(x_n)\theta)^2}{N}}$$

less the loss better the model.

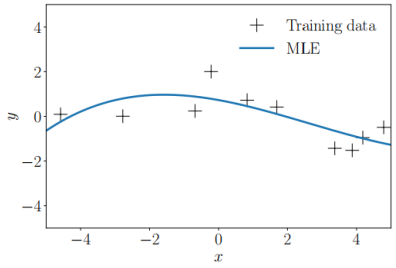
So by these two ways we can evaluate the quality of our model. Let us assume that we have determined the best degree of the polynomial by finding the polynomial degree M that minimizes the loss function or RMSE. If we assume that the polynomial degree is a natural number, we can perform brute-force search and enumerate all the reasonable values of M.
For a training set of size N it is sufficient to test $0 \leq M \leq N-1$. Because for $M \geq N$ we will have more parameters(θ) than the data points, that will lead to $\Phi^T\Phi$ being non-invertible, so that there will be infinite many possible maximum likelihood parameters(θ_{ML}). Suppose that we plot the graphs with different values of M with $N = 10$ training examples.



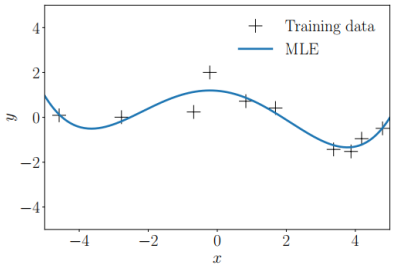
(a) $M = 0$



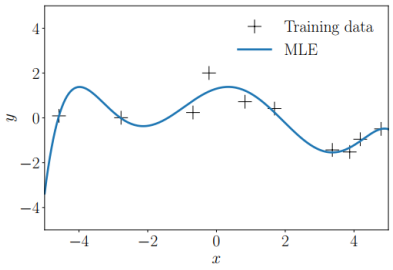
(b) $M = 1$



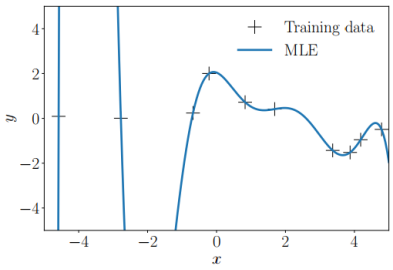
(c) $M = 3$



(d) $M = 4$



(e) $M = 6$



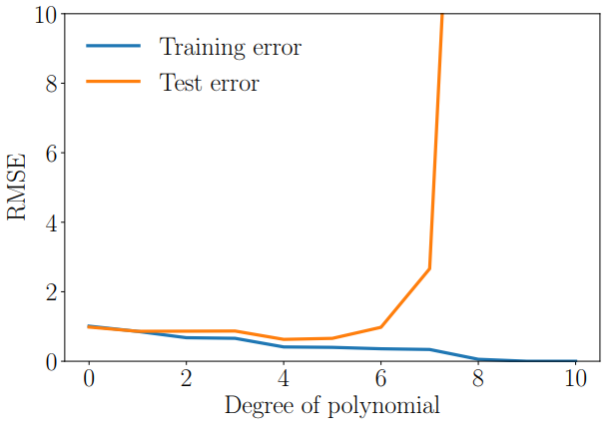
(f) $M = 9$

We clearly see that for low degree polynomials($M = 0, 1$) the curve fit the data poorly. This is called underfitting. For $M = 3 - 5$, the curve was reasonably fitting the data good.
Bur for $M = 9$, we see that the curve fit the data points very correctly without leaving any point. This is known as overfitting. It we analyse the graph for $M = 9$, we will see that if we will provide a new test data point to our model then it will predict rubbish values as the graph is sometimes overshooting and is not that much stable. But as we know that our prime objective of linear regression is to predict correct values for unseen data points. But our model for high degree polynomials though fit the training data into account but does not generalizes the result. So, to check our model performance, we have to take into account both the training error and test error. We can split our data set into two parts

1. Training Set - For training the model

2. Test/Dev Set - For testing our model

For example - For the above graphs if we plot a graph between *Degree of polynomial Vs RMSE*.

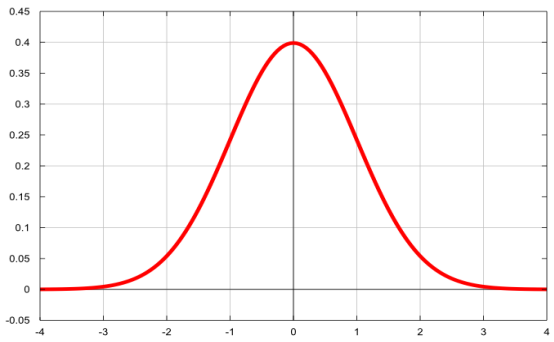


So we see that $M = 4$ generalizes test data well as well as fits the training data well.

2.7 Maximum A Posteriori Estimation

We know that MLE(Maximum Likelihood Estimation) is prone to overfitting.
Also it is observed that when we run into overfitting then our parameters(θ) becomes relatively large. So we can reduce overfitting by making our parameters values not too large. We can achieve this by placing a prior distribution $p(\theta)$ on the parameters.
Example Consider a Gaussian prior $p(\theta) = \mathcal{N}(0, 1)$. Therefore,

$$ep(\theta) = \frac{e^{\frac{-\theta^2}{2}}}{\sqrt{2\pi}}.$$



We see that here probability of θ is maxed around 0, and θ is expected to lie in $[-2, 2]$ (two standard deviations around mean). Given our training data X, Y. Here we will maximize the posterior distribution $p(\theta|X, Y)$. This method is called maximum a posterior(MAP) estimation. Now, applying Baye's

theorem, we get

$$p(\theta|X,Y) = \frac{p(Y|X,\theta)p(\theta)}{p(Y|X)} \tag{2.3}$$

Now we have to find θ_{MAP} that maximizes the $p(\theta|X,Y)$. Applying log to equation (2.3)

$$\log p(\theta|X,Y) = \log p(Y|X,\theta) + \log p(\theta) - \log p(Y|X)$$

Here, $p(Y|X)$ is independent of θ. Hence,

$$\log p(\theta|X,Y) = \log p(Y|X,\theta) + \log p(\theta) + const \tag{2.4}$$

We see that *log posterior* is the sum of *log- likelihood* $p(Y|X,\theta)$ and *log-prior*. Hence MAP estimate will be a compromise between prior and likelihood. Now, $\theta_{MAP} \in \arg\max_{\theta}(-\log p(Y|X,\theta) - \log p(\theta))$. We assume that $p(\theta) = \mathcal{N}(0, b^2 I)$. Here, I is the identity matrix, which is used for maintaining correct dimensions of the equations. Using equations (2.4) and (2.1), we have

$$-\log p(\theta|X,Y) = \frac{(y-\Phi\theta)^T(y-\Phi\theta)}{2\sigma^2} + \frac{\theta^T\theta}{2b^2} + const$$

