**Project 1:**

**The motive of the project:**

A few months back, around May, my father lost his credit card in the month beginning. He wasn’t aware of it but when the bill came at the end of the month, we had a huge bill to pay off and that is when we realized that the card had been stolen and we immediately informed the bank. This is where I thought Machine Learning could come in handy and I thought of identifying/predicting fraudulent credit card transactions as my self-learning project.

**Overview of the project:**

I started the project with data visualization and **univariate and bivariate analysis** and drew some important insights from it. Since the data was highly skewed, I performed **undersampling** on the data before creating and training the model. I then went on to train the model on multiple machine learning algorithms like **Logistic Regression, k-Nearest Neighbours, Decision Tree, and Random Forest** achieving a **classification recall of 93%**. I benchmarked the performance of the models on the basis of the **F1 scores** and **Confusion Matrix** and also plotted the **ROC curves** and reached the conclusion that Logistic Regression gives out the best results.

**Conclusions from EDA:**

1. The data consisted of around 2,85,000 data points, 30 features including time and amount, and the labeled class of whether a transaction is actually fraud or not.
2. There were no null values present in the original dataset but the data was highly skewed with 99.83% of the data points being non-fraudulent transactions.
3. The time feature had a bimodal distribution i.e. peaks falling and rising. I have concluded that the peaks might fall due to lesser transactions during nighttime.
4. Very small proportion of transactions had amounts > 10,000 hence they were eliminated from the dataset.
5. Most of the fraudulent transactions were of small amounts (<1000 units - since we don’t know about the units about the currency).
6. The occurrence of fraudulent transactions was independent of the time of the day.

**Univariate and Bivariate Analysis:**

Univariate analysis generally refers to the data analysis where there is only one dependent variable. The main goal of the univariate analysis is to summarize the data. We can easily identify measures of central tendency like mean, median, mode, the quartiles, and the standard deviation.

Bivariate analysis happens between 2 variables to identify the relationship between them. There are three types of bivariate analysis:

1. Numerical-Numerical (scatter plot, linear correlation)
2. Categorical-Categorical (Chi-Square)
3. Numerical-Categorical (Z-test, T-test)

**Undersampling in Layman terms:**

Undersampling refers to a technique designed to balance the class distribution of highly skewed classification data. An imbalance dataset is referred to as the case where we have one or more classes with few examples (the minority class) and one or more classes with many examples(the majority class). Undersampling involves removing examples from the training dataset of the majority class in order to balance the class distribution. In my case, I reduced it to a 1:1 class distribution of fraudulent to non-fraudulent transactions. I randomly selected examples from the majority class and deleted them from the training dataset (Random Undersampling).

Limitation: Examples are removed without any concern for how useful or important they might be in determining the decision boundary between the classes. This means it is possible, or even likely, that useful information will be deleted.

**Logistic Regression:**

It is one of the most common methods employed for binary classification. It makes use of the sigmoid or logistic function. It is an S-shaped graph that can take any real values between (0,1) but never touch the limits. The sigmoid function is:

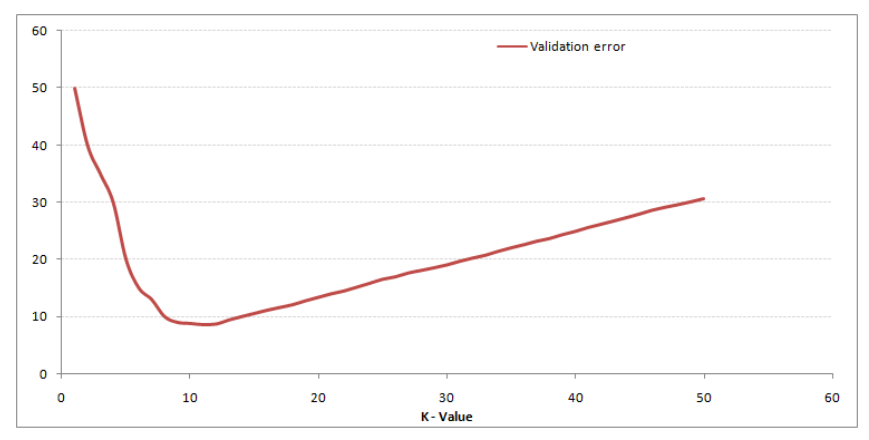
**f(x) = 1/(1+exp(-x))**

But in most cases our input values are combined linearly using weights or coefficients to predict values i.e. we will have **exp(-(b0+b1x))** in the denominator. Here b0 is the bias or intercept term and b1 is the coefficient for the input variable x. b0 and b1 are learned from your training data. In memory, the coefficients will be stored (b0, b1, and so on).

