# PRML ASSIGNMENT LAB 3-4

## Q.1 DECISION TREE :

**TASK 1:**

- In this task I have imported the data into a dataframe and preprocessed the data meaning I have cleaned the data by checking for null values and filling the null values by dividing them into groups, or checking for faults in our data by checking for outliers.
- Dropping the columns which do not have any impact on our output.
- Further I have encoded the categorical data wherever required.
- Categorical data :
- Nominal data :
- Ordinal data :
- To visualize the data I have plotted the strip plot.
- And further I have split the data into 70-20-10.

**TASK 2:**

In this task we had to create a function to calculate the entropy. We take unique values of the column and their counts. This function calculates the probability of each unique value in the column and thus calculates the entropy mathematically.

Formula used = -sum(prob * log2(prob))

**TASK 3 and 4:**

In these tasks we convert the continuous data to categorical data, we do this with the help of two functions i.e. div() and information_gain(). We did so since it becomes easy to make decision in decision tree. This task is done by div() function. With the help of information_gain() function we find the best split. It quantifies the effectiveness of a split in reducing uncertainty about the target variable. Features with higher information gain are prioritized for splitting.

**TASK 5:**

Here we have made a class of decision tree where we have one of its method as infer() function. The function starts by comparing the value of a specific feature (self.split_feature) of the input sample (test) with a predetermined split value (self.split_value) stored in the current node of the decision tree.If the feature value is greater than the split value, the function proceeds to the right subtree of the decision tree. It checks if there is a right subtree (self.right). If there is no right subtree (indicating a leaf node), the function returns the target class label (self.target). If a right subtree exists, the function recursively calls itself on the right subtree.This recursive process continues until a leaf node is reached, where there are no further left or right subtrees. At this point, the function returns the target class label associated with the leaf node.

**TASK 6:**

We have then computed the accuracy of the tree from testing set and the training set as a whole as well as class wise. We have made a function for doing so that computes the accuracy.

**TASK 7:**

True Positives (TP):
Instances where the model correctly predicts the positive class. In the context of survival prediction, this would be passengers correctly classified as "survived."

True Negatives (TN):
Instances where the model correctly predicts the negative class. In survival prediction, this corresponds to passengers correctly classified as "not survived."

False Positives (FP):
Instances where the model incorrectly predicts the positive class. For survival prediction, this represents passengers wrongly classified as "survived" when they did not.

False Negatives (FN):
Instances where the model incorrectly predicts the negative class. In this case, passengers wrongly classified as "not survived" when they did survive.

**TASK 8:**
From sklearn we import the fuctions we need to compute accuracy, precision and F1 score and using them we compute them.

## Q.2 Linear Regression :

**TASK 1:**

In this we load the dataset into our dataframe, analyze the data, plot the data in a scatter plot and see that with the increment of budget the sales are increasing too.