Using equation (2.2)

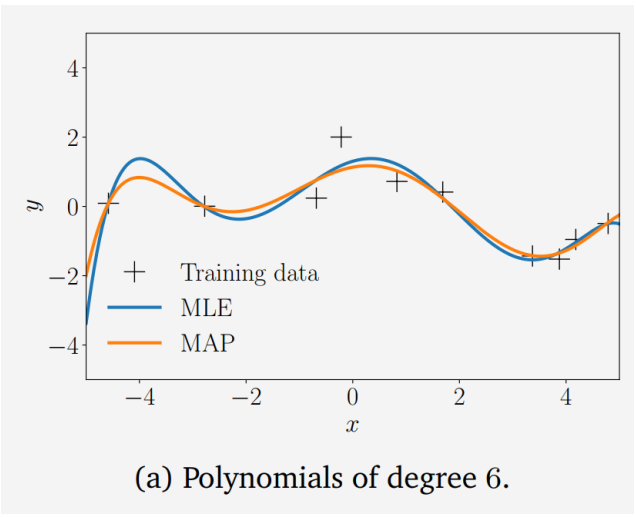
$$\frac{-\partial \log p(\theta|X,Y)}{\partial\theta} = \frac{(\theta^T\Phi^T\Phi - y^T\Phi)}{\sigma^2} + \frac{\theta^T}{b^2}$$

Now putting it to 0^T.

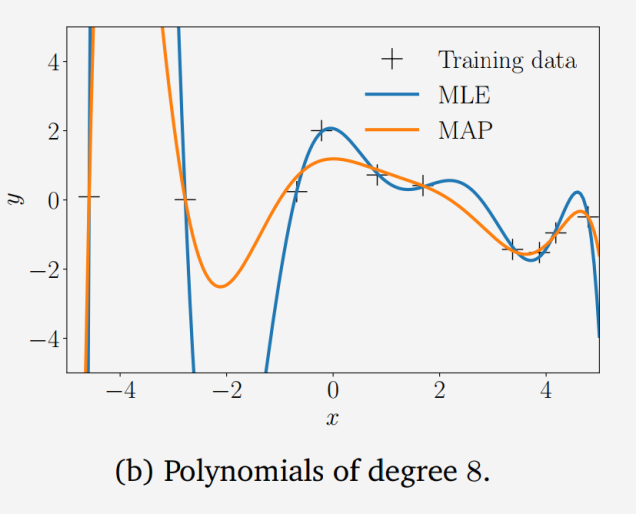
$$\begin{aligned}
& \frac{(\theta^T\Phi^T\Phi - y^T\Phi)}{\sigma^2} + \frac{\theta^T}{b^2} = 0^T \\
\implies \quad & \theta^T(\frac{\Phi^T\Phi}{\sigma^2} + \frac{I}{b^2}) - \frac{y^T\Phi}{\sigma^2} = 0^T \\
\implies \quad & \theta^T(\Phi^T\Phi + \frac{\sigma^2}{b^2}I) = y^T\Phi \\
\implies \quad & \theta^T = y^T\Phi(\Phi^T\Phi + \frac{\sigma^2}{b^2}I)^{-1} \\
\implies \quad & \theta_{MAP} = (\Phi^T\Phi + \frac{\sigma^2}{b^2}I)^{-1}\Phi^T y
\end{aligned}$$

So comparing this with MLE we see that the only change is the addition of $\frac{\sigma^2}{b^2}I$ term in the inverse matrix. This term $\frac{\sigma^2}{b^2}I$ ensures that $\Phi^T\Phi + \frac{\sigma^2}{b^2}I$ is symmetric and strictly positive definite. Hence, its inverse always exist. Also, it represents the impact of regularization which prevents θ from taking too large values.

Example. Consider the graphs which we analyzed in section 2.6 Overfitting in MLE. But now let us define a prior $p(\theta) = \mathcal{N}(0, I)$.



(a) Polynomials of degree 6.



(b) Polynomials of degree 8.

We see that for lower degree of polynomial, MAP estimate does not played a significant role. We see for high degree polynomials MAP kept the function relatively smoother and prevented overfitting to some extent.

2.8 MAP Estimation as Regularization

We can prevent overfitting by a method called regularization. In this method, our loss function is

$$\mathcal{L}(\theta) = \|y - \Phi\theta\|^2 + \lambda\|\theta\|_2^2 \qquad (\|\theta\|_2^2 = \sum \theta_i^2)$$

where $\|.\|$ is the euclidean norm and λ is regularization constant. $\lambda \in [0, \infty)$. This parameter decides to what extent we have to do regularization. More λ more regularization. As we know that we want to minimize $\mathcal{L}(\theta)$ hence this $\lambda\|\theta\|_2^2$ will prevent $\theta_i's$ to be large. Hence preventing overfitting. Now comparing $\lambda\|\theta\|_2^2$ with negative log prior = $\frac{\|\theta\|_2^2}{2b^2} + const.$ We can say that $\lambda = \frac{1}{2b^2}$. We see that here $\mathcal{L}(\theta)$ is very much identical to the MAP estimate. Hence,

$$\theta_{RLS} = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T y$$

For $\lambda = \frac{\sigma^2}{b^2}$, this is identical to MAP estimate.

