Applied Machine Learning - Assignment 3

Anurag Sethi

Nov 04, 2019

1 Introduction

This project focuses on the application of two machine learning classification techniques: artificial neural networks and K nearest neighbours. While algorithm application is a small but critical part of the project **the foremost motivation of the project is to get learning outcomes by varying control parameters of the algorithms and their effect on performance metric**. In first section, we do precrocessing and feature engineering, section 2 and the following subsections are the implementation of the two algorithms on the *UCI Appliance Energy Consumption* dataset along with cross validation implementation with each alogorithm. In the last subsection is perhaps the most critical one with model performance comparison. In section 3, same series of algorithm implementations and model performance comparison are done for *Kickstarter Campaign Success Prediction dataset*. In last section, section 4, the project concluding remarks which summarize the key learning takeaways of the project are presented. **Note: Section 1 for this assignment is similar to assignment two, as we are using the same two datasets, hence they will have same pre-processing and feature engineering steps. Also, for kickstarter campaign prediction a smaller set of 30k data points is used due to computational constraints for implementation of Artificial Neural Networks**

1.1 Datasets

The project makes use of two very interesting datasets **Appliance Energy Consumption Dataset** hosted at *UCI machine learning repository* a popular archive portal for open datasets for academic use. We create the new variable 'spike' where the recorded consumption exceeds 150 Kwh. By definition, our dependent variable indicates to the user potential spike in the electricity usage, such a working model **helps the users to anticipate spikes and regulate the appliance usage to reduce the energy consumption and subsequently reduce their carbon footprint**. There are multiple factors associated with selecting this particular dataset over others; from business point of view it is an interesting problem, crowdfunding is a recent phenomenon and it would be great insight if we know the factors that lead to success in crowdfunding campaigns. There can be many lessons for a person or a company aiming to launch their campaign, that could maximize their chances of raising money. Furthermore, this data-set had ample scope of *feature engineering*, which is always a great exercise, a model built with rudimentary parameter optimization with intelligent feature engineering can often beat a model built with advanced parameter optimization on raw features.

1.1.1 Data Preprocessing - Feature Engineering

UCI Appliance Energy Consumption : The dataset has 29 features to start with, there is dependent variable Appliances showing the energy consumption by appliances of the house. The other features can be broadly divided into four further categories, Weather Related Features,Humidity related features,Temperature Related Features: each category capturing the various entities at different parts of the house, also there are features like light, date, rv1 and rv2 which are put together as miscellaneous. In the pre-processing step numeric features are scaled using the z-score, implemented using the standard scaler operand inbuilt in python.

Engineered Variable Name	Description
Month	Descriptor labelled 1 to 5 representing month
Is Weekend	Flag if the day is a weekend or weekday
Time of Day	Section of the Day eg. night time, evening time, work hours etc.

We engineer new feature from date variable, which are relevant based on the understanding of the context. There are listed in the table below. The rationale behind this is that intuitively appliance consumption is affected by seasons (captured by month), day of week (captured by isweekend), and section of the day (captured by time of the

day). For our usage in this project, we selected set of 21 features (reduced after pairwise correlation test), In the logistic regression implementation in assignment one, we noted this set of variables gave the best classification result, though this doesn't mean that this set is the best set of variables to go for svm, tree based implementations but is a good starting point. Final feature set used is lights, RH1, T2, RH2, T3, T4, RH6, T7, RH7, T8, RH8, RH9, Tout, Pressmmhg, RHout, Windspeed, Tdewpoint, rv2, month, is-weekend,time-of-day-encoded.

The dependent variable is defined as *spike alert*; here category is defined as 1 when the consumption for the hour exceeds 150 kwh. In the dataset, the class split at this mark is 88.5 and 11.5 percent, which is a reasonable split from the prospective of modelling ease.

Kickstarter Campaign Success Prediction: The original set of raw variables can be obtained from here (https://www.kaggle.com/k projects). Over these features, we have engineering a set of new features, these help us inject some domain knowledge in our prediction.

Engineered Variable Name	Description
Duration	Descriptor labelled 1 to 5 representing month
Is Weekend	Flag if the day is a weekend or weekday
Launched year, quarter	Year and seasonality information derived from date of launch
average amount per backer	Average amount each patron has given till now
goal level	Money campaign is targetting divided by mean goal of projects in the category
competition quotient	Total same category projects lanuched in the same month with given campaign
syllable count	Number of syllables in the campaign name

The categorical columns are converted using one hot encoding and we get 42 features in our set. To reduce the dimensionality furthur, univariate analysis of variance is performed with the dependent variable and 14 features are retained for modelling after feature reduction. Energy data set is split into 70-30 train test split and kickstarter in 80-20 spilt (since we have more data points here 20 percent test size is a large enough sample to get meaningful out of sample results).

