**Kaggle: Personalized Medicine: Redefining Cancer Treatment**

**Problem Statement:**

Clinical pathologist manually reviews the clinical literature and distinguishing the mutations that contribute to tumour growth (drivers) from the neutral mutations (passengers). After identification of Cancer specific genetic variations need to classify them in to 1 to 9. All this is a laborious and time-consuming process.

Here we need to apply Machine Learning to review the Text Literature, Gene, Variation and classify them into respective Cancer class automatically.

**Data Review:**

1. Missing values in Text
2. Text data and variant data was provided in two different files. So Need to merge based on ID
3. Refine the Text and remove the stopwords

**Exploratory Data Analysis ( EDA):**

1. All the data is categorical
2. Target Analysis: More number of variations belongs to the class 7 and least to the class 8
3. Most of the variations present in the following genes BRCA1, TP53, EGFR, PTEN, BRCA2
4. Top variation types are Truncating Mutations, Deletion, Amplification, Fusions, Overexpression
5. Same type of variation is present in different genes and different classes.
6. Same Text is present for Different Genes,Variations and Class
7. EGFR Gene is present in 8 classes

#### **Data Pre-processing:**

* To Handle whitespaces in Gene, Variation Columns
* Impute missing values

**Machine Learning :**

**Train\_test\_split:**

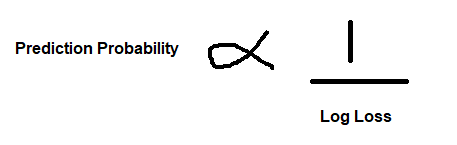
* Split the data to Train, test, CV @60:20:20 ratiowith stratify the target
* Observed the same distribution of target variable in y\_train, y\_test, y\_cv

#### **Build a Random model:**

Building a Random model to evaluate the performance with Machine learning model. ML model should be better than the random model. Evaluation Metric is Log Loss

**Log Loss :**

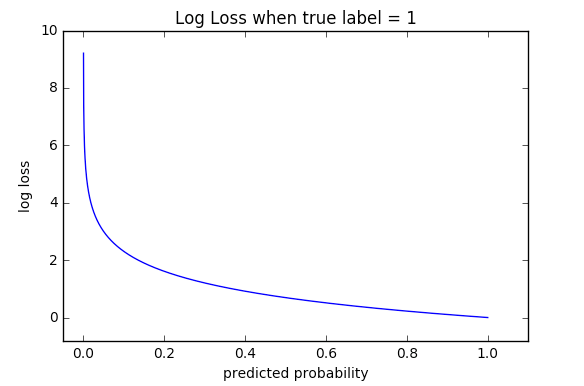
Logarithmic loss / binary cross-entropy is a loss function for the Prediction probability between 0 and 1.



A perfect model would have a log loss of 0. Log loss increases as the predicted probability diverges from the actual label.

Example:

Actual observation label is 1 but predicting a probability as 0.2 which is a bad result. So the log loss will be very high.



## Log Loss vs Accuracy

* Accuracy is the count of predictions where your predicted value equals the actual value.
* But **Log Loss assign a probability to each class** rather than simply yielding the most likely class.

**Feature Engineering:**

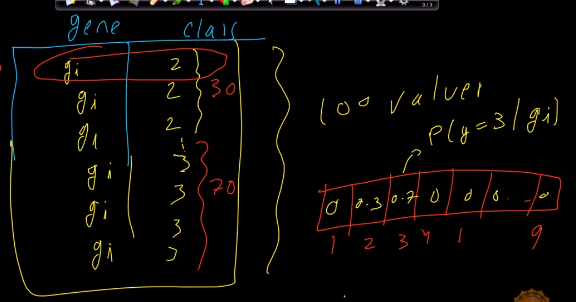
Transforming features, suitable for modelling;

1. One Hot encoding – pd.get\_dummies
2. Response Encoding: Probability of particular gene in Respective class.

One Hot encoding creates separate columns for all the items in the features i.e if there are 100 genes then it will create 100 columns which causes high dimensionality. But the Response Encoding create columns only wrt to number of items in the Target. I.e Class has 9 items. So Response encoding create 9 columns and impute Probability of particular gene in Respective class. This helps to maintain Low dimensionality.

Response Encoding Formula :

**Number of times that gene in specific class + (alpha\*10) / total numb of times of gene in all classes + (alpha \*90)**



**Feature Selection:**

1. **Create a alpha list -** [1e-05, 0.0001, 0.001, 0.01, 0.1, 1]
2. **Run the** stochastic gradient descent (SGD) for the each feature vs target
3. Calculate the log\_loss and store it in the list
4. Identify the best alpha and run again the SGD with it.
5. If the loss\_loss is less than random model then the feature is important.
6. Also check coverage of a feature in train & test. i.e Number of test genes in train data

should be high.

**ML Model:**

1. Combine all the encoded features to X\_train by hstack ( horizontal stack)
2. Trained with following ML models

'Naive', 'knn', 'logistic\_reg', 'svm', 'Random forest'

1. Method:

* model.fit(X\_train, y\_train)
* new\_model = CalibratedClassifierCV(model, method="sigmoid")
* new\_model.fit(X\_train, y\_train)
* y\_cv\_pred = new\_model.predict\_proba(cv\_x)
* log\_loss(cv\_y, y\_cv\_pred))
* y\_test\_pred = new\_model.predict\_proba(test\_x)
* log\_loss(y\_test, y\_test\_pred))

**Results of all ML models:**



**From all the above Machine Learning Results Logistic Regression given the best Results of Log\_Loss 1.257**