2.9 Bayesian Linear Regression

Bayesian linear regression does not attempt to compute a point estimate of the parameters, but instead the full posterior distribution over parameters(θ) is taken into account when making predictions. This means we compute the mean over all plausible parameter settings.
For Bayesian linear regression, consider,
prior $p(\theta) = \mathcal{N}(m_o, S_o)$
likelihood $p(y|x, \theta) = \mathcal{N}(y|\phi^T(x)\theta.\sigma^2)$,

which is somewhat very similar to the MAP estimate except the prior.

2.9.1 Prior Predictions

In this method, we take the parameters(θ) and average over all the plausible parameter settings when we make predictions.

$$\begin{aligned} p(y_*|x_*) &= \int p(y_*|x_*,\theta)p(\theta)d\theta \\ &= E_\theta[p(y_*|x_*,\theta)] \end{aligned}$$

We see that this is an average prediction of $p(y_*|x_*,\theta)$ for all the plausible values of θ. Also, note that while making predictions using prior distribution, it requires only the input x_* and no training data. Hence, according to to our model

$$\begin{aligned} p(\theta) &= \mathcal{N}(m_o, S_o) \\ p(y_*|x_*,\theta) &= \mathcal{N}(y_*|\phi^T(x_*)\theta, \sigma^2) \end{aligned}$$

Hence on solving this we get,

$$p(y_*|x_*) = \mathcal{N}(\phi^T(x_*)m_o, \phi^T(x_*)S_o\phi(x_*) + \sigma^2)$$

Here on analysing $\phi^T(x_*)S_o\phi(x_*)$ is the uncertainity associated with parameters θ and σ^2 is the uncertainity contribution due to measurement noise. We see that we have find out the distribution $p(y_*|x_*)$. But if we want to find out the distribution of $f(x_*) = \phi^T(x)\theta$. We see that $f(x_*)$ will be noise-free. Hence,we will see that only the σ^2 will be ommited in the expression. We obtain

$$p(y_*|x_*) = \mathcal{N}(\phi^T(x_*)m_o, \phi^T(x_*)S_o\phi(x_*) + \sigma^2)$$

Now,

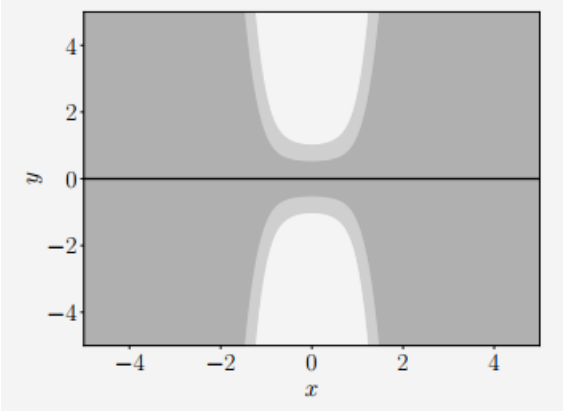
$$E[f(x_*)] = E[\phi^T(x_*)\theta] = \phi^T(x_*)E[\theta] = E[f(x_*)] = \phi^T(x_*)m_o$$

$$Var[f(x_*)] = Var[\phi^T(x_*)\theta] = \phi^T(x_*)Var[\theta]\phi(x_*) = \phi^T(x_*)S_o\phi(x_*).$$

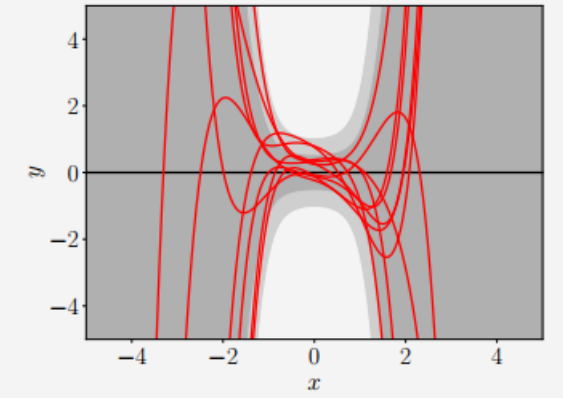
Hence,

$$f(x_*) = \mathcal{N}(\phi^T(x_*)m_o, \phi^T(x_*)S_o\phi(x_*))$$

We see that the mean function will be $\phi^T(x)m_o$.
Example If we chose parameter prior to be $p(\theta) = \mathcal{N}(0, \frac{I}{4})$. We see that $m_o = 0$. Hence, the mean function is also 0.



Shaded area is the distribution of the function and the black line is the mean function.



(b) Samples from the prior distribution over functions.

2.9.2 Posterior Predictions

Similar to prior predictions, we can also make posterior predations. Before making predictions using posterior distribution, we will first derive the posterior distribution. Given training inputs $x_n \in \mathbb{R}^D$ outputs $y_n \in \mathbb{R}$ $n = 1, 2, 3, \ldots, N$. Posterior over parameters(θ)

$$p(\theta|X, Y) = \frac{p(Y|X, \theta)p(\theta)}{p(Y|X)}.$$

Here, X is the training set, Y is corresponding training targets, $p(Y|X, \theta)$ is the likelihood and $p(\theta)$ is the prior.

$$p(Y|X) = \int p(Y|X, \theta)p(\theta)d\theta = E_\theta[p(Y|X, \theta)] \tag{2.5}$$

is the marginal likelihood which is independent of θ and ensures that posterior is normalized($i.e.$ it integrates to 1).

$$p(Y|X,\theta) = \mathcal{N}(y|\phi(x)\theta, \sigma^2 I)$$

This implies

$$p(\theta) = \mathcal{N}(\theta|m_o, S_o)$$

As we know that $p(Y|X)$ is independent of θ we can ignore it. Our main aim is to find the posterior distribution

$$p(\theta|X,Y) = \mathcal{N}(\theta|m_N, S_N)$$

So we have to find these m_N and S_N. The sum of log-prior and log-likelihood.

$$\log \mathcal{N}(y|\Phi\theta, \sigma^2 I) + \log \mathcal{N}(\theta|m_o, S_o) = \frac{-1}{2}(\frac{(y-\Phi\theta)^T(y-\Phi\theta)}{\sigma^2} + (\theta - m_o)^T S_o^{-1}(\theta - m_o)) + const$$

Here, constant represents the terms independent of θ.