Note: The best coefficients would result in a model that would predict a value very close to 1 for the default class and a value very close to 0 for other values.

**K-Nearest Neighbours:**

KNN is a supervised machine learning algorithm that predicts the label of the unknown data point on the basis of the votes of its k-nearest neighbors. We draw a circular boundary such that it consists of k-nearest data points to the unknown data points and then we predict on the basis of the majority of the class predict in that area. Usually, k varies from 1 to infinity but in practical cases, k is always taken to be less than 30. Choosing a higher k will require either a high processing time or a costlier processor. Lower k will fail to generalize leading to



overfitting and higher k will make the entire process costlier. On higher k’s, the boundaries become smooth. But this causes some of the data points in the opposite labeled regions which is collateral damage. This accounts for the loss in training accuracy but better generalization and high test accuracy.

K-NN is a lazy learner because it doesn’t learn a discriminative function from the training data but memorizes the training dataset instead.

**Note:**

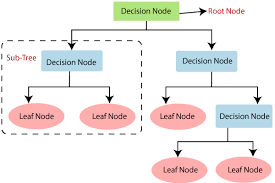
When training your model, you train it on the training data. Your performance on the training data/the training error does not tell you how well your model is overall, but only how well it has learned the training data.

The validation error tells you how well your learned model generalizes, which means how well it fits data that it has not been trained on. Generally, you try to train your model on the training data so that it has good results on the (unseen) validation data.

However, if you do that and optimize the training for the validation set, this is kind of training on the validation set. Not memorizing the validation set, but overfitting is still possible.

So if your model is finally trained and done, you can test it on the testing data and see how well it performs with completely unseen data. But you do not optimize it for the test set - if you do that, you don’t know how well it performs with unseen data.

**Decision Trees:**

They are used for predictive modeling machine learning.

Decision trees use some cost functions in order to choose the best split. It will try to find the best split/attribute that performs the best at classifying the training data and this is repeated until a leaf node is reached. Since the algorithm repeatedly partitions the data into smaller subsets, the final subsets (leaf nodes) consist of few or only one data point. This causes the algorithm to have low bias and high variance. It works in such as way as to maximize the purity of the classes when making splits. Entropy is calculated before and after the splits. If the entropy increases, another split is tried or the branch of the tree is stopped (when the current tree has the lowest entropy).

Entropy denotes the purity of the class. The higher the entropy, the lower will be the predictive power of the class.

**Random Forests:**

Random forest is a popular supervised machine learning algorithm that follows the principle of ensemble learning where it combines multiple classifiers, in our case, decision trees to solve complex problems and to improve the performance of the model.

Working:

1. Take k points from the training dataset
2. Build the decision tree on the basis of the selected points
3. Choose the number of decision trees you want to build
4. Repeat till N trees are created

The unknown data points are then predicted using all the decision trees and the class to the data point is assigned based on the majority voting from all the decision trees

Advantages:

1. Capable of performing both classification and regression tasks
2. Can handle huge dataset with high dimensionality
3. Enhances the accuracy of the model
4. Reduces the problem of overfitting in decision trees

Disadvantages:

1. High computational power
2. High time requirement for training as it combines multiple decision trees
3. Difficult to interpret and can’t understand the significance of each variable

**Recall:**

It is calculated as the ratio of correctly predicted positive examples divided by the total number of positive examples in the dataset.

***=> Recall = True Positive/(True Positive + False Negative)***

**Precision:**

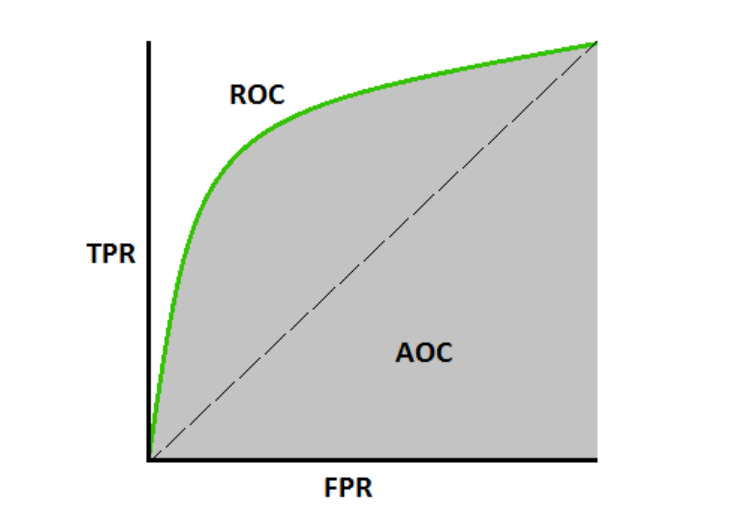
It is calculated as the ratio of correctly predicted positive examples divided by the total number of positive examples that were predicted.

