***Abstract***

The rate of readmission of inpatient diabetic encountered patients depends on a lot of factors such as the age or race of the patient, medications are given to the patient, service quality of the hospital management system and much more. These patients are categorized into 3 divisions of never being readmitted to the hospital, or being readmitted after a period of 30 days or being readmitted within 30 days. The diabetic dataset was first divided into 2 parts – “training dataset” and “testing dataset” keeping the percentage class distribution the same in comparison to the original dataset. Before the training data was every given to the model the data needed to be cleaned and pre-processed and various measures such as treatment of missing values, outlier detection, and removal, categorical data merging, feature dimensionality expansion or reduction and normalization were used. The pre-processed data was then sent to various types of classifiers to train them for the incoming testing dataset. It was found that an ***Ensemble Random Subspace kNN Model*** gave the best ***F1-score of 0.82*** on the testing dataset. It was found that the hospital management services – “***time in hospital***”, “***number of lab procedures***”, “***admission source id***”, ”***admission type***” and ”***discharge disposition***” were one of the key factors that lead to patients returning back to the hospital or not. It was also found that the dosage of ***insulin, age, race, gender and diagnosis 1*** also played an important in correctly classifying the inpatient diabetic encounters.

***Goals and Methodology***

The primary goal of this project is to find a classification model which can accurately categorize incoming unknown data points with any overall accuracy of greater than 65 % and also identify the main features that contribute to readmission of patients back to the hospital. To achieve this goal meant that the data had to be pre-processed and cleaned thoroughly and then sent to suitable classifiers to model them for the testing dataset.

***Pre-processing Of Dataset***

The data set had to be first divided into 2 parts to give us a training set and a testing set which would be used once the classifier is trained on the training set.

1. ***Training and Testing Set Formation***

The dataset is divided into two parts randomly with an even distribution of the each class in each part using the ***cvpartition ()*** MATLAB inbuilt function using 2 fold split. This split the 101766 x 49 data set into a 50883 x 49 data set - ***feature\_train*** and a 50883 x 49 data set - ***feature\_test***. Column number 50 is changed from a string array to an integer array and also split according to its relative split of the rest of the 49 column thus leaving us with a 50883 x 1 data set - ***label\_train*** and a 50883 x 1 data set ***label\_test***.

The training data set will be used for preprocessing and designing the pattern recognition system. The test data set will not be touch until the final classification system is not ready. Any kind of preprocessing applied to the training data set will directly apply to the testing data while testing the classifier.

1. ***Preprocessing Of Training Data Set***

The training set is now sent to another MATLAB function – ***preprocessing.m*** where the training data set gets preprocessed such that the relevant data is sent to the classifier. Some of the techniques used to preprocess this data set are: recasting of the features into numeric form, removing/replacing of missing data with an appropriate representation of the data present in that column, removing sample points and perform feature dimensionality reduction/expansion on each column of the data set, normalization of the data set if required.

1. ***Recasting the representation of Feature***

The training set is imported into MATLAB as a ***table*** which contains different data types. We need to firstly recast the data set into single datatype. ***Numeric*** columns/features are not changed throughout the entire data set. On the other hand, ***String*** data set is converted into ***Numeric*** form by allocating different numbers for a different string set in the column entry. In medications columns entries - 25 to 47 the data is given by any one of these ***strings {'No'}, {'Down'}, {'Steady'}, {'Up'}*** and each string is allocated numerical values from 1 to 4 respectively.

1. ***Data Point Reduction***

At this stage of pre-processing, there were some duplicate/irrelevant data entries in the ***patient number feature*** and ***discharge disposition feature*** and these were dealt with as follows. There are occurrences where a particular patient is readmitted to the hospital which can be assessed by checking the ***patient number feature*** dataset. Patients returning to the hospital are not the first admits and hence cannot be considered as a part of the training data set. These data samples are hence discarded. Also, patients ***discharge disposition feature*** tells us if the patient has ***expired or hospice*** (terminally ill) which means that these patients have near to zero chance of returning back to the hospital and is considered irrelevant data and hence discarded. For the ***testing data*** set these patients were directly classified into class ***“NO”*** i.e. as these were expected to never return.

