Classification loss functions:

1. Cross Entropy Loss/Negative Log Likelihood and Log loss seem to be same.
2. Hinge Loss/Multi class SVM Loss.

Logarithmic loss minimization leads to well-behaved probabilistic outputs.

Hinge loss leads to some (not guaranteed) sparsity on the dual, but it doesn't help at probability estimation. Instead, it punishes misclassifications (that's why it's so useful to determine margins): diminishing hinge-loss comes with diminishing across margin misclassifications.

So, summarizing:

* Logarithmic loss leads to better probability estimation at the cost of accuracy
* Hinge loss leads to better accuracy and some sparsity at the cost of much less sensitivity regarding probabilities

<https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23>

GB

<https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/>

<https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d>

ROC,AUC

<https://acutecaretesting.org/en/articles/roc-curves-what-are-they-and-how-are-they-used>

L1 and L2:

<https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/>

Decision trees split algos (ID3, Gini, Chi-sq, Reduction in Variance):

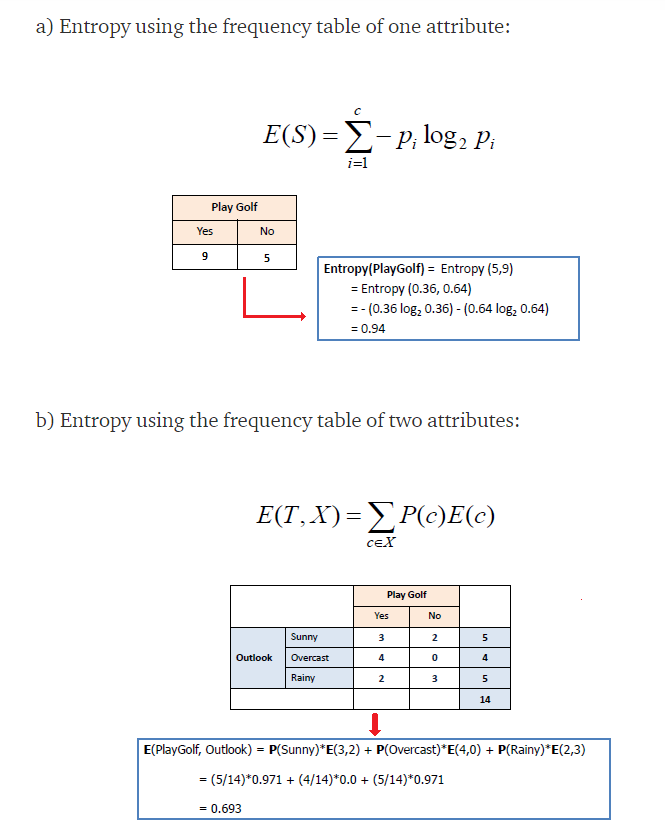
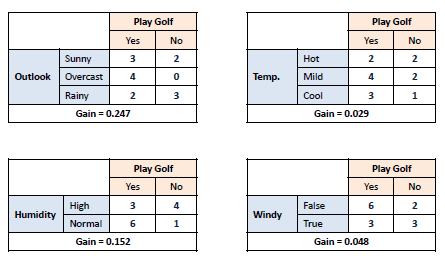
<https://medium.com/@rishabhjain_22692/decision-trees-it-begins-here-93ff54ef134>

=============================================================================

<https://www.linkedin.com/posts/srivatsan-srinivasan-b8131b_datascience-end2endds-activity-6567018330446794752-v1BX>

1. recommender systems metric on ranking
2. decision trees gini index and other metrics for splitting a node
3. assumptions of a logistic regression model
4. decision tree on a numeric variable
5. z-test vs t-test
6. poisson distribution
7. Feature selection
8. Dimensionality reduction
9. CNN
10. Optimization techniques
11. Loss Functions
12. MLE
13. Optimization used for XGBoost and random forest as well.
14. How is AUC curve formed?
15. XGBoost model working.
16. L1 and L2 regularization working.
17. What encoding can be used for a lot of values in an important categorical variable?
18. Softmax and sigmoid?

Decision trees split algos:

1. **ID3**: Top-down greedy search algo through space of possible branches. No backtracking. ID3 uses Entropy and Information gain to construct decision tree.
   1. Entropy (information loss): The lesser, the better. Inverse of Information gain. Decision tree partitions the data into subsets that contain instances with similar values (homogeneous). ID3 uses entropy to calculate homogeneity of a sample. If the sample is completely homogeneous (the samples are correctly put into subsets, each of similar instances), the entropy is zero. Reverse of it makes entropy as one.
   2. Information gain: The higher the better. Info gain is decrease in the entropy after a dataset is split on an attribute. The attribute with highest info gain is considered as the preferred node for decision tree. Steps to calculate information gain:
      1. Calculate entropy of the target variable.
      2. Calculate entropy of each of the possible branches (variables). Information gain is entropy of current – entropy of the possible branch. Node with highest info gain is considered.
      3. Repeat i and ii. So info gain is calculated over entropy.
      4. A branch with entropy = 0 is leaf node and entropy > 0 needs further splitting.

WORD REPRESENTATION:

1 hot encoding doesn’t give any relation between two relevant words.

Eg: I would like to have a glass of orange juice.

I would like to have a glass of apple . (here the one hot representation doesn’t show any relation between apple and orange, to say that juice will come after apple, rather than any other word)

So we represent each word with high dimensional feature vectors.

Representing these n dimensional vectors to 2D to visualize is done by t-SNE.

The n dimensional feature vectors are called embeddings. Thus we are visualizing the word embeddings.

USING WORD EMBEDDINGS:

1B to 100B corpus of unlabeled words can be used to train a model which can create relations between entities (words). This relation can be used to label the words and then we can do transfer learning i.e. use these labelled words for NER task using a BRNN. (The representation vectors are reduced to labelled data using t-SNE).

PROPERTIES OF WORD EMBEDDINGS:

Suppose if we are supposed to find ANALOGUES like:

Man -> Woman as King -> ? (It is supposed to be Queen)

So we take eman - ewoman almost similar to eking - e?

Thus e? =-eman + ewoman - eking

The similarity function most commonly used is cosine similarity.

We can also do a measure of dissimilarity using Euclidean distance.

Whichever value is similar, to the vector in the left is the correct entity.

EMBEDDING MATRIX:

E . Oj = ej

LEARNING WORD EMBEDDINGS:

Neural Language Model: i/p (n word window X features) -> neural n/w (with wts and biases) -> softmax (with wt and bias)

The neural N/W learns on the i/p representation vectors and predicts the next word.

Context is the n word window. This can be n prev and n next word/s also.

WORD2VEC:

Skipgrams: Randomly chosen context and target words with +- n words window. This acts as a training set for our NN.

Computation is an issue for softmax classifier as it traverses through every word in the vocab. This can be solved using hierarchical softmax classifier. Frequent words at the upper level of the tree.

As in the training data, there can be more of the frequent words the context, so we do sampling (not random) in such a way that the freq and infreq words in context are balanced. This is a word2vec skipgram model