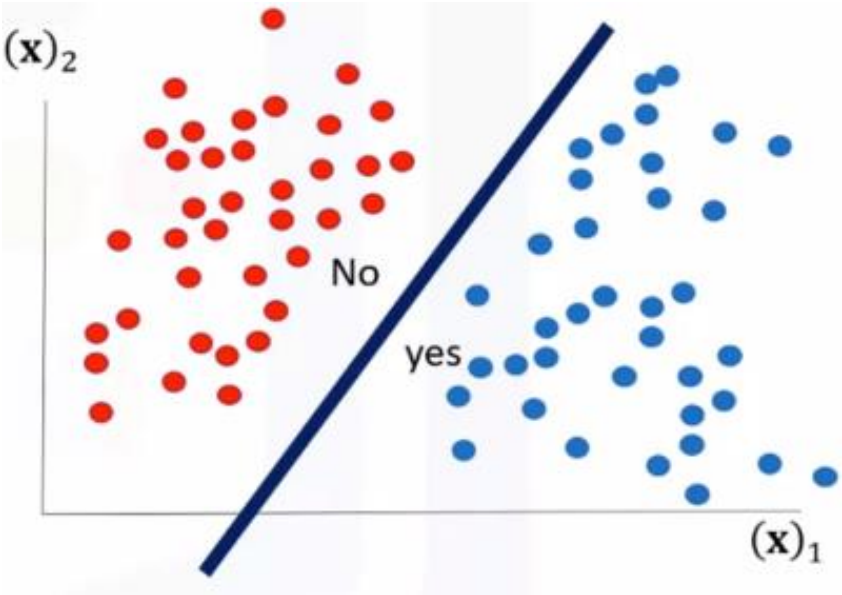
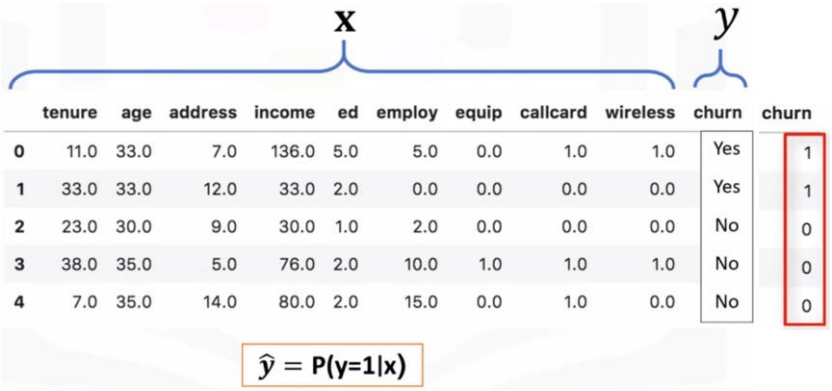
LOGISTIC REGRESSION



Logistic Regression can be described as

- A statistical and Machine Learning technique for classifying records of a dataset based on the values of the input fields or in short Logistic Regression is a classification algorithm for categorical variables
- In Logistic Regression we use one or more independent variables to predict an outcome or dependent variable
- This regression is analogous to Linear Regression but tries to predict a categorical (variable which is binary) or discrete target field instead of a numeric one
- In Logistic Regression, dependent variables should be continuous value and if categorical value then they should be dummy or indicator coded
- Logistic Regression can be used for both binary classification and multi-class classification
- We should use Logistic Regression when:
    - The target field in the data is categorical or specifically is binary
    - We need the probabilistic results of our prediction
        - This regression returns a probability score between 0 and 1 for a give sample data
    - We need a linear decision boundary and our data is linearly separable
        - The decision boundary of Logistic regression is a line or a plane or a hyperplane
    - We need to understand the impact of a feature
        - We can select the best features based on the statistical significance of the Logistic Regression model coefficients or parameters
        - That is, after finding the optimum parameters a feature $\boldsymbol{x}$ with the weight $\boldsymbol{\beta_1}$ close to 0 has a smaller effect on the prediction than features with large absolute values of $\boldsymbol{\beta_1}$
        - Indeed, it allows us to understand the impact an independent variable has on the dependent variable while controlling other independent variables