1. ***Missing Values Treatment***

Dealing with missing values is really important as it can lead to the biased performance of the classification model on the training dataset. There are various ways of treating missing data depending on the kind of features that are dealt with. The missing data in the diabetic data set is represented as ***NaN*** values or strings as containing ***‘?’***. Firstly we need to find out which features have missing data and the amount of occurrence of the missing data. This can be done using the ***isnan()*** or ***strcmp(feature(:,:),{‘?’})*** function in Matlab.

***Weight Feature***

This feature had more than ***97 % of missing*** data and hence this feature held no relevant information and was too sparse to be considered for classification at all and hence this column was ***completely removed*** from the training dataset.

***Paper Code and Medical Specialty Feature***

The ***paper code feature*** data also has > 52 % data missing and for the same reason mentioned above this feature is removed too. On the other hand, the ***medical specialty feature*** too has > 53 % data missing but we don’t discard those as we consider that irrespective of the specialty is not mentioned that patient would have received some kind of specialist with a medical specialty assigned as ***other***.

***Gender Feature***

The gender feature had ***unknown/invalid*** as few of the entries (only 3) in the entire diabetic data set and those data points were removed from ever being classified in the training or the testing data set.

***Diagnosis 1 2 and 3 Features***

Diagnosis 1 2 and 3 features had some missing data (~ 1%) which needed to be dealt with in a different manner as compared to the above features which were completely eradicated. There were various methods though of while handling this kind of missing data i.e. complete ***deletion*** of the data sample, ***mean/mode imputation*** of the data or using a ***KNN imputation*** (***kNNimpute (feature\_train\_pp (: , 19) , 2***). Due to randomly placed missing values, it was concluded that best way of handling missing data was using the ***deletion method***. It was also found that diagnosis 1 2 and 3 are high dependent features and hence using only diagnosis 1 – Primary diagnosis was decided. This meant that data points having missing values in diagnosis 1 feature column were only removed as diagnosis 2 and diagnosis 3 columns were completely removed later on.

1. ***Feature Dimensionality Reduction***

Many features in the diabetic data set contain irrelevant data and that leads to the poor classification of testing dataset if they are not discarded. This kind feature selection is done to reduce the dimensionality of the dataset and is carried out in two ways – finding the ***correlation*** between features and looking of near to ***zero value variance*** of a feature. Features that also have a large number of ***unique values*** that is the number of unique values is close to the number sample points those features are also eradicated from the training as well as the testing data set.

***Correlation and Near Zero Variance***

* This kind feature dimensionality reduction depends on the correlation among different features and if a particular ***feature is highly correlated*** with other features then that feature is ***completely removed*** from the training and testing data set. This is achieved using the Matlab function ***corrcoef(feature\_train\_pp,’rows’,’pairwise’)****.* This function checks the correlation between all the 49 features and results into a correlation matrix.
* Many features in the diabetic dataset have near to zero variance which means that those features very ***less differentiable data points*** and these data do not hold any importance when given to classifier to model the data set. This is done using the ***var()*** function and then finding out which column variances were near zero value (< 0.1).

This lead to the conclusion of removal of “***diagnosis 2***”, “***diagnosis 3***” and “***bags of medications***” except for the ***insulin*** feature column which had the highest variance in comparison to the other medications that were prescribed to the patients.

