H1B VISA PETITIONS

RESEARCH ON LEARNING ALGORITHMS

From-

ANISH SINGH WALIA-15BIT0116

ABSTRACT:

This paper is a research on the 2 major types of *supervised learning* Algorithms which are *K-NN* , *Boosting* and their different Variants and extensions. K-NN is also called a class of *Lazy-learners* , which means that it starts learning only when a Test Point is given to it , only then it looks up the complete training dataset for the nearest *located neighbors* to the test point '**q**' using a distance metric which calculates the distance of between the test point and the training points and classifies the Test point based on the **Average** of the *Y values* of the closest K points for a Regression problem and for a Classification problem labels the point **'q'** based on the **votes**. K-NN is considered a very Stable and a simple learner learner . It is said that K-NN is used 1/3 of the times. Secondly , We researched on Boosting which is a Ensembling technique in which we train various ***Weak Learners*** on subsets or distributions of *Hard examples* and at the end combine those Learner's output Hypothesis to generate a *Strong, stable* and *accurate* Final Hypothesis by doing weighted averaging. What Boosting ends up doing is converting those *Weak* Learners to a *Strong* and accurate learner. Then we have the 2 extensions and different variants of K-NN and Boosting called *Locally weighted Learning* and *AdaBoost*.

Here in this paper we have compared the **ROC** (receiver operating curves for each learners for a given problem which we tried to solve for a given dataset). *ROC* curves are the plot of True Positive Rates(Correct classification Rate) Vs the False Positive Rates(Misclassification Rates)