$$\begin{aligned}\log \mathcal{N}(y|\Phi\theta, \sigma^2 I) + \log \mathcal{N}(\theta|m_o, S_o) &= \frac{-1}{2}\big(\frac{y^T y}{\sigma^2} - \frac{2y^T\Phi\theta}{\sigma^2} + \frac{\theta^T\Phi^T\Phi\theta}{\sigma^2} \\ &\qquad + \theta^T S_o^{-1}\theta - 2m_o^T S_o^{-1}\theta + m_o^T S_o^{-1} m_o\big) \\ &= \frac{-1}{2}(\theta^T(\sigma^{-2}\Phi^T\Phi + S_o^{-1})\theta - 2(\sigma^{-2}\Phi^T y + S_o^{-1}m_o)^T\theta)\end{aligned}$$

Now log-posterior

$$\begin{aligned}\log \mathcal{N}(\theta|m_N, S_N) &= \frac{-(\theta - m_N)^T S_N^{-1}(\theta - m_N)}{2} + const \\ &= \frac{-(\theta^T S_N^{-1}\theta - 2m_N^T S_N^{-1}\theta + m_N^T S_N^{-1} m_N)}{2} + const \\ &= \frac{-(\theta^T S_N^{-1}\theta - 2m_N^T S_N^{-1}\theta)}{2} + \text{constant.}\end{aligned}$$

Now comparing linear and quadratic terms the above equations.

$$\begin{aligned}S_N^{-1} &= \Phi^T\sigma^{-2} I\Phi + S_o^{-1} \\ S_N &= (\sigma^{-2}\Phi^T\Phi + S_o^{-1})^{-1}\end{aligned}$$

and

$$\begin{aligned}m_N^T S_N^{-1} &= (\sigma^{-2}\Phi^T y + S_o^{-1}m_o)^T \\ m_N &= S_N(\sigma^{-2}\Phi^T y + S_o^{-1}m_o)\end{aligned}$$

Now that we have calculated the posterior distribution.

$$p(\theta|X,Y) = \mathcal{N}(\theta|m_N, S_N).$$

Hence, we can now make predictions using posterior distribution. This is very similar to the prior predictions.

$$\begin{aligned} p(y_*|X,Y,x_*) &= \int p(y_*|x_*,\theta)p(\theta|X,Y)d\theta \\ &= \int \mathcal{N}(y_*|\phi^T(x_*)\theta,\sigma^2)\mathcal{N}(\theta|m_N,S_N)d\theta \\ &= \mathcal{N}(y_*|\phi^T(x_*)m_N, \phi^T(x_*)S_N\phi(x_*)+\sigma^2) \end{aligned}$$

Here we see that the predictive mean is $\phi^T(x_*)m_N$. We can see that

$$p(y_*|X,Y,x_*) = E_{\theta|X,Y}[p(y_*|x_*,\theta)]$$

where it is the expectation of $p(y_*|x_*,\theta)$ with respect to parameter posterior $p(\theta|X,Y)$. We see that we have find out the distribution $p(y_*|X,Y,x_*)$. But if we want to find out the distribution of $f(x_*) = \phi^T(x)\theta$. We see that $f(x_*)$ will be noise-free. Hence,we will see that only the σ^2 will be committed in the expression. We obtain

$$p(y_*|X,Y,x_*) = \mathcal{N}(y_*|\phi^T(x_*)m_N, \phi^T(x_*)S_N\phi(x_*)+\sigma^2)$$

Now

$$E[f(x_*)|X,Y] = E[\phi^T(x_*)\phi|X,Y] = \phi^T(x_*)E[\theta|X,Y] = \phi^T(x_*)m_N$$

and

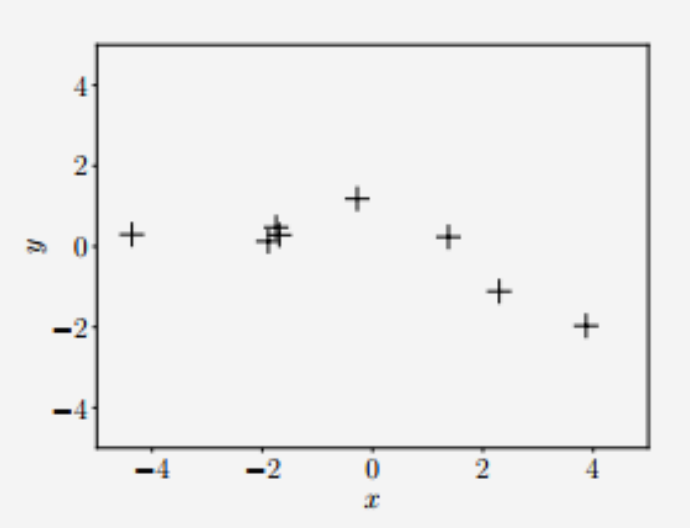
$$Var[f(x_*)|X,Y] = Var[\phi^T(x_*)\theta|X,Y] = \phi^T(x_*)Var[\theta|X,Y]\phi(x_*) = \phi^T(x_*)S_N\phi(x_*)$$

Therefore,

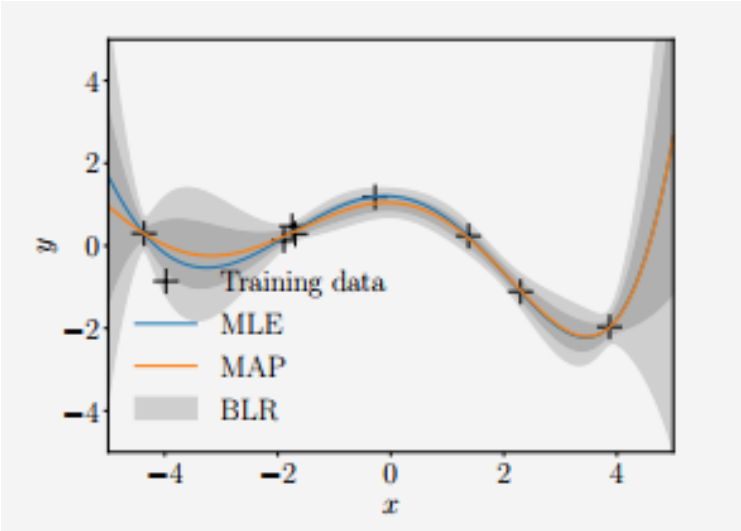
$$f(x_*) = \mathcal{N}(\phi^T(x_*)m_N, \phi^T(x_*)S_N\phi(x_*))$$

We see that the mean function will be $\phi^T(x)m_N$ which will coincide with the MAP estimate.