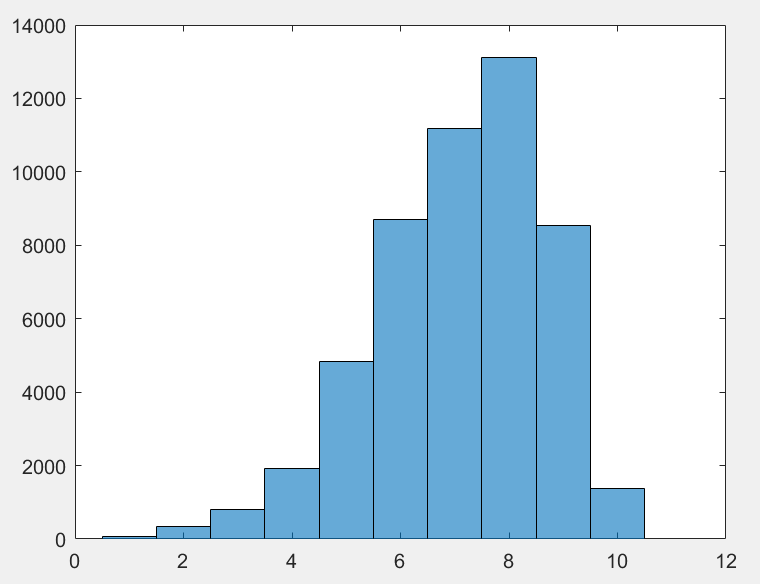
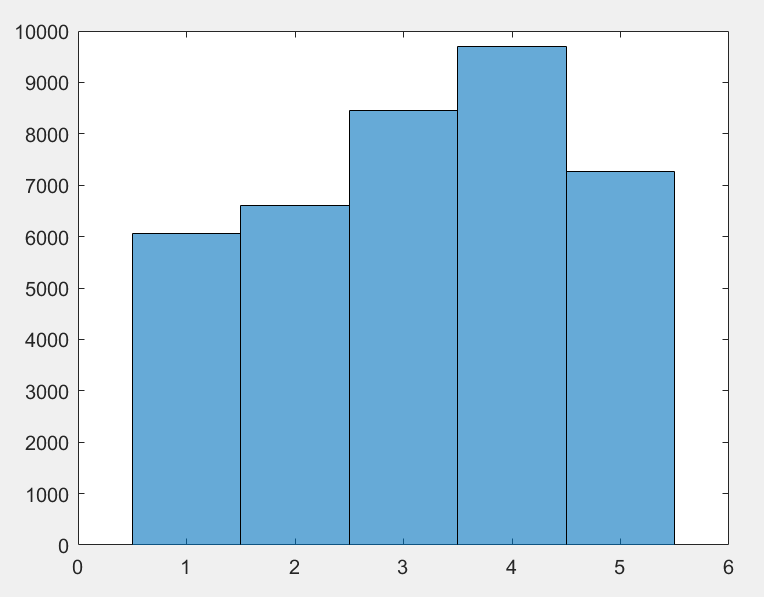
***Unique Values -***

Some of the features such as the “***encounter id***” and “***patient number***” have a large number of unique data and these features are also removed from the entire dataset.

1. ***Encoding Categorical Data***

In this section of pre-processing, each column was analyzed to check if it was applicable for data point merging with the same feature space. This was done to remove vagueness within a particular feature column were more than one data value would be pointing towards the same class or depicting the same data interpretation. This was done in reference to <http://www.hindawi.com/journals/bmri/2014/781670/tab3/>

This is kind of data merging was done on some features using the histogram plot were few data values of less importance were merged into one so increase this importance during classification because if not done they would be too sparse to be used by classifiers. A feature that was used for histogram merging were ***“age”*** and the rest of features were merged in accordance with the table provided in the above link.

***Figure 1:*** *Feature “age” Before Merging* ***Figure 2:*** *Feature “age” After Merging*

Let us consider feature column ***“age”*** which has 10 unique values but the age distribution of patients is not uniform over the range from 0 to 10 (0 – 100 years old) hence, these data points were merged into 5 categories as: ***(0-50), (50-60), (60- 70), (70-80), (80-100)***.

* The ***“race”*** feature column was also merged on the same principle into 4 categories as ***missing, African American, Caucasian, others***.
* The ***“diag 1”*** feature was categorized into 9 features based on the data provided by ***“List of ICD-9 Codes”*** referenced from here: <http://www.hindawi.com/journals/bmri/2014/781670/tab2/>
* Additional features like ***“admission type id”, “discharge disposition id”***, ***“admission source id”***, ***“medical specialty”*** were merged in reference to the link provided containing [Table 3](http://www.hindawi.com/journals/bmri/2014/781670/tab3/).