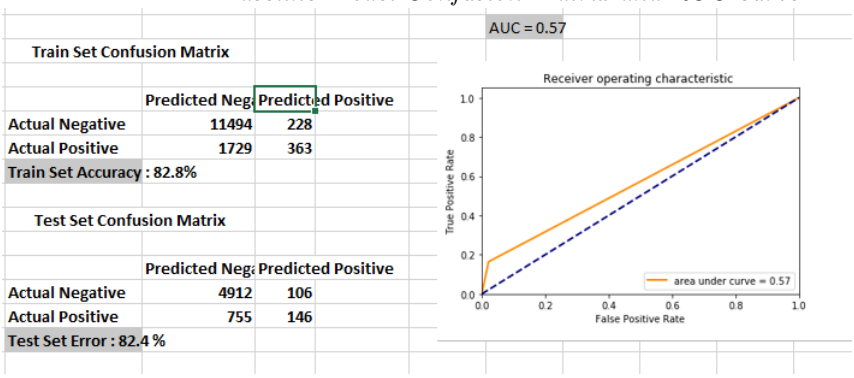
2 Appliance Energy Consumption

In the subsequent subsections, we deeply explore the two classification techniques and predict spike in appliance energy usage. We see we have 15.18 % of classes marked as positive (2995 out of 19735) where the usage exceeds 120Kwh. We vary the control parameters and see the effect on training and out of sample errors. We also present ROC curves and confusion matrices (corresponding test set accuracy) as our means of evaluating the multiple models build. **In the previous project, we came across parameters specific to SVM, Decision trees and ensemble models**. Parameters specific to ANN's that are used for tuning are not easily intuitive and often one relying on experimentation to select optimal set of features.

2.1 Appliance Energy Consumption: Artificial Neural Networks

In this section we study the application of **ANNs** and apply them to predict spike in the energy when it exceeds 120 KwH. For the baseline models, there is no intuition driving decision to select parameters values, we randomly select 3 hidden layers and 60 neurons in each layers to build the first cut of the model. Other params are also randomly initialized like loss function, activation function and converging function. We see the resultant model has decent train and test set accuracy but the AUC metric is quite poor. This is due to high false negatives predicted.

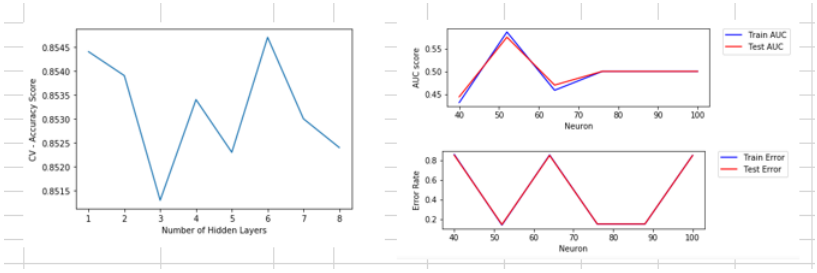
Baseline Model Confusion Matrix and ROC curve



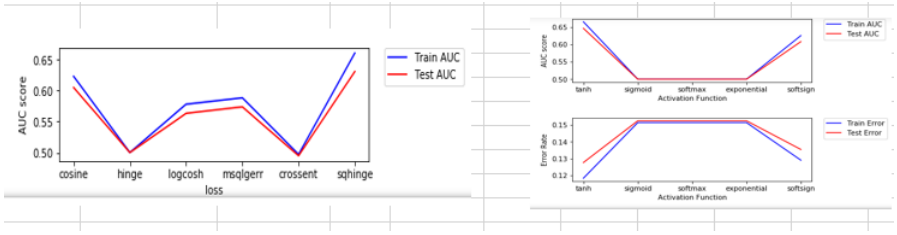
Baseline Model Conf Matrix and ROC curve

We consider AUC to be our key metric for spike prediction since we want to reduce false negatives, a typical model can go on and label all the test set data as negative and can achieve the accuracy of 85 percent given the class imbalance, we wish to avoid such a scenerio hence AUC is the desired metric for a more feasible and rational implementation. Below is the output of basline model and in subsequent steps we vary the parameters, number of layers, neuron count, loss function, activation function, epoch count, batch size, learn rate of the optimizer and try to improve the out of sample AUC, in the end we use grid search for cross validation and vary the parameters simultaneously instead of individually and attempt to arrive a more optimal version of the model with reduced out of sample AUC.