Logistic Regression vs Linear Regression



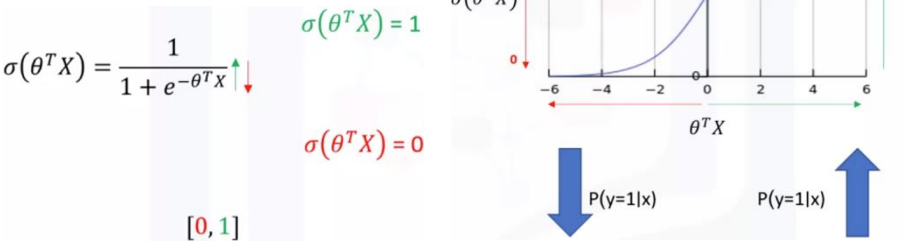
	tenure	age	address	income	ed	employ	equip	callcard	wireless	churn	churn
0	11.0	33.0	7.0	136.0	5.0	5.0	0.0	1.0	1.0	Yes	1
1	33.0	33.0	12.0	33.0	2.0	0.0	0.0	0.0	0.0	Yes	1
2	23.0	30.0	9.0	30.0	1.0	2.0	0.0	0.0	0.0	No	0
3	38.0	35.0	5.0	76.0	2.0	10.0	1.0	1.0	1.0	No	0
4	7.0	35.0	14.0	80.0	2.0	15.0	0.0	1.0	0.0	No	0

We'll take an example as for the explanation:

- Let's take a look at the dataset above, the goal of Logistic Regression is to build a model to predict the class of each observation and also the probability of each sample belonging to a class
- Ideally, we want to build a model $\hat{y}$ that can estimate that the class of an observation is $\mathbf{1}$ given its feature is $\boldsymbol{x}$ and $\boldsymbol{y}$ is the label's vector also called actual values that we would like to predict and $\hat{y}$ is the vector of the predicted values by our model
- Mapping the class labels of the target value from categorical into integer numbers
- If we using Linear Regression algorithm to predict our dataset using equation $\beta^T X = \beta_0 + \beta_1 x_1 + \cdots$ then we probably get the predicted value either smaller than 0 or bigger than 1
- Instead of having the step function in above, it would be nice if we had a smoother line one that would project predict values between 0 and 1
    - The scientific solution for this is a function called **Sigmoid** that changes traditional Linear Regression into $\boldsymbol{\sigma(\beta^T X) = \sigma(\beta_0 + \beta_1 x_1 + \cdots)}$ which gives us the probability of a point belonging to a class instead of the value of $\boldsymbol{y}$ directly
    - This function always returns a value between 0 and 1 depending on how large the $\boldsymbol{\beta^T X}$ actually is
    - And now $\boldsymbol{\hat{y} = \sigma(\beta^T X)}$ is our model for Logistic Regression which is Sigmoid of $\boldsymbol{\beta^T X}$ and represents the probability that the output is 1 given x – $\boldsymbol{P(y = 1|x)}$

Sigmoid function can be described as

- Also called the Logistic function, resembles the step function and is used by the following expression in the Logistic Regression
    - $\sigma(\beta^T X) = \frac{1}{1+e^{-\beta^T X}}$



- Notice in this equation, when $\boldsymbol{\beta^T X}$ gets very big the $\boldsymbol{e^{-\beta^T X}}$ in the denominator of the fraction becomes almost 0 and the value of the Sigmoid function gets closer to 1
- If $\boldsymbol{\beta^T X}$ is very small the Sigmoid function gets closer to 0
- So, the Sigmoid functions output is always between 0 and 1 which makes it proper to interpret the results as probabilities

- The output of our model when we use the Sigmoid function are
  - $P(Y = 1|X)$
    - Example - $P(target = 1|feat1, feat2) = 0.8$
  - $P(Y = 0|X) = 1 - P(Y = 1|X)$
    - Example – $P(target = 0|feat1, feat2) = 1 - 0.8 = 0.2$
- Now our job is to train the model to set its parameter values in such a way that our model is a good estimate of probability of y equal 1 given x – $P(y = 1|x)$
- In fact, this is what a good classifier model built by Logistic Regression is supposed to do for us also it should be a good estimate of probability y belongs to class 0 given x that can be shown as $1 - \sigma(\beta^T X)$

We can achieve $\boldsymbol{\sigma(\beta^T X) \rightarrow P(y = 1|x)}$ by finding $\boldsymbol{\beta}$ through the **training process**