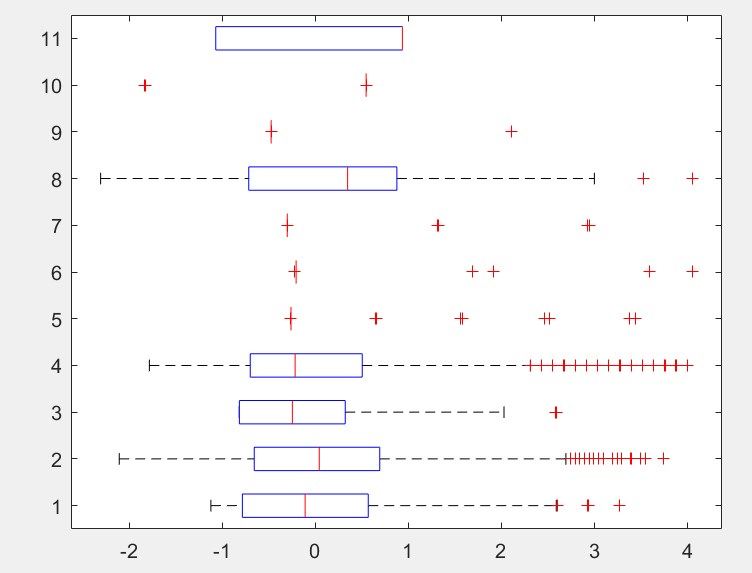
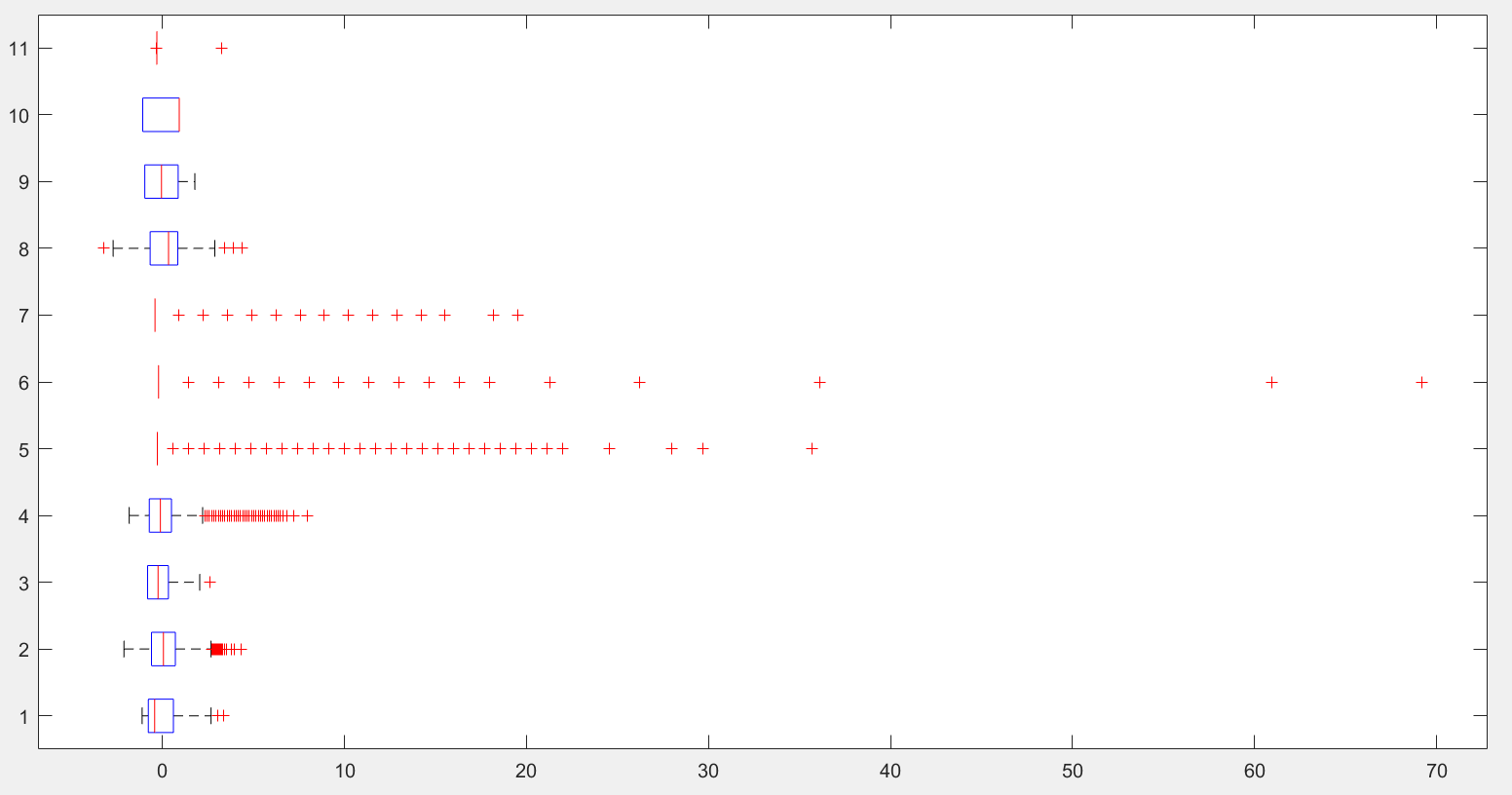
1. ***Categorical Feature Expansion***