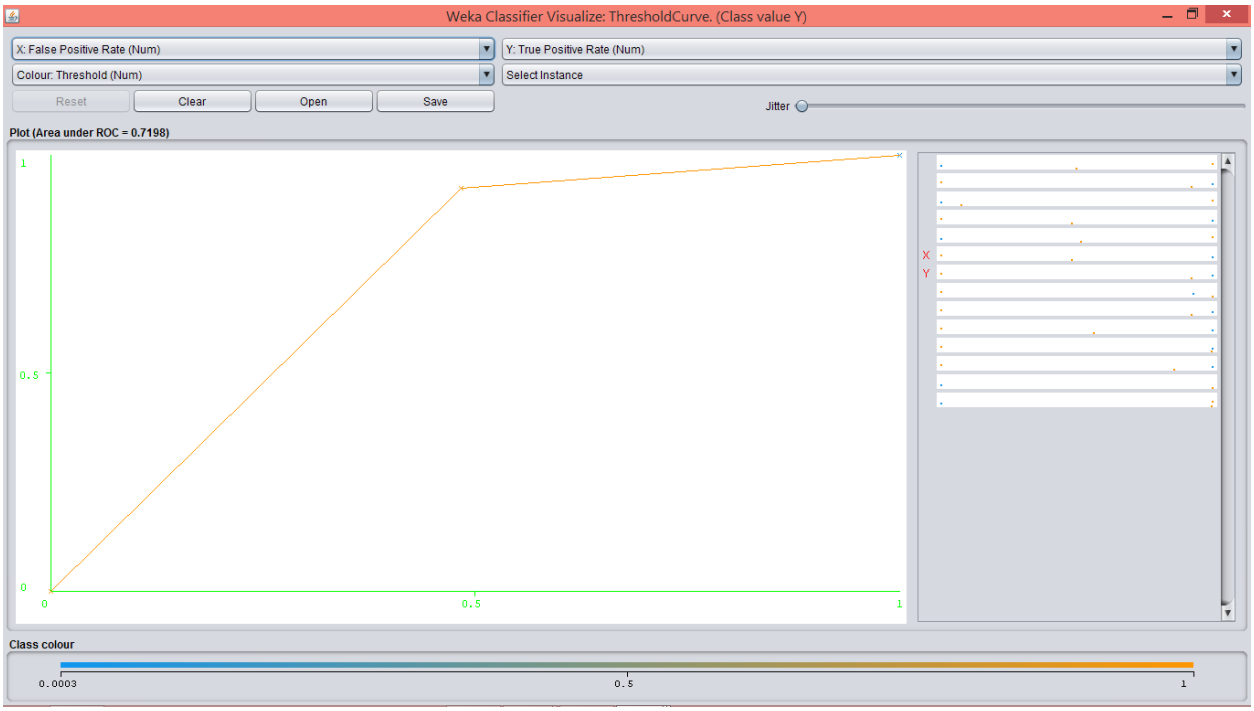
INTRODUCTION

Data Mining is a process of extraction of knowledge and hidden patterns from large Data sets using various Statistical Techniques and Artificial Intelligence techniques. *Machine learning* is a sub branch of AI which is used to train computers and machines using Large data sets so that they can be able to predict the future outcomes by learning and analyzing from that data and by their experience. Machine Learning is being used everywhere . From Amazon's Recommendations Systems to Google Translator , Fraud Detection , Email Classification(Spam),Text Classification , Sentimental Analysis , Computer Vision ,Image processing, Natural Language Processing, Image Captioning etc . *Machine Learning* also plays a major role in Business Analytics. It can be used to predict the Stock Prices , Market Basket analysis , Recommendation System , Decision Making etc are just some of the small examples of applications of ML in real life. There are 3 major types of Learnings- 1) *Supervised Learning 2) Unsupervised Learning 3) Reinforcement Learning.* **Supervised Learning** or *Instance* Based Learning is a type of learning in which we train a Algorithm using **labelled** training data , a data set which has both set of inputs(Xi) and Target(Yi) values- {(Xi,Yi)}. In **unsupervised learning** we train the algorithm on unlabeled data . A data set which has only a set of Inputs(Xi).

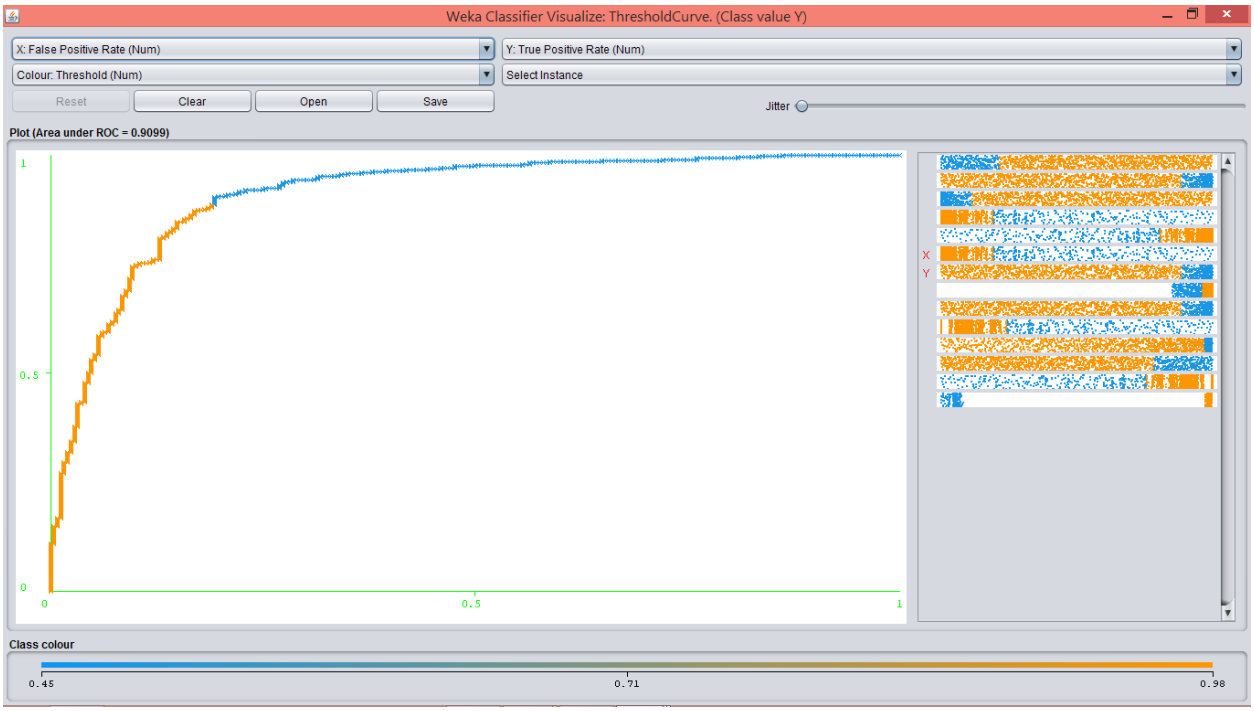
What actually is done is the learner analyzes and learns from the training examples and then generates a inferring function which relates X and Y variables , such that we are able to map X --> Y. And using this inferring function which is then optimized to have least error and inaccuracies is given a **Test Set** which was not given to it during training and is completely unseen and random . Then the Model's accuracy is tested on the *Test set* and his ability to *Generalize* to unseen examples. The biggest aim of Machine Learning is to **'Generalize**'.