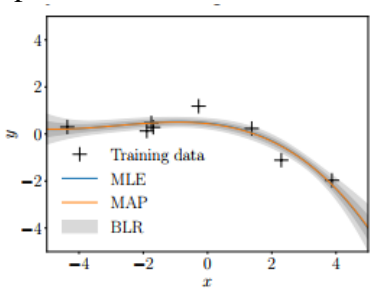
Example. Suppose this is the training data

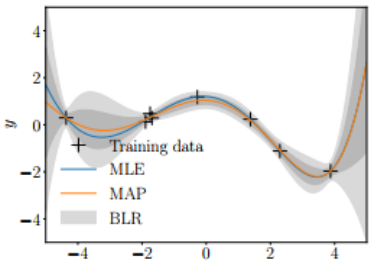


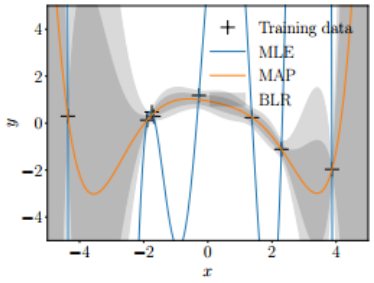
Now consider this graph:



Shaded area represents the distribution of the functions. Also the mean of this distribution will correspond to the MAP estimate curve. Now, consider the following graphs







The first graph is for the polynomial of degree 3, the second graph is for polynomial of degree 5 and third is for degree 7. From these graphs, we can clearly see that as the polynomial degree increases the posterior uncertainty increases rapidly. Hence, we can conclude that the BLR(Bayesian Linear Regression) can

help us to find the extent to which the uncertainty will occur in making predictions from the MAP estimate as the MAP estimate is the mean of the posterior distribution. Hence, if we are given a training data and we need to fit a function to it

1. Then we can start by using MLE estimate to get some sort of unregularized function to it.
2. Then we can go with MAP estimate to regularize the MLE estimate.
3. Then we can go with BLR to compare the uncertainity between different models and get the distribution over functions.

2.10 Marginal Likelihood

We saw in Bayesian linear regression while making posterior predictions **(2.5)** that $p(Y|X)$ is called the marginal likelohood, and it is used to normalize $p(\theta|X,Y)$.

$$p(\theta|X,Y) = \frac{p(Y|X,\theta)p(\theta)}{p(Y|X)}$$

and

$$\begin{aligned} p(Y|X) &= \int p(Y|X,\theta)p(\theta)d\theta \\ &= E_\theta[p(Y|X,\theta)] \end{aligned}$$

Hence, the marginal likelihood can be interpreted as the average value of likelihood $p(Y|X,\theta)$ for all the plausible values of θ over the prior. Now, the given model is

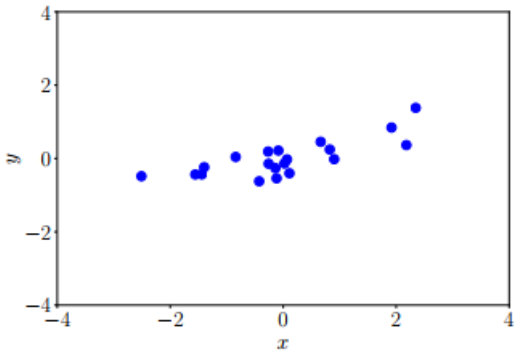
$$\begin{aligned} p(\theta) &= \mathcal{N}(m_o, S_o) \\ p(Y|X,\theta) &= \mathcal{N}(X\theta, \sigma^2 I) \end{aligned}$$

Hence, solving the above equations we get the marginal likelihood as

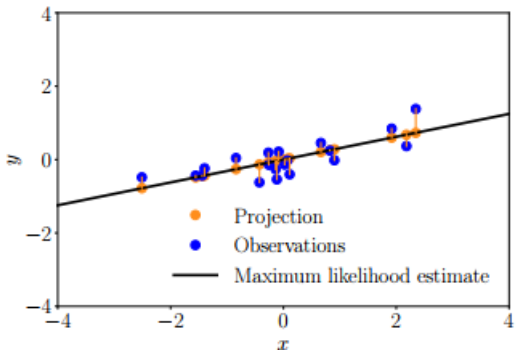
$$p(Y|X) = \mathcal{N}(y|Xm_o, XS_oX^T + \sigma^2 I)$$

2.11 Maximum Likelihood As Orthogonal Projection

Given the noisy dataset



We can fit a line to this data by finding out $\theta_{ML} = (X^T X)^{-1} X^T y$. During our derivation of this formula we aimed at maximizing the squared errors. Geometrically, we can see that



We have to minimize the orthogonal projection of the given data onto the line.

3 Main Results

An insurance company has the dataset of the charges they charge for their insurance for a person considering the following features:

1. Age
2. Sex
3. BMI
4. Number of Children
5. If he/she is a smoker
6. Region

Link for the dataset - *Insurance Datset*. So we have to train a regression model to predict the insurance price for a new person given his/her data.