***=> Precision = True Positive/(True Positive + False Positive)***

**Precision**: Appropriate when **minimizing false positives** is the focus  
**Recall**: Appropriate when **minimizing false negatives** is the focus.

**F1 scores:**

F1 score is the harmonic mean of precision and recall and is one of the most commonly used performance metrics in the case of highly imbalanced data. F1 score tries to seek a balance between recall and precision by taking the harmonic mean so that we can have a better analysis of our model. We don’t take the arithmetic mean because it would not punish the model for either bad recall or precision, hence it will not be a very good measure to benchmark the performance of the model.

**ROC-AUC curves:**

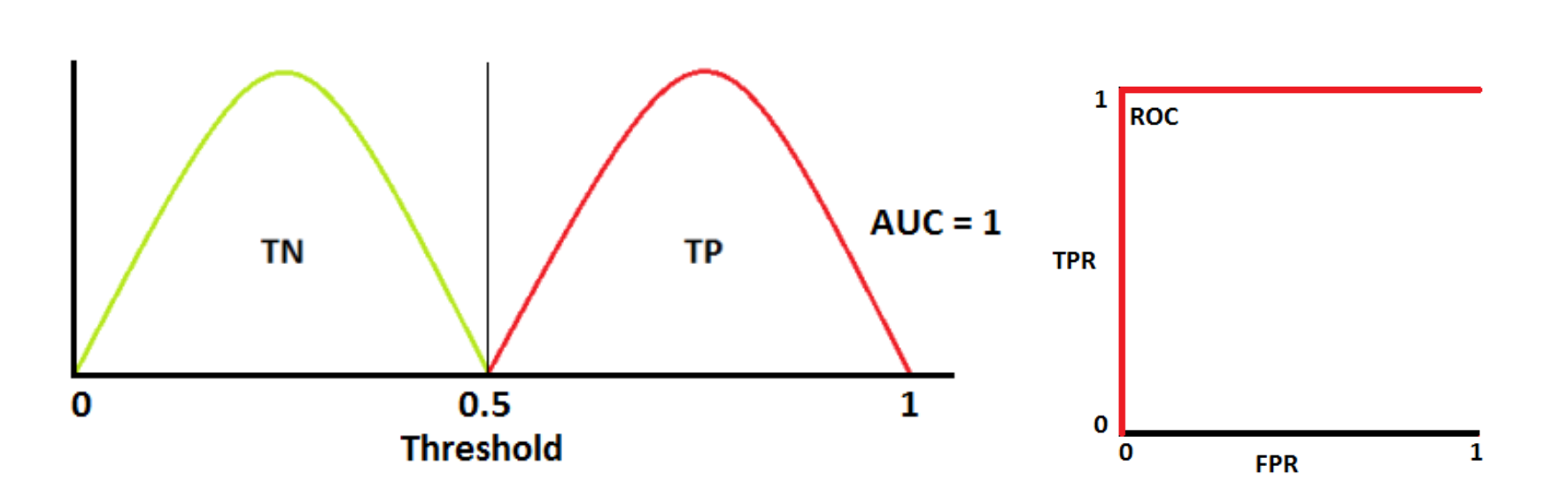
AUC-ROC curves are performance measurement curves for classification problems in machine learning. ROC is a probability curve and AUC represents the degree or measure of separability. It tells us how much the model is capable of distinguishing between classes. Higher the AUC, better is the model at predicting class 0 as 0 and class 1 as 1.