Various Machine Learning Algorithm implemented in Our Project-

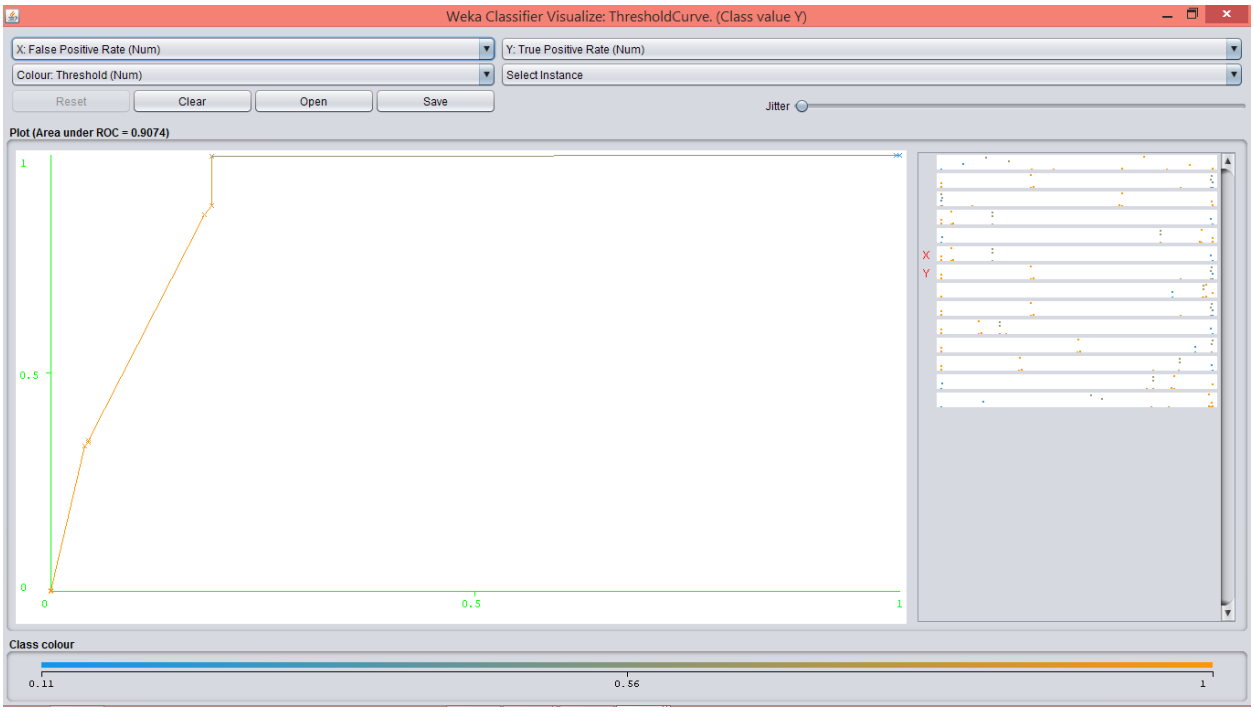
1)**K-NN** : Also known as *K-Nearest Neighbor* Algorithm. It comes under the category of Lazy learners, which actually start off learning only when given a Query point (test point). What we do is we store the training data set into a database and when we have a test point , then we actually go through the complete training data to find the K – nearest neighbors to that query point 'Q' using a Distance Metric 'd()' and make a set of those closest situated training examples and do averaging of the outputs of the K-NN training examples if a regression Problem or do voting for the class labels and classify the Q point to Class having highest vote. *K-NN is just a Generalization on Linear Models* to learn **non-linear interactions** in data. It is a *stable* learner and a *simple* one. The *Inductive Bias* for this learner is that all the Local points or points which are closer to each other in a plane are *similar* to each other and have same *structural similarities* and behavior. Hence the *distance metric* 'd()' used to compute the *nearest situated points* to Q is just a *notion of similarity*. The only drawback in *K-NN* is that *high noise* and *outliers* can decrease the degrade the performance of the K-NN algorithm , hence it is necessary before doing predictive modelling using K-NN , we should properly *Normalize* or *Standardize* the data set to remove outliers and Noise and do some pre-processing. It is said that K-NN algorithm is used 1/3 of the times.