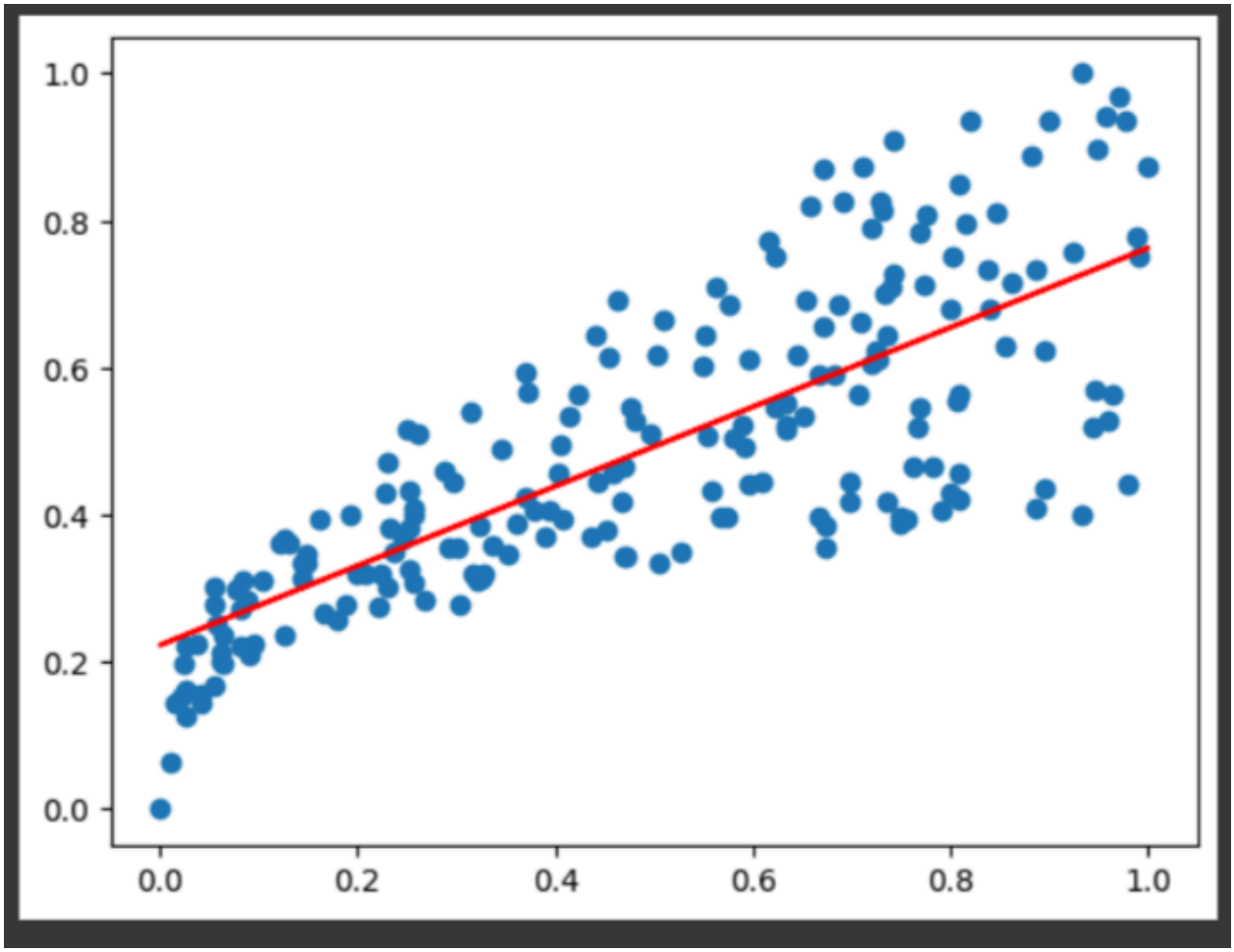

**TASK 2:**

Since there are no missing values in the data we simply normalize the data by min max normalization and further split the data into train and test. We normalize the data because our regression hypothesis needs to have scaled data.

**TASK 3:**

This task implements our model of linear regression, it takes two columns and number of iterations and the learning rate. We use the algorithm of gradient descent to minimize the cost function. We iterate the loop for a large number of times so that we can obtain the minimum of cost function. And it returns us the slope and intercept.

Obtaining the intercept and slop we can now plot the regression line.



**TASK 4:**

We now compute the mean squared error and absolute error from the test set by taking our line as reference.

```
Mean Squared Error: 0.016824022109584 43
absolute error: 0.10650439027916 35
```

# Q.3 Linear Regression (Multivariate) :

**TASK 1:**

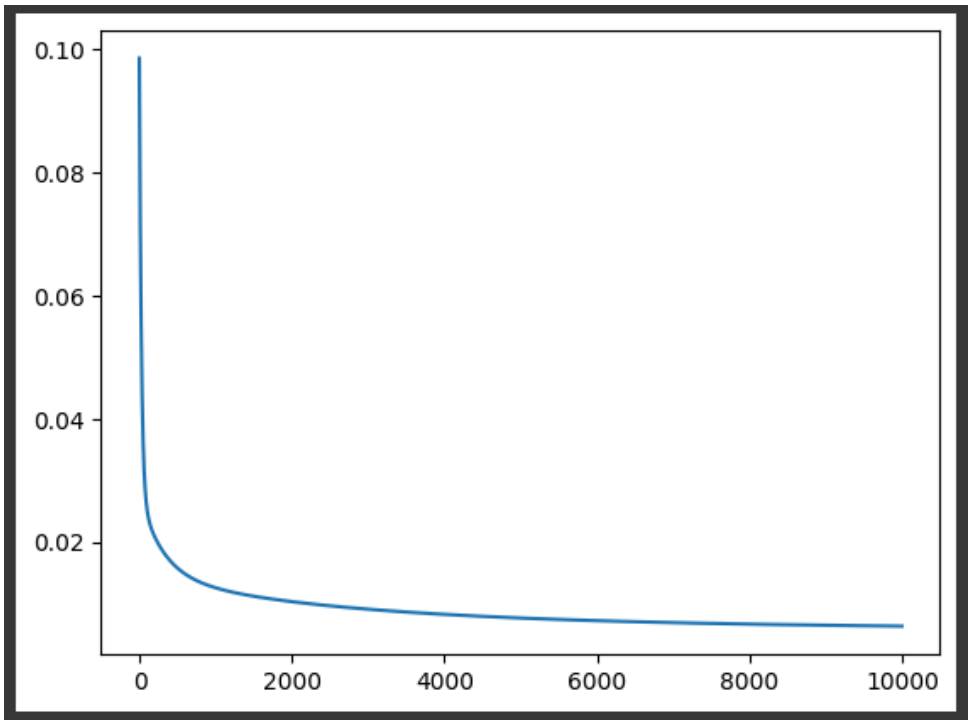
In this we load the dataset into our dataframe, analyze the data, plot the data in a scatter plot.

**TASK 2:**

Since there are no missing values in the data we simply normalize the data by min max normalization and further split the data into train and test.We normalize the data because our regression hypothesis needs to have scaled data.

**TASK 3:**

This task implements our model of linear regression, it takes a dataframe and a column and number of iterations and the learning rate. We use the algorithm of gradient descent to minimize the cost function. We iterate the loop for a large number of times so that we can obtain the minimum of cost function. And it returns us the slope and intercept. We can see the graph obtained where we see the value of cost function going down. With every 100 iterations.




**TASK 4:**

Now we compute the mean squared error and the absolute error and the accuracy of our model.

```
mean squared error :  0.012294930098656497
absolute error is : 0.0772196188238697
Test Accuracy is : 92.27803811761302 %
```