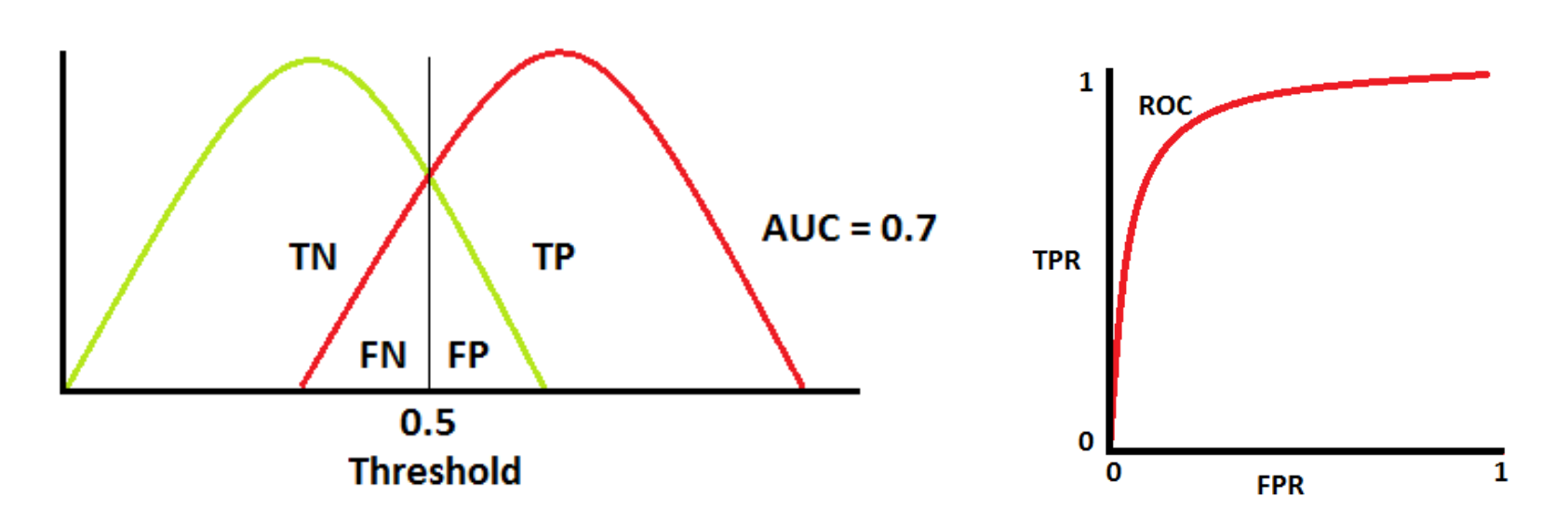
The ROC curve is plotted with TPR(True Positive Rate) on the y-axis and FPR(False Positive Rate) on the x-axis.

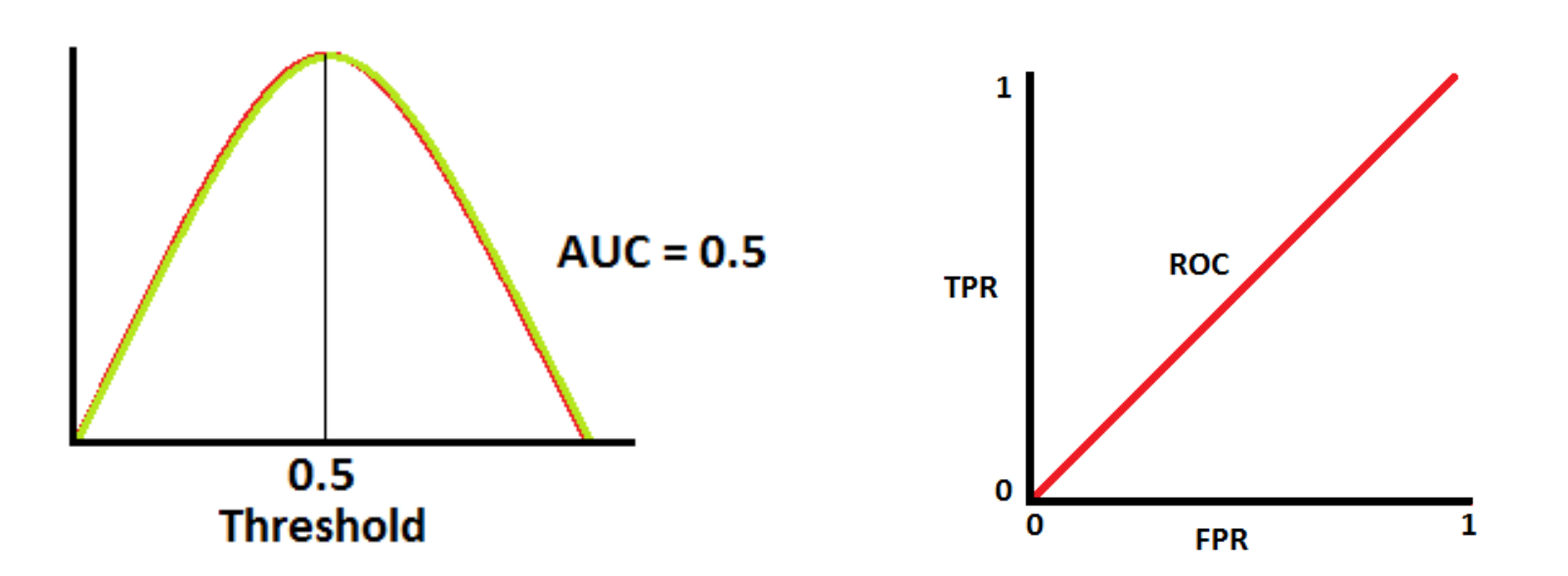
***TPR = True Positive/(True Positive + False Negative) = recall***