2) Locally Weighted Learning – It is also an extension of K-NN algorithm. K-NN has a special power .When we find the *local* nearest neighbors in K-NN , we can actually fit a Model to those local points and gain more and learn more information from those locally fitted points. E.g – is Locally weighted regression. We can actually apply any sort of learning algorithm on those local points and learn some new functional mapping and extra information. We can apply regression , Classifier , SVM , Decision trees etc . This method is also assumed to improve the accuracy of the K-NN model.



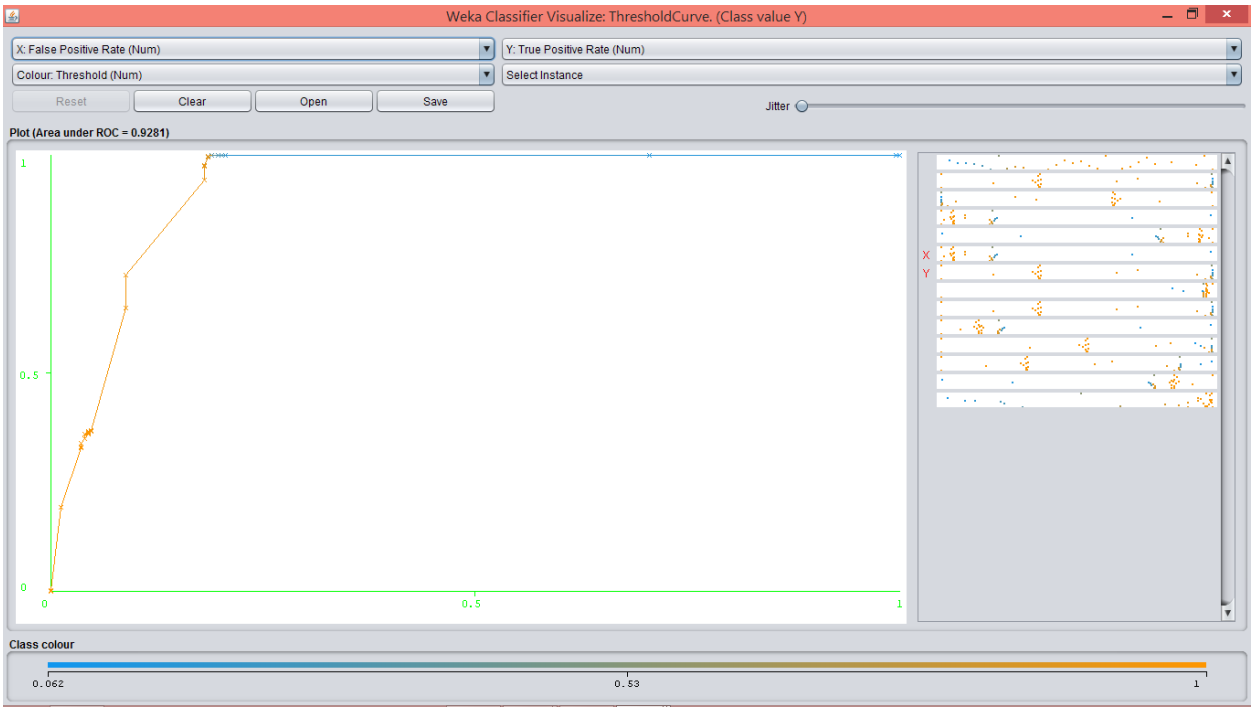
3)**Boosting** – boosting is an ensembling technique In which we train multiple learners on *hard* distributions of data generated from all the *misclassified* examples from the previous learner. What we do is we start off with random distribution of data and apply a **Weak Learner** to it. A *Weak learner* is a learner which always learns something and does better then chance i.e it has error less than ½ or 50% . Hence we generated new distributions of data from **previously *misclassified(Hard*) examples** of last Model , and feed those *hard* examples to the next new weak learner , and eventually we end up having lesser error in each iteration .It is because the algorithm when generating a new **distribution** tends to put more **weight** on the examples which were *misclassified*(incorrectly classified by previous learner's hypothesis). In the same way we apply various Weak learners to the *hard* examples and at the end we combine and do a *weighted average* of the output (hypothesis) of each learner to get a **Final Hypothesis** which we assume would be the best and has least error and most accuracy. Now Boosting is consider a very nice algorithm is terms of improving prediction accuracy of the Models on out of sample datasets . It is generally said that *Boosting* normally doesn't **Overfit** . The exceptions being when we use **A-NN(Artificial Neural networks)** as a learner. Because the whole concept in boosting is to fit hard examples of data on a Weak learner and an *A-NN* Model is definitely considered one of the most powerful , strong ,complex and complicated amongst all of the other learners . Because A-NN has the power to learn almost anything from discontinuous to continuous functional mappings and linear to *Non-linear* complex functions. Due to this A-NN can easily make sense of something complicated and highly complex data sets and learn from the complex examples. Hence applying **boosting** on **A-NN** will surely tend to Overfit.