Importing the required libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import pearsonr
```

Loading the Dataset

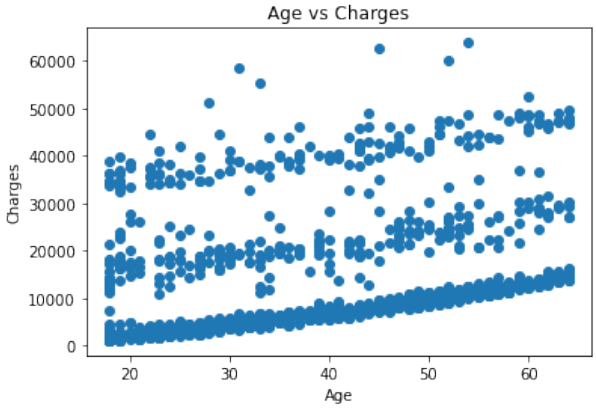
```
df=pd.read_csv('insurance.csv')
df.head()
```

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

Let us analyse the effect of Age on Charges

Age Vs Charges

```
plt.scatter(df['age'], df['charges'])
plt.ylabel('Charges')
plt.xlabel('Age')
plt.title('Age vs Charges')
```

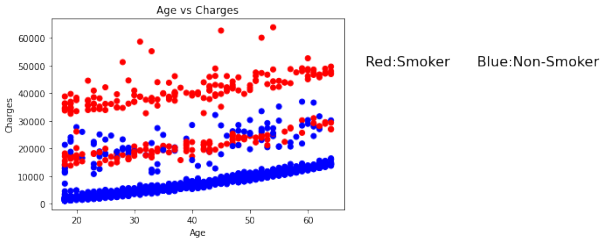



```
corr, _ = pearsonr(df['age'],df['charges'])
print("Correlation between AGE and CHARGES ---> ", corr)
```

```
Correlation between AGE and CHARGES --->  0.29900819333064765
```

We can see that the graph is divided into 3 parts. Let us see the effect of smoker and non-smoker on the same graph.

```
colors={'yes':'red','no':'blue'}
plt.scatter(df['age'], df['charges'],c=df['smoker'].map(colors))
plt.ylabel('Charges')
plt.xlabel('Age')
plt.title('Age vs Charges')
plt.text(x=70, y=50000,s='Red:Smoker      Blue:Non-Smoker', fontsize=16)
```



Hence we can see that we can segregate the data for smoker and non-smoker and train different models on both. Before segregating the data let us define the functions for the models that we will use to train using the results from the literature survey.

```
def model(x,Y, regularization=0, power=1):
  X=np.ones(x.shape[0])
  temp=np.ones(x.shape[0])
  for i in range(power):
    temp=temp*x
    X=np.c_[X,temp]
  theta=np.dot(np.dot(np.linalg.inv(np.dot(X.T,X)+regularization),X.T),Y)
  Y_pred=np.dot(X,theta)
  loss=np.sum(np.square(Y-Y_pred))/x.shape[0] + regularization*np.sum(np.square
  (theta))
  return theta,Y_pred,loss
```

```
def model1(x,Y, regularization=0, power=1):
  temp=np.ones(x.shape[0])
  X=np.c_[x,temp]
  theta=np.dot(np.dot(np.linalg.inv(np.dot(X.T,X)+regularization),X.T),Y)
  Y_pred=np.dot(X,theta)
  loss=np.sum(np.square(Y-Y_pred))/x.shape[0] + regularization*np.sum(np.square
  (theta))
  return theta,Y_pred,loss
```

```
def graph(x,Y,power,df_smoker):
  theta, Y_pred, loss=model(x,Y,regularization=0,power=power)
  plt.scatter(df_smoker['age'], df_smoker['charges'])
  plt.plot(x,Y_pred,'r+')
  plt.ylabel('Charges')
  plt.xlabel('Age')
  plt.title('Age vs Charges')
  print('loss = ',loss)
```

Smokers

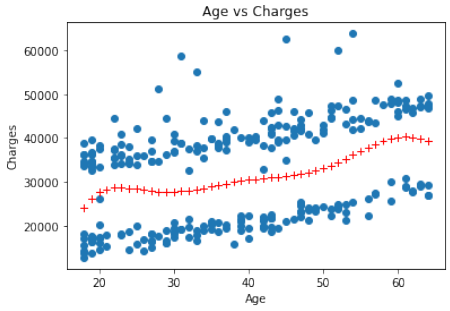
```
df_smoker=df[df['smoker']=='yes']
x=df_smoker['age'].to_numpy()
Y=df_smoker['charges'].to_numpy()
```

```
corr, _ = pearsonr(df_smoker['age'],df_smoker['charges'])
print("Correlation b/w AGE and CHARGES for smokers", corr)
```

```
Correlation b/w AGE and CHARGES for smokers 0.3682244437307778
```

Applying the model on the smokers dataset

```
graph(x,Y,10,df_smoker)
```



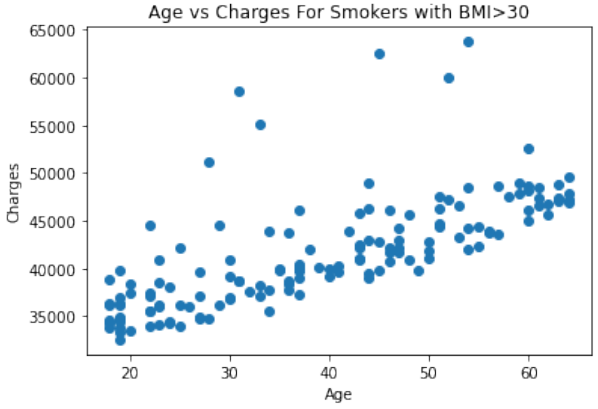
We see that our graph is still distributed into two parts. So we can segregate it more by dividing it into two parts -

1. People with BMI<=30
2. People with BMI>30

```
df_smoker_high=df_smoker[df_smoker['bmi']>30]
df_smoker_low=df_smoker[df_smoker['bmi']<=30]
```

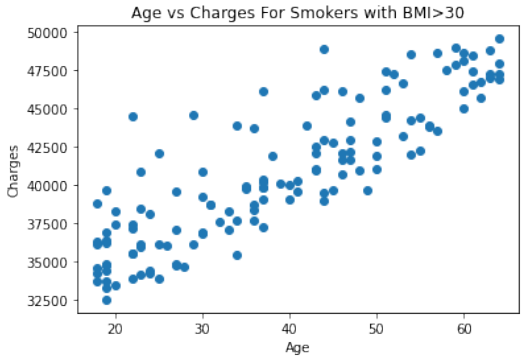
Smokers with BMI>30

```
plt.scatter(df_smoker_high['age'], df_smoker_high['charges'])
plt.ylabel('Charges')
plt.xlabel('Age')
plt.title('Age vs Charges For Smokers with BMI>30')
```



We can see that this is a good distribution for our linear regression model. We can improve a bit by removing the points with charges >50,000. This will improve our model.