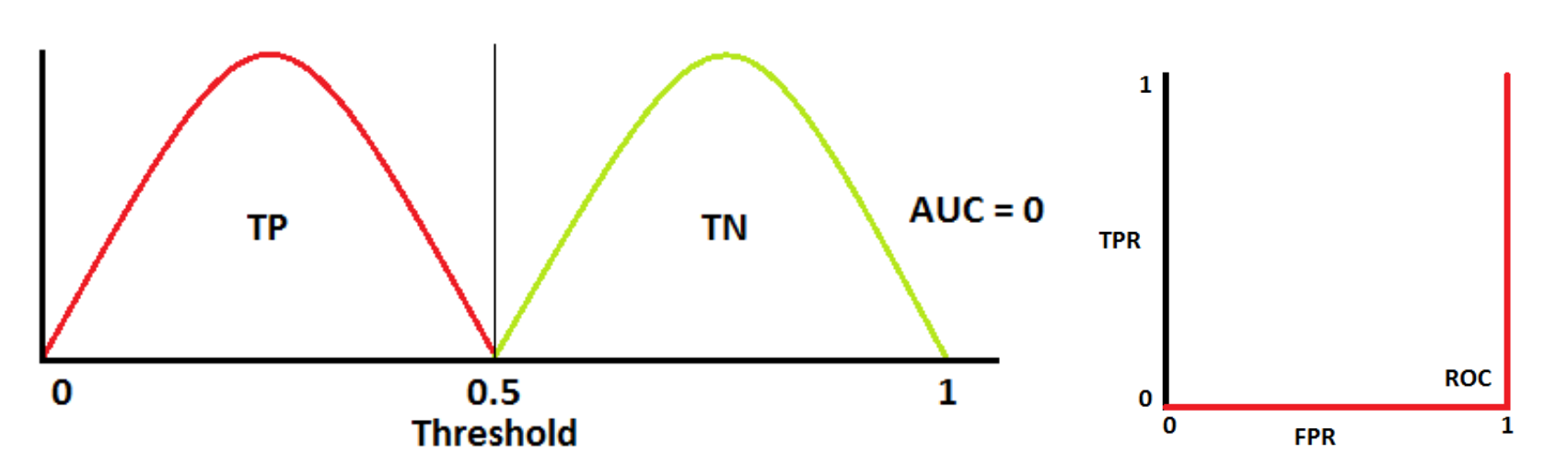
***FPR = False Positive/(False Positive + True Negative)  
Specificity = 1 - FPR***

An excellent model has AUC near to the 1 which means it has a good measure of separability. A poor model has an AUC near 0 which means it has the worst measure of separability. In fact, it means it is reciprocating the result. It is predicting 0s as 1s and 1s as 0s. And when AUC is 0.5, it means the model has no class separation capacity whatsoever.







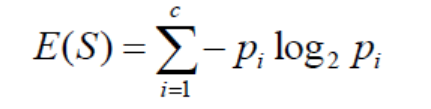




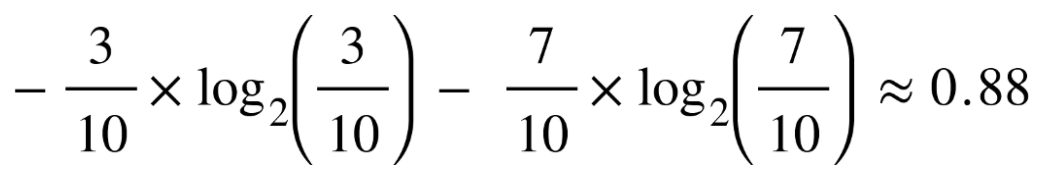
**Entropy:**

Entropy is a measure of disorder or uncertainty and the goal of machine learning models and Data Scientists in general is to reduce uncertainty.

Entropy in layman’s terms is a measure of disorder (or purity). Mathematical formula:

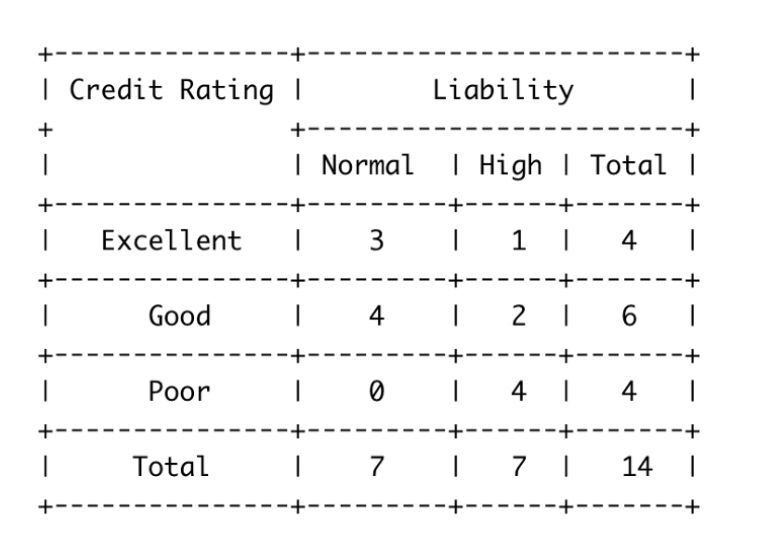


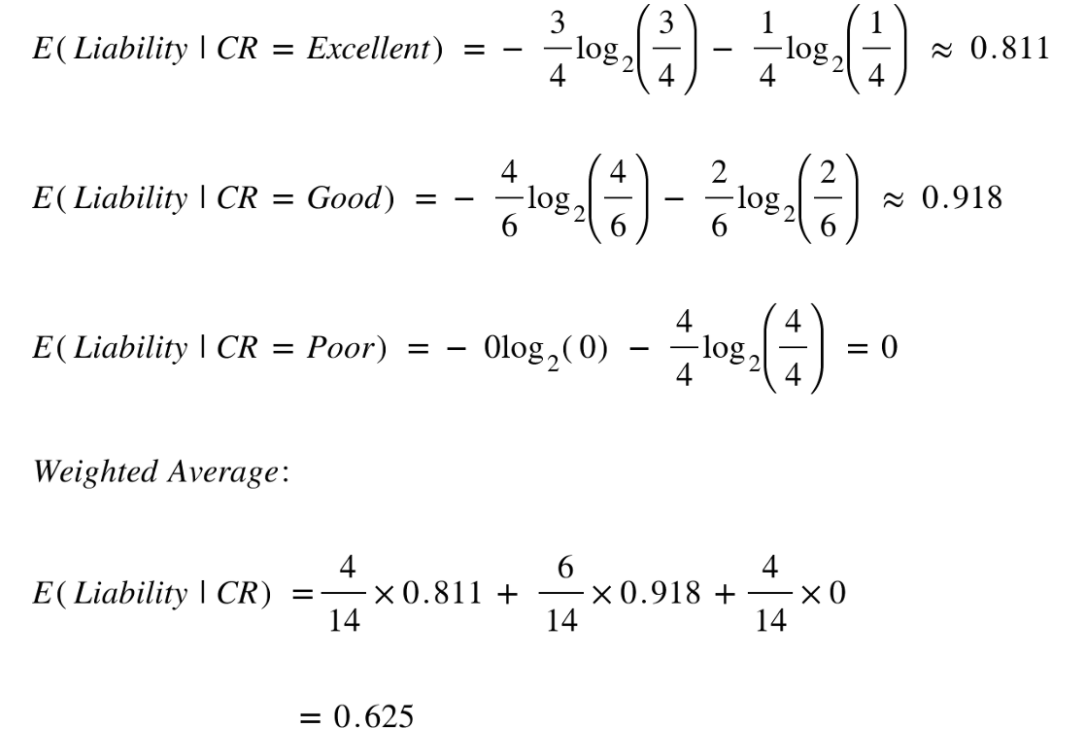
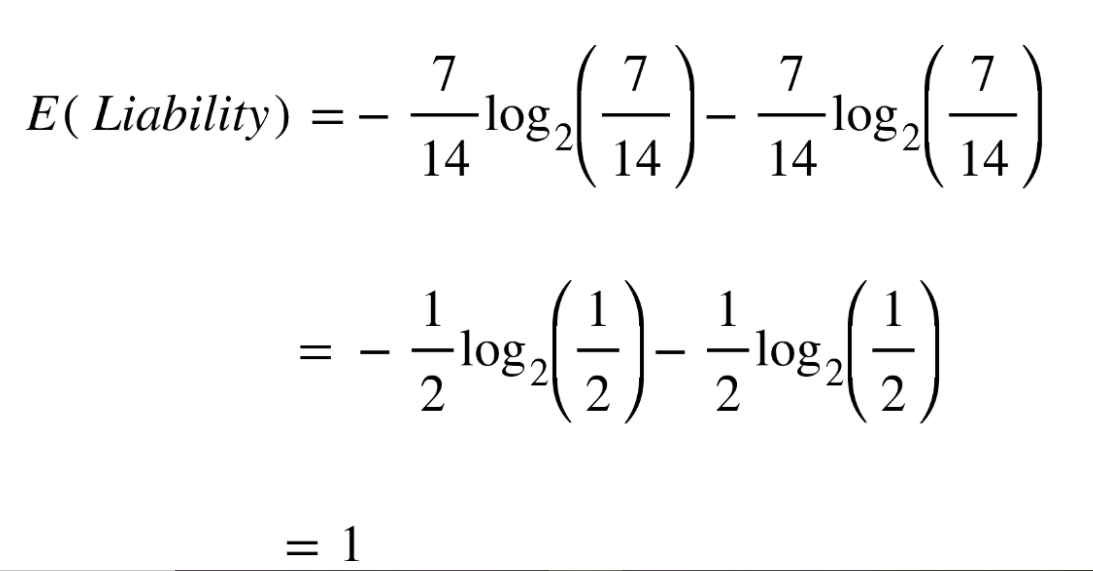
Where pi is the frequentist probability of an element/class in a data. Example:

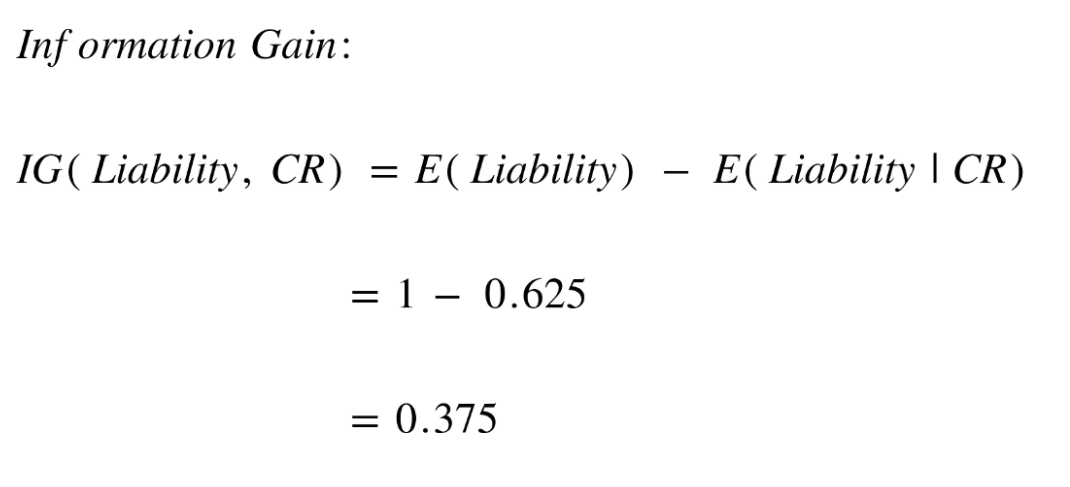


This is a higher value of entropy meaning a low level of purity or high levels of disorder. Entropy is the lowest when the set contains only one kind of elements meaning the set is pure and disorder is 0. Entropy is highest when the set contains equal instances of all the classes, extreme disorder because there is no majority. A high entropy indicates less predictive power.

**Information Gain:**

We need a metric to measure the reduction of this disorder in our target variable/class given additional information( features/independent variables) about it. This is where Information Gain comes in. 





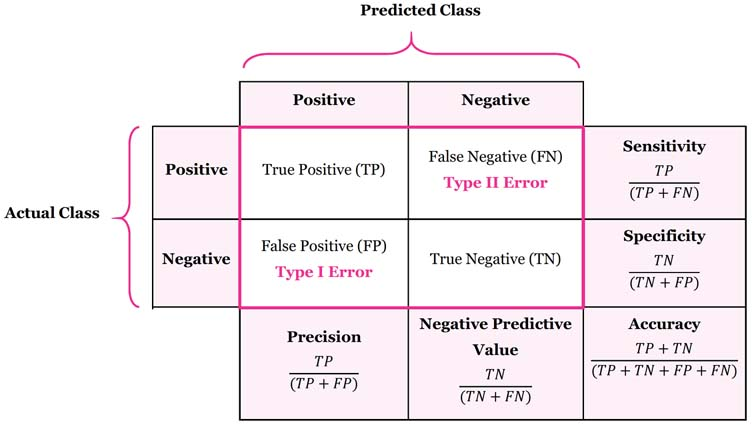
Knowing our credit rating decreased our uncertainty around the target variable liability.

**How do decision trees make decisions?**

Decision tree calculates the entropy of the parent node (or root node in case of the first split) and then calculates the entropy after making a split using a particular feature. It repeats this process for all the features and then calculates the information gain from all the features. The one feature which gives highest information gain or gives the lowest entropy after the split is used as the defining feature for the current split. Decision trees work to maximise the purity of the classes when making splits. If the entropy increases, another split is tried or the branch of the tree will stop i.e. the current tree has the lowest entropy.

**Confusion Matrix:**

A confusion matrix is a table often used to describe the performance of a classification model on a set of test data for which the true values are known.