In this method of pre-processing, we will look into expanding the categorical features into ***indicator features*** using numerical to binary conversion. This kind of creation of new features helps in highlighting hidden information which wasn’t accessible before. The categorical features that were expanded are ***“age”***, ***“race”,*** ***“gender”,*** ***“admission type id”, “discharge disposition id”***, ***“admission source id”***, ***“medical specialty” and “diagnosis 1”.*** The purpose of this kind of feature transformation was to separate irrelevant data point values from the data set without the need to completely removing that data point from the diabetic dataset. Categorical values having very low importance/occurrence within a feature space would be removed completely when expanded to indicator feature space due to ***near to zero variance*** property of that particular feature.

***Note:*** Categorical feature expansion was done by ***merging the testing and training dataset*** together to ***prevent an unequal number of features*** after expansion and also prevent different features from being removed.

1. ***Outlier Detection and Removal***

An outlier is a data point that statistically inconsistent with rest of the data set. In this project, we looked into ***univariate*** outliers only. Most of these outliers were dealt with by using the categorical feature expansion method. To find the remaining outliers we can look at the box and whisker plots. The outliers are only removed from features having continuous values as categorical features have been converted to indicator feature space. Hence, outlier detection was done on the basis of the ***modified Thompson tau*** ***technique***. In this kind of outlier detection and removal, only 1 data point is considered at a time and the new means and standard deviations are calculated every single time an outlier is detected and removed.



***Figure 3:*** *The above box plots shows that features 4, 5, 6 and 7 (****'num\_medications', 'number\_outpatient', 'number\_emergency', 'number\_inpatient'****) have the maximum number of outliers and the modified Thompson tau technique yields the same results when a maximum of 1000 outliers are chosen.* ***>> [~,idx] = outliers(feature\_train\_pp,2000); >> unique(idx(:,2)) ans = 4 5 6 7***

1. ***Normalization***

All the data set were normalized such that the entire data set is within a predefined set range. Un-normalized data was also run on various classifiers and it was found that the classifiers were very sensitive to vast differences in the data set values or skewed data and hence needed to be normalized. The categorical features that were expanded were also normalized. The reason behind doing this was to adjust the dataset for the kNN classifier to prevent every data point with 0s and 1s to lie within the 34-dimensional hypersphere. Normalizing the features would result in the irregular distribution of the data points within and outside the hypersphere and hence would prevent any conflict arising during selecting the k nearest neighbors.

***Classifiers***

The project uses 3 classifiers: ***Naïve Bayes***, ***Perceptron,*** and ***Ensemble Subspace kNN Model.*** The ensemble classifier gives the best results in comparison to the other 2 classifiers. Many other classifiers like MSE, SVM, Decision Tree, Bagging and Adaptive Boosting were also tried out and the results were not satisfactory, hence the above 3 classifiers were chosen and are explained briefly below.

***Ensemble Random Subspace KNN Model***

The ***Ensemble Random Subspace KNN Model*** using the classification learner gave the best accuracy among the other 2 classifiers. This model was run using the Matlab toolbox ***“Classification Learner”.*** The toolbox was used to import the training set and an option of K-Fold cross-validation was also provided. Different values of K-Fold were chosen (10:10:50) during the classification modeling stage. It was seen that increasing the number of K for cross-validation would lead to increasing the accuracy on the testing set i.e. increase in F1 score.

To ***validate such a*** **classifier** a few tests were run to find the best:

* ***Number of K nearest neighbors***
* ***Random subspace dimension***
* ***Number of learners required***
* ***K-Fold cross-validation value*** which would give minimum classification error.