```
df_smoker_high=df_smoker_high[df_smoker_high['charges']<=50000]
plt.scatter(df_smoker_high['age'], df_smoker_high['charges'])
plt.ylabel('Charges')
plt.xlabel('Age')
plt.title('Age vs Charges For Smokers with BMI>30')
```



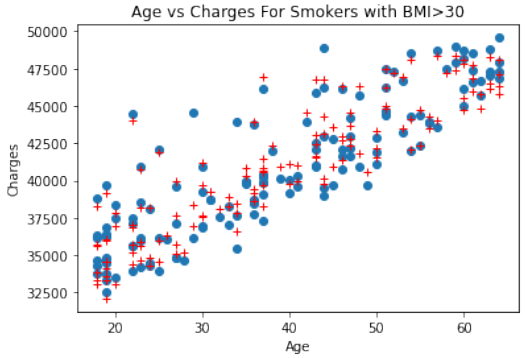

```
corr, _ = pearsonr(df_smoker_high['age'],df_smoker_high['charges'])
print("Correlation b/w AGE and CHARGES", corr)
```

```
Correlation b/w AGE and CHARGES 0.8623745411239981
```

We can see that we are having a high correlation as compared to previous ones. Now, training the model for smokers with BMI>30.

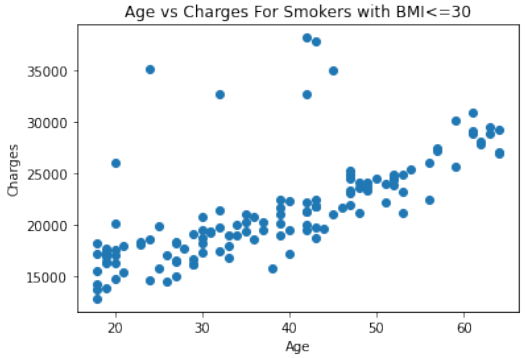
```
df_smoker_high=df_smoker_high.replace({'male':0, 'female':1, 'northwest':0, '
  northeast':1, 'southwest':2, 'southeast':3})
x=df_smoker_high.drop(columns=['charges','smoker']).to_numpy()
Y=df_smoker_high['charges'].to_numpy()
theta1,Y_pred,loss = model1(x,Y)
```

```
xp=df_smoker_high['age'].to_numpy()
plt.scatter(df_smoker_high['age'], df_smoker_high['charges'])
plt.plot(xp,Y_pred,'r+')
plt.ylabel('Charges')
plt.xlabel('Age')
plt.title('Age vs Charges For Smokers with BMI>30')
```



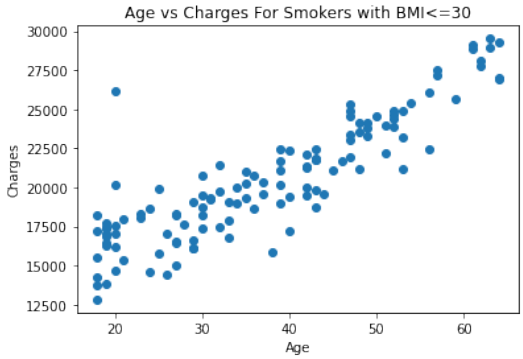
Smokers with BMI<=30

```
plt.scatter(df_smoker_low['age'], df_smoker_low['charges'])
plt.ylabel('Charges')
plt.xlabel('Age')
plt.title('Age vs Charges For Smokers with BMI<=30')
```



This is good but we can improve by putting an upper limit on charges i.e. 30,000.

```
df_smoker_low=df_smoker_low[df_smoker_low['charges']<=30000]
plt.scatter(df_smoker_low['age'], df_smoker_low['charges'])
plt.ylabel('Charges')
plt.xlabel('Age')
plt.title('Age vs Charges For Smokers with BMI<=30')
```

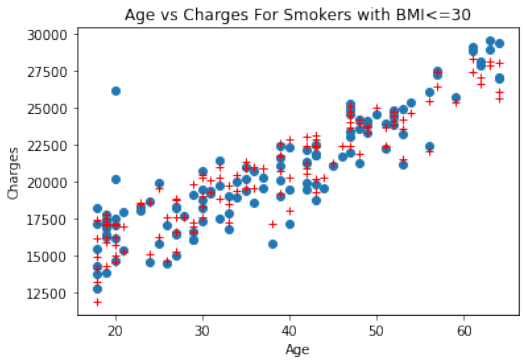



```
corr, _ = pearsonr(df_smoker_high['age'],df_smoker_high['charges'])
print("Correlation b/w AGE and CHARGES", corr)
```

```
Correlation b/w AGE and CHARGES 0.8623745411239981
```

We are having a good correlation, so we will train the model for this also..

```
df_smoker_low=df_smoker_low.replace({'male':0, 'female':1, 'northwest':0, '
  northeast':1, 'southwest':2, 'southeast':3})
x=df_smoker_low.drop(columns=['charges','smoker']).to_numpy()
Y=df_smoker_low['charges'].to_numpy()
theta2,Y_pred,loss = model1(x,Y)
xp=df_smoker_low['age'].to_numpy()
plt.scatter(df_smoker_low['age'], df_smoker_low['charges'])
plt.plot(xp,Y_pred,'r+')
plt.ylabel('Charges')
plt.xlabel('Age')
plt.title('Age vs Charges For Smokers with BMI<=30')
```

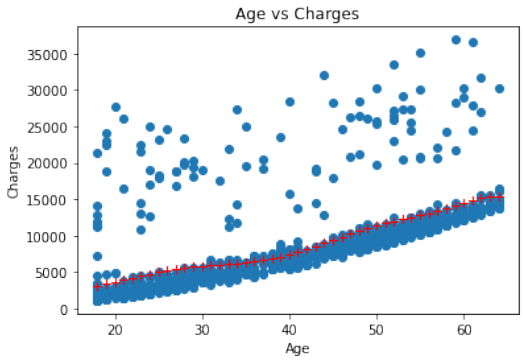


Non-Smokers