This is a list of rates that are often computed from a confusion matrix for a binary classifier:

* **Accuracy**: Overall, how often is the classifier correct?
  + **(TP+TN)/total**
* **Misclassification Rate**: Overall, how often is it wrong?
  + **(FP+FN)/total** = 1 - Accuracy
* **True Positive Rate**: When it's actually yes, how often does it predict yes?
  + **TP/Actual yes** = Recall
* **False Positive Rate**: When it's actually no, how often does it predict yes?
  + **FP/actual no**
* **Precision**: When it predicts yes, how often is it correct?
  + **TP/predicted yes**
* **Prevalence**: How often does the yes condition actually occur in our sample?
  + **actual yes/total**

**Cross-Validation:**

Cross validation is a statistical method used to estimate the performance of ML models. It is used to protect against overfitting in a predictive model, particularly in a case where the amount of data may be limited.

What is wrong with testing the model on the training dataset?

If we do so, we assume that the training data represents all the possible scenarios of real-world and this will also be the case. Although, the training data is also real world data, it represents a small set of all the possible data points.

1. **Holdout method**: Usually the sie of the training data is set to more than twice the test data. The data is first shuffled randomly before splitting. Hence, we can never assure that train dataset is representative of entire dataset
2. **K-fold cross validation**: This method guarantees that the score of our model does not depend on the way the data has been picked for training and testing. Steps:
   1. Randomly split the dataset into k number of folds
   2. For each fold, build your model on k-1 folds, then test for the effectiveness of the model on kth fold
   3. The average of your k recorded accuracy is called the cross validation accuracy and will serve as the performance metric of the model
   4. Advantages:
      1. It validates the performance of the model on multiple folds
      2. It can balance out predicted features’ classes if you are dealing with unbalanced data
   5. Disadvantages:
      1. The algo has to rerun from scratch k times, which means it takes k times as much computation time to make an evaluation as the holdout method
3. **Stratified k-fold cross validation:** Stratified sampling is a sampling technique where the samples are selected in the same proportion as they exist in the population. Implementing the same concept in cross validation causes the training and test sets to have the same proportion of the feature of interest as in the original dataset. Before the implementation, the data is stratified to ensure that each fold is a good representation of the whole dataset.

**What is the difference between KNN and K-Means Clustering?**

Being a **supervised classification algorithm**, K-nearest neighbors needs labelled data to train on. With the given data, KNN can classify new, unlabelled data by analysis of the k number of the nearest data points. Thus, the variable k is considered to be a parameter that will be established by the machine learning engineer. Thus, KNN needs training data to make predictions.

On the other hand, K-means clustering represents an **unsupervised clustering algorithm** that needs unlabelled data to train. K-means clustering is able to gradually learn how to cluster the unlabelled points into groups by analysis of the mean distance of said points. In this case, the variable k depicts the number of clusters or different groups in which the data will be gathered. The algorithm functions by moving the data in such manner that error function is minimized.

K-nearest neighbor algorithm is mainly used for **classification and regression** of given data when the attribute is already known. This stands as a major difference between the two algorithms due to the fact that the K-means clustering algorithm is popularly used for scenarios such as getting deeper understanding of demographics, social media trends, marketing strategies evolution and so on.

Hence, KNN and k-means clustering are important algorithms when it comes to machine learning.

But each algorithm Is meant to deal with different problems and provide different meaning of what the variable k stands for.

* KNN represents a supervised classification algorithm that will give new data points accordingly to the k number or the closest data points,
* while k-means clustering is an unsupervised clustering algorithm that gathers and groups data into k number of clusters.

Anyhow, there is a common aspect which can be encountered in both algorithms: KNN and k-means clustering represent distance-based algorithms that rely on a metric.

**Why do we perform undersampling only after splitting the data?**

If precision is poor and recall is good which indicates that your model is good at predicting fraud class as fraud but the model is confusing for non fraud class, most of the times it is predicting non fraud class as fraud (if you set 0 for majority class 1 for minority class). This means that you have to try on reducing the undersampling rate for the majority class.

Typically undersampling/oversampling will be done on train split only, this is the correct approach. However,

1. Before undersampling, make sure your train split has class distribution as same as the main dataset. (Use stratified while splitting)
2. If you are using the python *sklearn* library for training your classifier set the parameter *class\_weight='balanced'*.

For example:

***from sklearn.linear\_model import LogisticRegression Lr = LogisticRegression(class\_weight='balanced')***

1. Try with different algorithms with different hyperparameters, if the model is underfitting then consider choosing XGboost.

If you do undersample before splitting then the test split distribution may not replicate the distribution of real-world data. Hence people typically avoid sampling before splitting.

**How did you conclude that Logistic regression gave the best results?**

**Could you guess what the features could have been on which the fraud detection is dependent?**