1. Initialize $\boldsymbol{\beta}$ vector with random values as with most ML algorithm
2. Calculate the model output which is $\hat{\boldsymbol{y}} = \boldsymbol{\sigma(\beta^T X)}$ for observation in our training set
   - $\boldsymbol{X}$ is the feature vector values
     - Example – $\boldsymbol{X} = [\mathbf{2}, \mathbf{5}]$ >> these values we take from 2 features
   - And is $\boldsymbol{\beta}$ the confidence or weight that we've set in the previous step
     - Example – $\boldsymbol{\beta} = [-\mathbf{1}, \mathbf{2}]$
   - The output of this equation is the prediction value, in other words the probability that the observation belongs to class 1
     - Example – $\hat{\boldsymbol{y}} = \boldsymbol{\sigma}([-\mathbf{1}, \mathbf{2}] \times [\mathbf{2}, \mathbf{5}]) = \mathbf{0.7}$
3. Compare the output of $\hat{\boldsymbol{y}}$ from our model with the actual label of the observation, $\boldsymbol{y}$, by subtracting them and record it as error – $\boldsymbol{error = y - \hat{y}}$
4. Calculate the error for all observations as we did in the precious step and add up these errors
   - The total error is the cost of our model and is calculated by the models **Cost function** $\boldsymbol{Cost = J(\beta)}$
   - The Cost function basically represent how to calculate the error of the model which is the difference between the actual and the models predicted values
   - The Cost shows how poorly the model is estimating the customers labels
   - Therefore, **the lower the Cost the better the model is** at estimating the observations label correctly so what we want to do is try to minimize this Cost
5. Because the initial values for $\boldsymbol{\beta}$ were chosen randomly, it's very likely that the Cost function is very high so we change the $\boldsymbol{\beta_{new}}$ in such a way to hopefully reduce the total cost
6. After changing the values of $\boldsymbol{\beta}$, we go back to step 2 then we start another iteration and calculate the Cost of the model again and keep doing those steps over and over until the Cost is low enough

The 2 main questions from all the processes above are:

- How can we change the values of $\boldsymbol{\beta}$ so that the cost is reduced across iterations?
  - There are different ways to change the values of $\boldsymbol{\beta}$ but one of the most popular ways is **Gradient Descent**
- When should we stop the iteration?
  - There are various ways to stop iterations but essentially, we stop training by calculating the accuracy of our model and stop it when it's satisfactory

Logistic Regression – Training

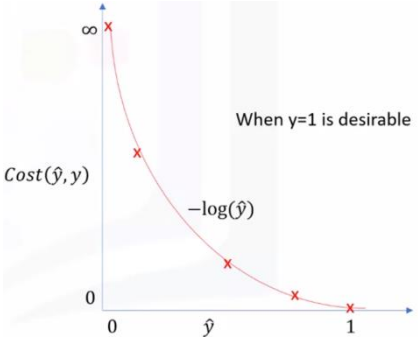
In this part, our mainly focuses are:

- Learn more about training a logistic regression model
- Know how to change the parameters of the model to better estimate the outcome
- Define and describe about the Cost function and Gradient Descent in Logistic Regression as a way to optimize the model

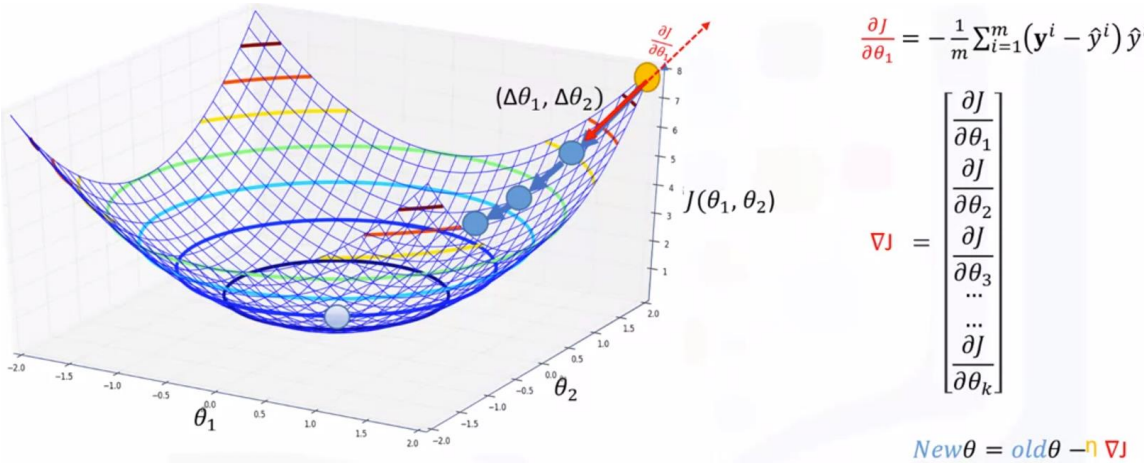
---

The main objective of **Training** in Logistic Regression is to change the parameters of the model so as to be the best estimation of the labels of the samples in the dataset and this is how we do