```
df_non_smoker=df[df['smoker']=='no']
x=df_non_smoker['age'].to_numpy()
Y=df_non_smoker['charges'].to_numpy()
corr, _ = pearsonr(df_non_smoker['age'],df_non_smoker['charges'])
print("Correlation b/w AGE and CHARGES for non-smokers", corr)
```

```
Correlation b/w AGE and CHARGES for non-smokers 0.6279467837664197
```

```
graph(x,Y,8,df_non_smoker)
```

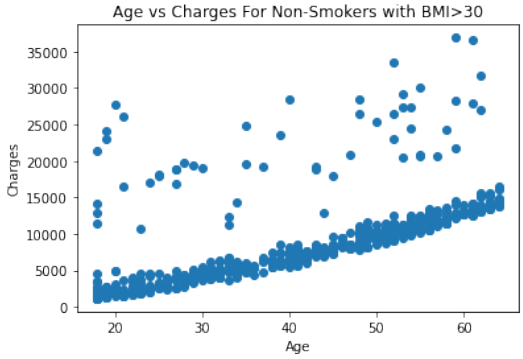


We see that though the model is fine but we can try to segregate it similarly using BMI..

```
df_non_smoker_high=df_non_smoker[df_non_smoker['bmi']>30]
df_non_smoker_low=df_non_smoker[df_non_smoker['bmi']<=30]
```

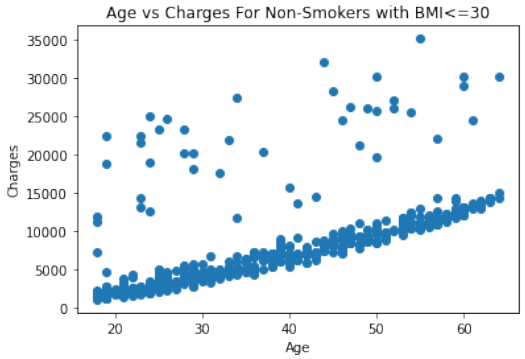
Non-Smokers with BMI>30

```
plt.scatter(df_non_smoker_high['age'], df_non_smoker_high['charges'])
plt.ylabel('Charges')
plt.xlabel('Age')
plt.title('Age vs Charges For Non-Smokers with BMI>30')
```



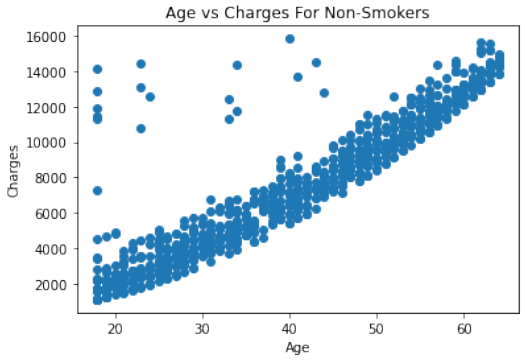
Non-Smokers with BMI<=30

```
plt.scatter(df_non_smoker_low['age'], df_non_smoker_low['charges'])
plt.ylabel('Charges')
plt.xlabel('Age')
plt.title('Age vs Charges For Non-Smokers with BMI<=30')
```



We see that we are unable to segregate the data on the basis of BMI.
So, lets put an upper bound of 16,000 on charges on the non-smoker dataset

```
df_non_smoker=df_non_smoker[df_non_smoker['charges']<=16000]
plt.scatter(df_non_smoker['age'], df_non_smoker['charges'])
plt.ylabel('Charges')
plt.xlabel('Age')
plt.title('Age vs Charges For Non-Smokers')
```



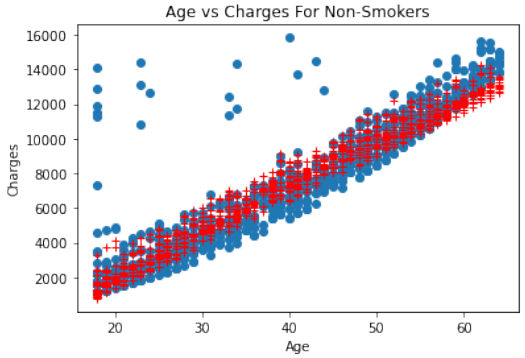

```
corr, _ = pearsonr(df_non_smoker['age'],df_non_smoker['charges'])
print("Correlation b/w AGE and CHARGES for non-smokers", corr)
```

```
Correlation b/w AGE and CHARGES for non-smokers 0.9287459037578168
```

High correlation means we can train a model for this.

```
df_non_smoker=df_non_smoker.replace({'male':0, 'female':1, 'northwest':0, '
  northeast':1, 'southwest':2, 'southeast':3})
x=df_non_smoker.drop(columns=['charges','smoker']).to_numpy()
Y=df_non_smoker['charges'].to_numpy()
theta3,Y_pred,loss = model1(x,Y)
xp=df_non_smoker['age'].to_numpy()
plt.scatter(df_non_smoker['age'], df_non_smoker['charges'])
plt.plot(xp,Y_pred,'r+')
```

```
plt.ylabel('Charges')
plt.xlabel('Age')
plt.title('Age vs Charges For Non-Smokers')
```



Making Predictions

For making predictions we will pass a numpy array to the following function with following entries in the mentioned order:

1. Age
2. 0 if Male, 1 is Female
3. BMI
4. Number of children
5. 1 if Smoker, 0 if Non-Smoker
6. Region: 0 if NorthWest, 1 if NorthEast, 2 if SouthWest, 3 if SouthEast

```
def predict(x):
  yp=[]
  x1=np.append(x,[1])
  x1=np.append(x1[:4],x1[5:])
  if x[4]==1:
    if x[2]>30:
      yp=np.dot(x1,theta1)
    else:
      yp=np.dot(x1,theta2)
  else:
    yp=np.dot(x1,theta3)
  return yp
```

4 CONCLUSION

Based on the understandings and results from the Literature Survey we trained a model that predicts the insurance price for a person given his Age, Sex, BMI, Number of Children, Smoking Status and Region. First, we analysed the dataset. The given data was too much scattered to be able to get a good model that fits it. So we segregated the data for smokers and non-smokers separately. Then we further segregated it using BMI. Still some scattered points were left due to unknown reasons of a particular person so we made a upper-bound to the data. This way we trained the model and finally we were able to get a decent result.