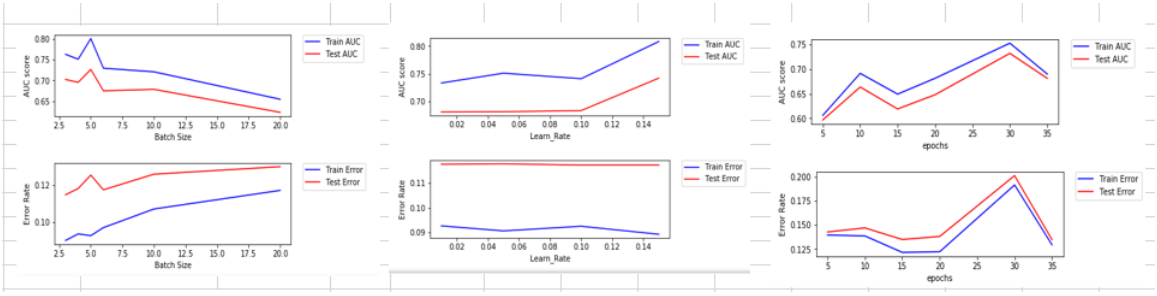
Below we can see that when we vary hidden layers and count of neurons there is no clear trend of convergence, both out of sample error and auc reach optimal at multiple intervals and there is an absence of trend which is observed. It is because Neural Networks topography isn't simplified easily and two different ANNs with dissimilar topography can give similar performance. This does give a starting point we can fixate the values for layer count and neuron count for now and optimize other params. Parameters such as bias initializer have been avoided since foremost other seven hyperparams when tuned and synced to form the neuron topography give good results which were unchanged when bias initializer over baseline was tried. Further here sequential model from keras is used to construct the neural network, means as a means of generation model will be created layer by layer and there will not be two nodes with overlapping shared layers. **Both these assumptions above are important because this level of simplicity suffices for the classification problems presented in the project.**



Learning curves for different params- Hidden Layer Count and Neurons per layer



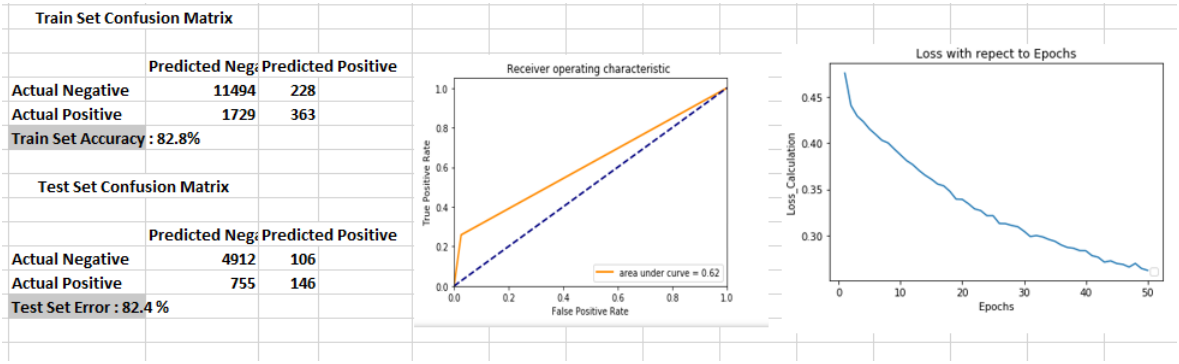
Learning curves for different params- Loss and Activation Function



Learning curves for different params- batch size. learn rate, epoch

Here we observe after tuning activation function and loss function, that **soft sign activation** gives maximum test set AUC and less out of sample error, and **squared hinge loss** gives the most auc for test set. These two functional forms (activation function and loss function) are most important of all parameters to optimize (more than topography features neural and hidden layer count and learn-ability features mentioned below since they lead to a higher increase in AUC when tuned on out of sample performance) **This is an important learning takeaway as is again highlighted in the last section of the project.**. Mathematically softsign is ratio of exponential for given sample probability divided by all other samples and Squared Hinged loss function is a maximum margin classifier (similar to usage in support vectors) it gives leverage in back-propagation error calculation.

The other three parameters (batch size, learn rate, epochs) are an indicator of the learn-ability' or 'bias variance trade off. The results are in line with our intuitive expectations, for epoch size 30 we get best results and then on increasing the epoch size there is a drop in out of sample auc (an indicator of over-fitting); similarly a high batch size means (more samples to work through before updating layers) model isn't trained enough and small batch size leads to better train and test set results (also the convergence time of the small batch size is more). In the learn rate of the optimizer however we find the optimal learn rate is a comparatively higher value of rate at 0.15.