- First, we have to look at the Cost function and see what the relation is between the Cost function and the parameters $\boldsymbol{\beta}$
    - We should formulate the Cost function
- Then, using the derivative of the Cost function we can find how to change the parameters to reduce the cost or rather the error
- Now dive into it to see how it works:
    - First find the Cost function equation for a sample case
        - $Cost(\hat{y}, y) = \frac{1}{2}(\sigma(\beta^T X) - y)^2$
        - There is normally a general equation for calculating the cost
        - This is a general rule for most Cost functions in ML
        - The cost function is the difference between the actual values and predicted value of our model
        - We can show this as the cost of our model comparing it with actual labels, which is the difference between the predicted value of our model and actual value of the target field where the predicted value of our model is $\boldsymbol{\beta^T X}$ as
        - Usually the square of this equation is used because of the possibility of the negative result
        - For the sake of simplicity half of this value is considered as the Cost function through the derivative process
        - Now we can write the Cost function for all the samples in our training set
            - $J(\beta) = \frac{1}{m}\sum_{i-1}^{m} Cost(\hat{y}, y)$
        - For all observations we can write it as the average sum of the Cost functions of all cases and it is also called the **Mean Squared Error (MSE)**
        - And as it is a function of parameter vector $\boldsymbol{\beta}$, it shown as $\boldsymbol{J(\beta)}$
    - Next, we find or set the best weights or parameters that minimize the Cost function
        - We should calculate the minimum point of the Cost function and it will show us the best parameters for our model
        - Although we can find the minimum point of a function using the derivative of a function there is not an easy way to find the global minimum point for such an equation
        - The solution is **we should find another cost function instead**, one which has the same behavior but is easier to find its minimum point
        - Plot the desirable Cost function $\boldsymbol{Cost(\hat{y}, y)}$ for our model
        - Recall that our model is $\hat{\boldsymbol{y}}$ and our actual value is $\boldsymbol{y}$ which equals 0 or 1
        - Our model tries to estimate it as we want to find a simple Cost function for our model
        - Assume that our desired value for $\boldsymbol{y}$ is 1, this means our model is best if it estimates $\boldsymbol{y = 1}$



        - In this case, we need a Cost function that return 0 if the outcome of our model is 1 which is the same as the actual label
        - And the Cost should keep increasing as the outcome of our model gets farther from 1
        - The Cost should be very large ∞ if the outcome of our model is close to 0
        - We can see that the $-\boldsymbol{log}(\hat{\boldsymbol{y}})$ provides such a Cost function for us means if the actual value is 1 and the model also predicts 1 then the $-\boldsymbol{log}(\hat{\boldsymbol{y}})$ function returns 0 cost but if the prediction is smaller than 1 then the $-\boldsymbol{log}(\hat{\boldsymbol{y}})$ function returns a larger cost value
        - So, we can use the $-\boldsymbol{log}(\hat{\boldsymbol{y}})$ function for calculating the Cost of our Logistic Regression model
    - At this point we replace the previous Cost function $\boldsymbol{Cost}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{2}(\boldsymbol{\sigma}(\boldsymbol{\beta}^T\boldsymbol{X}) - \boldsymbol{y})^2$ with the new one $-\boldsymbol{log}(\hat{\boldsymbol{y}})$
        - $\boldsymbol{Cost}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\boldsymbol{log}(\hat{\boldsymbol{y}})$ if the desirable value of $\boldsymbol{y} = \boldsymbol{1}$
        - $\boldsymbol{Cost}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\boldsymbol{log}(\widehat{\boldsymbol{1-y}})$ if the desirable value of $\boldsymbol{y} = \boldsymbol{0}$
    - And now we can plug it into total Cost functions the Logistic Regression cost function and rewrite it as the following equation
        - $J(\beta) = -\frac{1}{m}\sum_{i-1}^{m} y^i \log(\hat{y}^i) + (1 - y^i)\log(1 - \hat{y}^i)$
- Next the thing that we have to do is using an optimization approach to minimize the Cost function and **Gradient Descent** is one of the most famous and effective approach
    - Gradient Descent is a technique to use the derivative of a Cost function to change the parameter values in order to minimize the cost or error
    - The main objective of Gradient Descent is to change the parameter values so as to minimize the cost
    - Think of the parameters or weights in our model to be in a 2-dimensional space
        - Example – $\hat{y} = \sigma(\beta_1 x_1 + \beta_2 x_2)$
    - Recall the Cost function that we already found before
        - $J(\beta) = -\frac{1}{m}\sum_{i-1}^{m} y^i \log(\hat{y}^i) + (1 - y^i)\log(1 - \hat{y}^i)$
    - We need to minimize the cost function $\boldsymbol{J}(\boldsymbol{\beta})$ which is a function of variables $\boldsymbol{\beta_1}$ and $\boldsymbol{\beta_2}$ so add a dimension for the observed cost or error $\boldsymbol{J}(\boldsymbol{\beta_1}, \boldsymbol{\beta_2})$