Boosting converts a Weak learner to a Strong Learner.

4**) AdaBoost**: It is one of the Boosting algorithms. AdaBoost seeks to find such a final classifier with **high margin** on all examples by combining many base classifiers(*Weak learners*). A nice property of **AdaBoost** is its ability to identify **outliers**, i.e., examples that are either mislabeled in the training data, or that are inherently ambiguous and *hard* to **categorize or classify** . Because AdaBoost puts more **weights** on the *hardest* examples, the examples with the highest weight often turn out to be *outliers*. *Hardest* examples means the training examples which were misclassified by the pervious learner. Hence we can assume that the misclassified examples from the previous learner tend to be some noise or some outliers.

Another important thing to take care is that in AdaBoost we should not use a **strong complex learner** as a *Base learner* to apply Boosting on it because as we have mentioned earlier Boosting tends to **Overfit** on Complex , Strong & complicated learners such as A-NN. **Weak learner** should not be too complex - to avoid overfitting.



As we can see the **AUC** value for **AdaBoost** algorithm when implemented has the highest area value , which indicates that it does the best in *predicting* correctly and highest **Classification accuracy of 97.125 % on Test Data Set**.

LITERATURE SURVEY :

One Fascinating issue covered in the reaserch is that we found out the Boosting whantever flavor , seems resitant to **Overfitting**.This is because in each iteration when we generate new Distribution by putting more weight on the *Hard* examples – which are near the **Decision Boundary** , they tend to move away from the **decision boundary** towards their respected class label distribution , which actually tends to *maximize the Margin* of the classes from the decision boundary(a Line which saperates the Data)**. And what do we know about maximizing margins- They reduce Overfitting as studied in SVM. Also Classifiers are hurt less by Overfitting than other functional estimators.**[6]

CONCLUSION:

In this overview, we have seen that there have emerged a great many views or interpretations of Ada-Boost. First and foremost, Ada-Boost is a genuine boosting algorithm: given access to a true weak learning algorithm that always performs a little bit better than random guessing on every distribution over the training set, we can prove arbitrarily good bounds on the training error and generalization error of Ada-Boost. For our problem the best results were obtained when we applied Boosting to it , and we had a amazing Generalization accuracy of **97.125%** when we applied AdaBoost algorithm and when we applied **Logit-Boost** and simple extension of Ada-Boost for Binary Classification problems we got the highest accuracy of **97.125%.** All of these connections and interpretations have greatly enhanced our understanding of boosting and contributed to its extension in ever more practical directions, such as to logistic regression and other loss-minimization problems, to multiclass problems, to incorporate regularization and to allow the integration of prior background knowledge.

References:

[1] Steven Abney, Robert E. Schapire, and Yoram Singer. Boosting applied to tagging and PP attachment. In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, 1999.

[2] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. Journal of Machine Learning Research, 1:113–141, 2000.

[3] Peter L. Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. IEEE Transactions on Information Theory, 44(2):525–536, March 1998.

[4] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning, 36(1/2):105–139, 1999.

[5] Eric B. Baum and David Haussler. What size net gives valid generalization? Neural Computation, 1(1):151–160, 1989. [6] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. Journal of the Association for Computing Machinery, 36(4):929–965, October 1989.

[6] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: A statistical view of boosting. The Annals of Statistics, 38(2):337–374, April 2000.

[7] StefanoMerler, Cesare Furlanello, Barbara Larcher, and Andrea Sboner. Tuning costsensitive boosting and its application to melanoma diagnosis. In *Multiple Classifier Systems: Proceedings of the 2nd International Workshop*, pages 32–42, 2001.

[8] C. J. Merz and P. M. Murphy. UCI repository of machine learning databases, 1999. www.ics.uci.edu/mlearn/MLRepository.html.

[9] Pedro J. Moreno, Beth Logan, and Bhiksha Raj. A boosting approach for confidence scoring. In *Proceedings of the 7th European Conference on Speech Communication and Technology*, 2001.

[10] Michael C.Mozer, RichardWolniewicz, David B. Grimes, Eric Johnson, and Howard Kaushansky. Predicting subscriber dissatisfaction and improving retention in the wireless telecommunications industry. *IEEE Transactions on Neural Networks*, 11:690–696, 2000.