Train Set Confusion Matrix

	Predicted Neg	Predicted Positive
Actual Negative	11494	228
Actual Positive	1729	363

Train Set Accuracy : 82.8%

Test Set Confusion Matrix

	Predicted Neg	Predicted Positive
Actual Negative	4912	106
Actual Positive	755	146

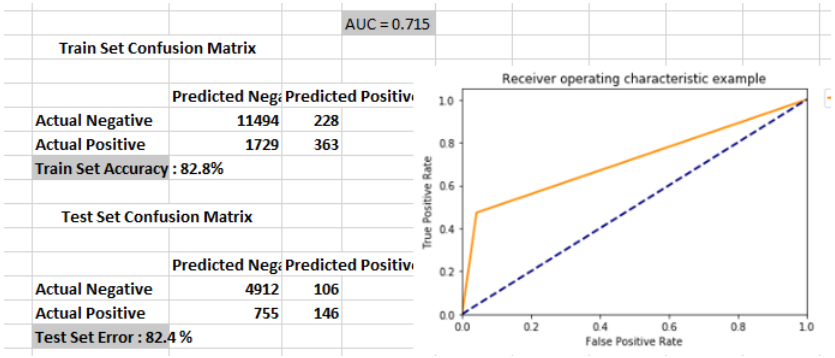
Test Set Error : 82.4 %

Hyper param tuned model, train test conf matrix and ROC, also: right cost plot wrt epoch

In the above figure, we apply CV grid, 3 fold cross validation (3 because of limited computational power) and do a grid search. We have a rough idea of the neighborhood the optimal values for the seven hyperparams lie, we initialize a grid search and arrive at the optimal model (refer code). Above are the results, there is a notable improvement in the test set AUC, where it rises to 0.62 from baseline 0.57. The test set accuracy is largely the same but there is a notable improvement in the reduction of false negatives (hence high AUC), means this version is a more learned version, it is not labelling more and more negative classes in a biased manner. This idea is been previously discussed at selection of the metric AUC. Correspondingly on the right we can see the trace of the loss function and its convergence upon increasing epochs of the optimal model.

2.2 Appliance Energy Consumption: K Nearest Neighbors

From parametric models like ANNs, we implement a much simpler instance based technique called k nearest neighbour, It is much easier and intuitive to visualize, it assigns a class label to that is most frequent among the k closest selection of points in the learning set to the data point on the prediction is made. Naturally as the name suggests the number of nearest neighbor selection is an important tuning parameter to optimize the implementation of this algorithm, along with metrics related to distance calculation and weights of the points. There are discussed in detail in the section.



AUC = 0.715

Train Set Confusion Matrix

	Predicted Neg	Predicted Positiv
Actual Negative	11494	228
Actual Positive	1729	363

Train Set Accuracy : 82.8%

Test Set Confusion Matrix

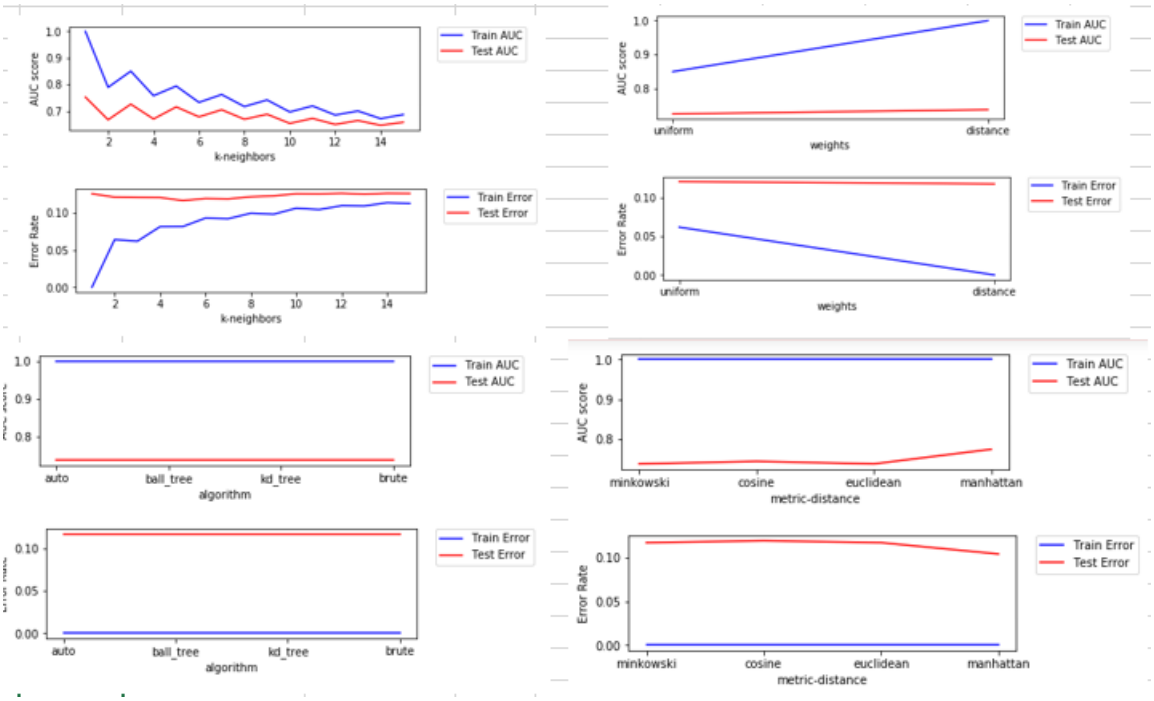
	Predicted Neg	Predicted Positiv
Actual Negative	4912	106
Actual Positive	755	146