    - Let's assume that we plot the Cost function based on all possible values of $\boldsymbol{\beta_1}$ and $\boldsymbol{\beta_2}$ we can see the graph like on the above
    - It represents the error value for different values of parameters that is error which is a function of the parameters and this is called error curve/bowl of Cost function
    - To get the best point for our Cost function we should try to minimize our position on the error curve by **finding the minimum value of the cost by changing the parameters**
        - We can select random parameter values that locate a point on the bowl – let's say the starting point is the yellow point on the graph

- Change the parameters by $\Delta\boldsymbol{\beta_1}$ and $\Delta\boldsymbol{\beta_2}$ take one step on the surface – assume we go down one step in the bowl oh the graph
- As long as we're going downwards, we can go one more step
- The steeper the slope the further we can steep and we can keep taking steps
- As we approach the lowest point the slope diminishes so we can take smaller steps until we reach a flat surface
- The grey point is the minimum point of error curve and the optimum $\boldsymbol{\beta_1}$ and $\boldsymbol{\beta_2}$
- **To find the direction and size of those steps or in other words to find how to update the parameters, we should calculate the Gradient of the Cost function at that point**
- The Gradient is the slope of the surface at every point and the direction of the Gradient is the direction of the greatest uphill
- To calculate the Gradient of a Cost function at a point, – *let's say we take a random point the yellow one on the surface* – take the partial derivative of $\boldsymbol{J}(\boldsymbol{\beta_1}, \boldsymbol{\beta_2})$ with respect to each parameter at that point then it'll gives us the slope of the move for each parameter at that point and now if we move in the opposite direction of that slope, it guarantees that we go down in the error curve
    - Example – if we calculate the derivative of $\boldsymbol{J}$ with respect to $\boldsymbol{\beta_1}$ and $\boldsymbol{\beta_2}$ we'll find out that it is a positive number
    - This indicates that function is increasing as $\boldsymbol{\beta_1}$ increases
    - So, to decrease $\boldsymbol{J}$ we should move in the opposite direction and this means to move in the direction of the negative derivative for $\boldsymbol{\beta_1}$ (i.e. slope)
- We have to calculate it for other parameters as well at each step
- The Gradient value also indicates how big of a step to take
    - If the slope is large, we should take a large step because we are far from the minimum
    - And if the slope is small, we should take a smaller step
- Gradient Descent takes increasingly smaller steps towards the minimum with each iteration
- The partial derivative of the Cost function $\boldsymbol{J}$ is calculated using the following expression
    - $\frac{\partial J}{\partial \beta_1} = -\frac{1}{m}\sum_{i=1}^{m}\left(y^i \hat{y}^i\right)\hat{y}^i(1-\hat{y}^i)x_1$
- The equation above returns the slope of that point and we should update the parameter in the opposite direction of the slope
- A vector of all those slopes in the gradient vector $\boldsymbol{\nabla J}$ and we can use this vector to change/update all the parameters
- To calculate the Gradient vector, we need all the training data to feed the equation for each parameter
- We take the previous values of the parameters and subtract the error derivative with $\boldsymbol{New\beta} = \boldsymbol{Old\beta} - \boldsymbol{\eta\nabla J}$ and this results in the new parameters for $\boldsymbol{\beta}$ that we know will decrease the cost, also we multiply the Gradient value by $\boldsymbol{\eta}$ a constant value which called the **Learning Rate**
    - Learning rate give us additional control on how fast we move on the surface
    - In sum, we can simply say **Gradient Descent** is like taking steps in the current direction of the slope and the **Learning Rate** is the length of the step we take
    - Notice that it's an iterative operation and each iteration we update the parameters and minimize the cost until the algorithm converge is on an acceptable minimum
- When the best parameter has been found after some iterations means the model is ready and we can use it to predict the probability of an observation target value