Test Set Error : 82.4 %

Baseline Model KNN, train and test matrix, also ROC curve

The choice of nearest neighbors in the decision trees is directly linked to bias variance tradeoff, a small k leads to overfitting of the model whereas a large k leads to underfit, here we see an interesting observation, starting k=2 and increasing we see the value of AUC metric of out of sample test set maximising at 3 and post that the model overfit, same can be seen from the error rate as it drops from 2 to 3 and subsequentyl increases. Another observation is that the positive and negative increase toggles between odd and even values of k, odd values have higher AUC since it is binary classification and values of 3,5,7,9 etc make a distinction between the classes much straightforward, compared to even iterations where a clash has to be determined with some technique inducing a bias in the model and hence affecting the out of sample performance. The uniform and distance weighting distinction isn't there as it has no effect on the test set (we ignore improved training efficiency). Manhattan distance is a slightly better metric

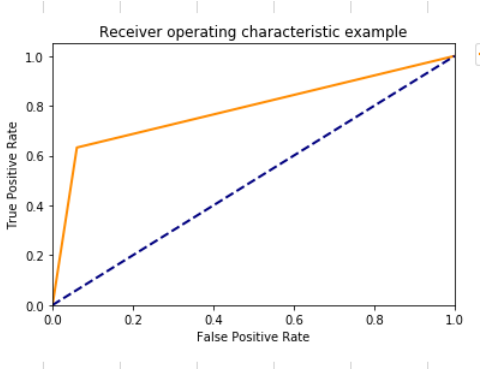
as evident from the out of sample validated results. Mathematically it is the sum of modulus differences between the individual coordinates, and performs slightly better out of sample, although the difference is insignificant



Learning curve for varying params: k, metric, algo, weights

Below we can see the accuracy on out of sample of the model which was obtained after hyper-parameter tuning using grid search, the changes are minimal from the first rudimentary implementation. **There is a slight improvement in the auc, it marginally increases, the test set error remains the same. One can make a speculative claim that the scope of accuracy and performance improvement using hyper parameter tuning is less in such an instance based algorithm.** It will be interesting to see the results of the other kickstarter dataset.

			AUC = 0.715
Train Set Confusion Matrix			
	Predicted Negativ	Predicted Positive	
Actual Ne	11122	0	
Actual Pos	0	2092	
Train Set Accuracy : 82.8%			
Test Set Confusion Matrix			
	Predicted Negativ	Predicted Positive	
Actual Ne	4714	304	
Actual Pos	332	571	
Test Set Error : 82.4 %			



Tuned Optimized knn train test matrix, right: ROC

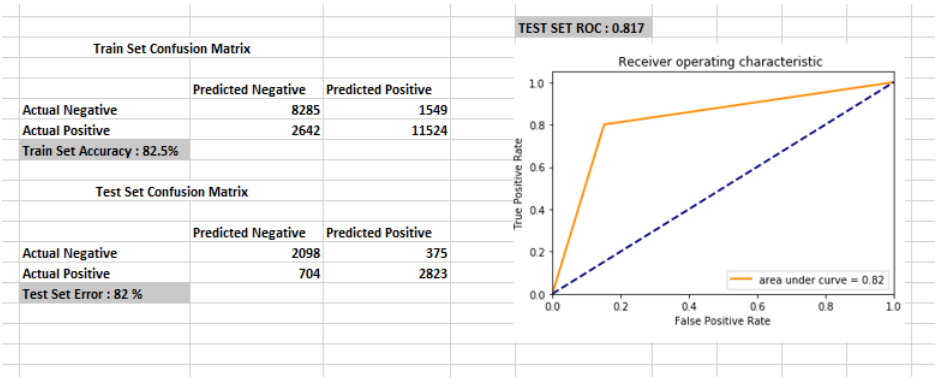
3 Kickstarter Campaign Success Prediction

In the next two subsections we implement Artificial Neural Networks and instance based k nearest neighbours and tune parameters to generate learning curves.

3.1 Kickstarter Campaign Success Prediction - Artificial Neural Networks

A lot of explanation of the explanations about the parameter definitions and intuitive meaning has been done in the preceding sections. Here we go straight to the learning curve implementations. The hidden layers show high variability in the errors of 3 fold cross validation, 6 layers have a much higher cv error accuracy than other layers and neuron at 40 per layer shown the right tradeoff between bias and variances. **To check the effect on other parameters we fixate these topographic structural parameters and see the**

marginal changes our metrics (auc and out of sample test error. The baseline model below has better auc to start with and make improvements.



Train Set Confusion Matrix

	Predicted Negative	Predicted Positive
Actual Negative	8285	1549
Actual Positive	2642	11524

Train Set Accuracy : 82.5%

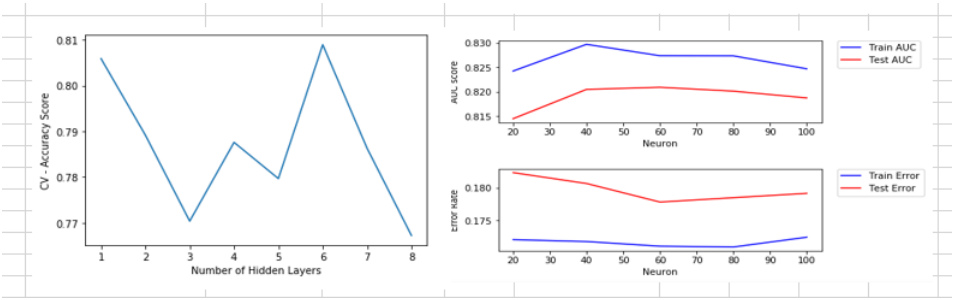
Test Set Confusion Matrix

	Predicted Negative	Predicted Positive
Actual Negative	2098	375
Actual Positive	704	2823

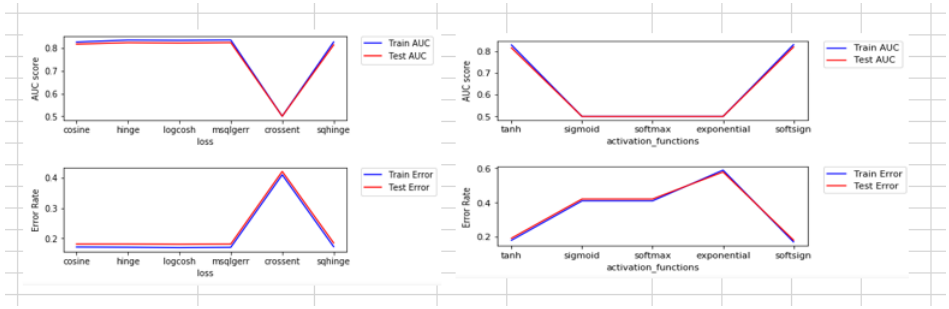
Test Set Error : 82 %

ROC and Confusion matrix for baseline model

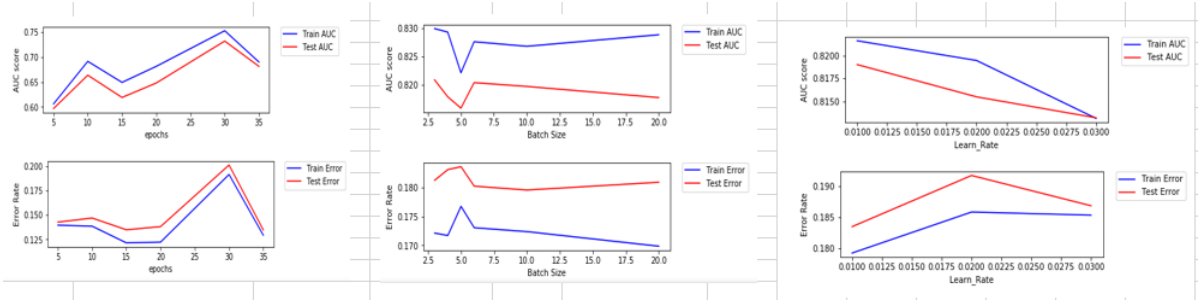
Here all the loss functional forms performs similar except the cross entropy, in this binary case it is similar to logistic regression cost function. Also softsign and tanh perform better than other activation functional forms. We have previously discussed softsign. tanh function is much similar to logistic except it approximates identity near the origin. We yet again see that the functional forms affect the model performance. The other 3 parameters have a predictable behavior and perform different with variability. We have higher optimal batch size this time, and also the model converges relatively quickly, optimal epoch are 30 and learn rate 0.02, note lower learn rate means optimizer takes more time to converge high could lead to overfit.



Learning curves for different params- Hidden Layer Count and Neurons per layer

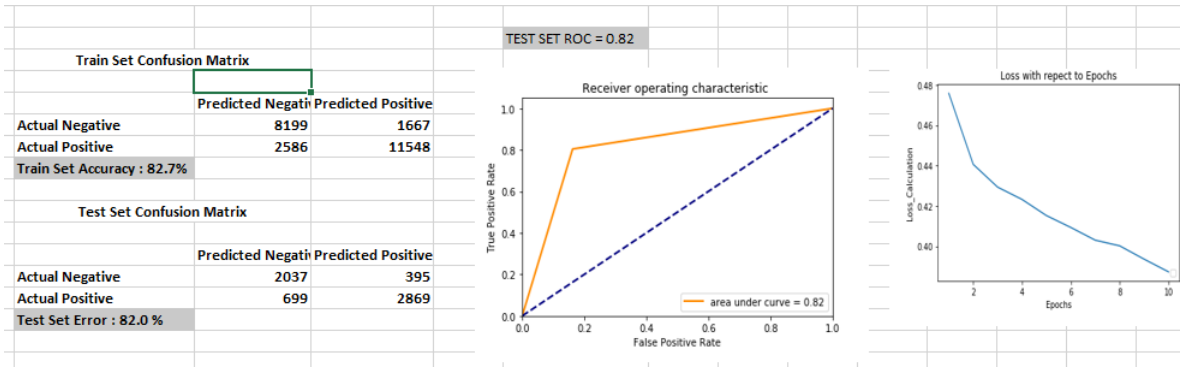


Learning curves for different params- Loss and Activation Function



Learning curves for different params- batch size. learn rate, epoch

Below we see the test set AUC, plotted, it is slightly improved compared to the baseline model, however in this case we do not see visible improvement over baseline model. Grid search generated best performing model is plotted below. Another important thing to note is that there is little different between train and test set error and auc metrics, it is a good indicator that the model is able to replicate the in-sample results out of sample as well. This concludes ANN implementation for Kickstarter campaign success prediction.



ROC and Confusion matrix for tuned model

3.2 Kickstarter Campaign Success: K Nearest Neighbors

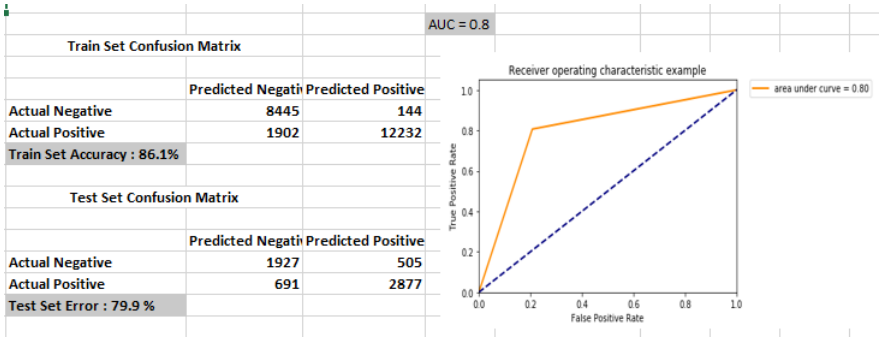


Fig: Baseline model and ROC curve

We have mentioned the key parameters that are needed in the optimization of the k nearest neighbours. In the applaince spike prediction we hadn't defined properly the weights metric, it has two forms uniform and distance, in uniform weight all the points are given same weight age, in the distance metric the weights are determined in terms of inverse of the distances, we see that the baseline model gives a poor out of sample error there is a case of overfiting as there is a gap between in sample and out of sample errors. We can inspect the reasons when were see the tuned individual parameters.

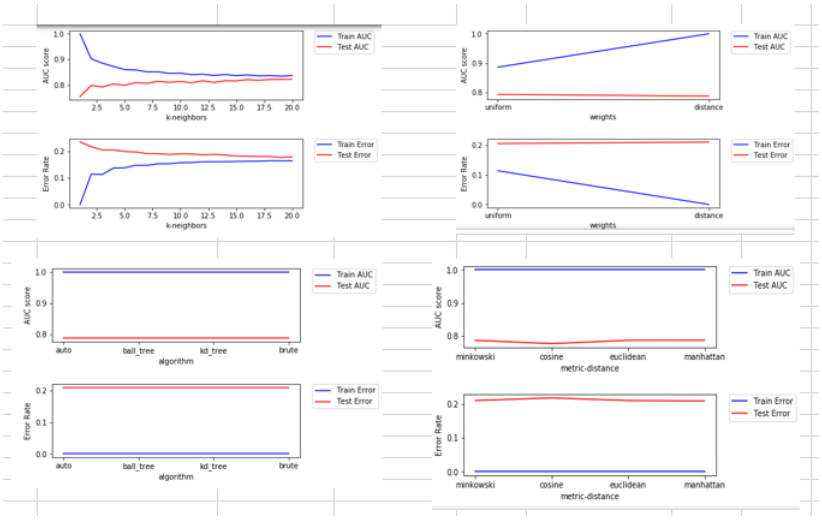
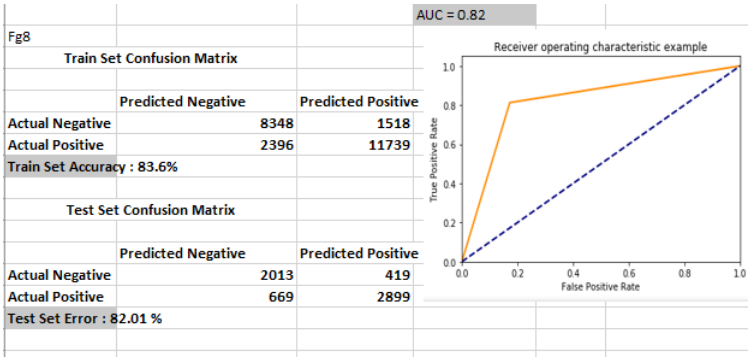


Fig: Knn Parameter tuning

As expected the test error constantly decreases as the number of neighbors increase, overfitting on the baseline model using default value of k is visible there. We also observe the patterns toggling between odd and even toggling, similar to previous dataset with appliances, the reason behind it is same as observed. In both this project and the previous data, we have used algorithm when using out of sample tuning by validation, it is the algorithm used to find nearest neighbours that has inbuilt variables and auto option. In this and previous they have no effect on the out of sample performance. This could indicate that in simple, uncomplicated datasets like these two, all the algorithms do there jobs just fine of finding k nearest neighbors. Below we see massive improvement in out of sample error, this is after implementing the hyper-parameter tuned model (for specific grid search params and optimal models refer the code), it indicates that the overfitting which was observed in the baseline model has largely been overcome, this is due to increase in k neighbour size of the tuned model.



Fg8

Train Set Confusion Matrix

	Predicted Negative	Predicted Positive
Actual Negative	8348	1518
Actual Positive	2396	11739

Train Set Accuracy : 83.6%

Test Set Confusion Matrix

	Predicted Negative	Predicted Positive
Actual Negative	2013	419
Actual Positive	669	2899

Test Set Error : 82.01 %

left: Test Error with varying train size, right : ROC for optimized model

4 Model Comparison and Important Learning Outcomes

Data Metric	Boosting	Decision Trees	SVM	ANN	KNN
Kickstarter AUC	0.74	0.80	0.80	0.82	0.82
Appliances AUC	0.80	0.77	0.72	0.62	0.715

- We compare the results across previous projects and this project, the consistency of the datasets gives us ample scope comparing the models, we use AUC test metric for the same. One of the reasons could be that it is a more complex problem in terms of engineered features and dimension of information conveyed as compared to Appliances Spike prediction. Hence ANN does a comparatively better job than relatively simpler parametric algorithms like tree based algorithm. The inverse is also true, as ANN and SVM perform relatively poorly compared to tree based algorithms in Applainces Spike prediction.

- Along with learning implementation of machine learning algorithms, one of the biggest focus of this assignment was the technique of cross validation (or tuning parameters with out of sample error metrics), it is an invaluable mechanism of knowing on how the model will perform out of sample without going to the test data. It is also a leverage to optimize various controlling levels of the model, basically the input parameters using cross validation as a technique.

- In Artificial neural networks parameter tuning *we learnt that functional parameters like activation function and loss functions are more important tuning parameters as compared to neural network structure parameters like layers and nodes or learn rate, epoch, batch size.* It is because parameter like activation function is clinical and present in every part of Neural network. More layers or nodes would likely result in inactive , sparse nodes and junctions of the network, specially in simpler classification problem.

- There is more scope of parameter tuning, and improving the two models, lot of parameters can be initialized by user callable functions, *these improvisations give a window to inject domain knowledge*, for instance metrics and weights of knn are callable thorough a function.

- *Importance of cross effects in parameter tuning* tuning a parameter individually and mapping the auc and out of sample error is different from doing combined grid search with a group of parameters. The end converged optimal value can be different in case of grid search/random search. It is because bias/variance trade off can be offset by one variable (say optimizer learn rate) and the expected optimal value of say epochs may be different than expected, it was observed in the case of ANN in kickstarter predictions.