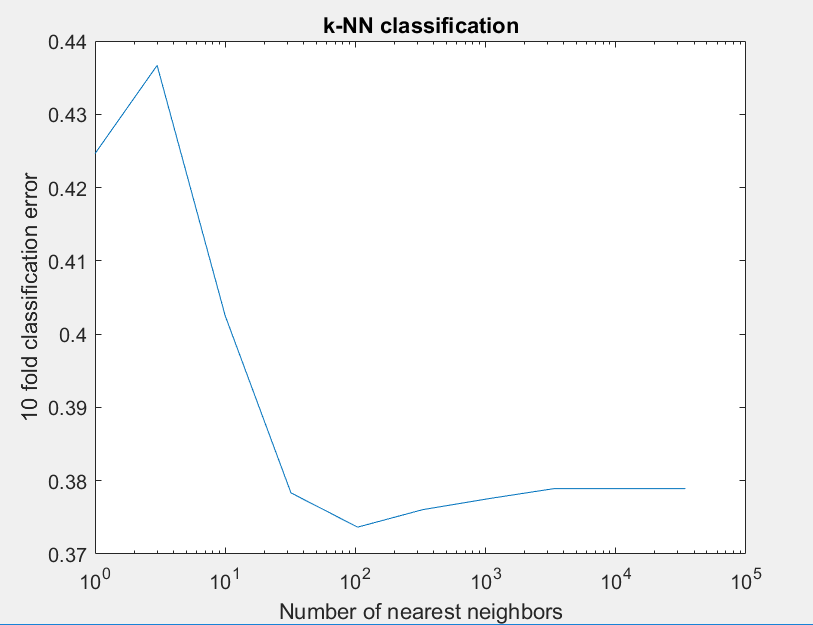
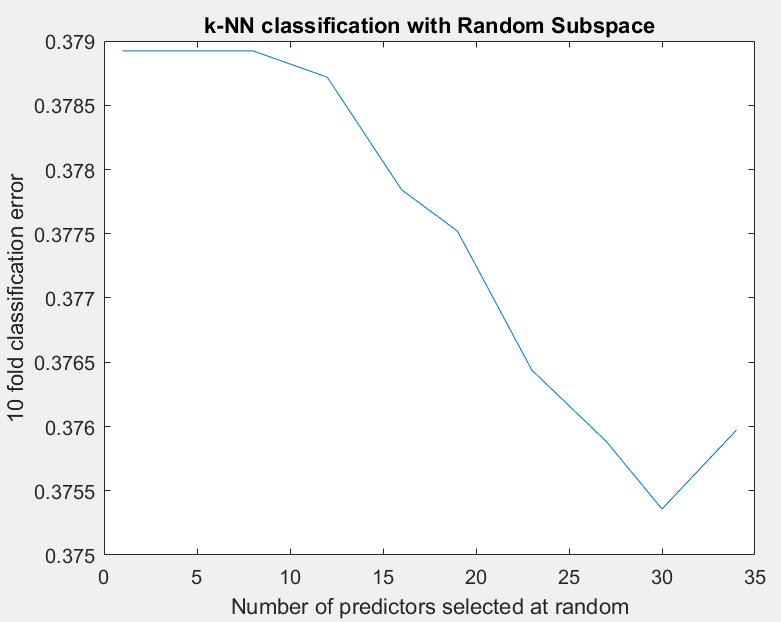
***Number of K nearest Neighbors***

This test was done on the training set itself. Here the ***fitckNN()*** function was used with 10 different values of the parameter ***‘NumNeighbors’*** over the range of number of samples using the ***logspace()*** function. It was found that a value of 104 nearest neighbors resulted in minimum classification error as it can be seen in the graph below.

***Random Subspace Dimension***

The kNN classifier is very ***sensitive*** classifier when it comes to ***irrelevant features*** in the dataset. The pre-processing that was carried out made sure that such irrelevant features do not occur while using any kind of classifier. In order to achieve diversity and accuracy with the kNN classifier we needed to sample out features if any that were highly sensitive to the classifier. The idea was to exploit the sensitivity of the NN classifier by generating diverse lower dimensional feature sets.

This idea was fed to an ensemble model such that multiple NN classifier (***learners***) would have access to random subsets of the features and their majority voted outputs would help in increase the accuracy of the overall ensemble classifier. To achieve this goal the ***best random subspace dimension*** needed to be found. This was done using an ensemble model with 100 learners, 10 fold cross-validation and 10 different subspace dimension over the range of a number of features using ***linspace()*** function. It was concluded that a ***random subspace of 30*** ***dimensions*** gave the best results as seen in the graph below.

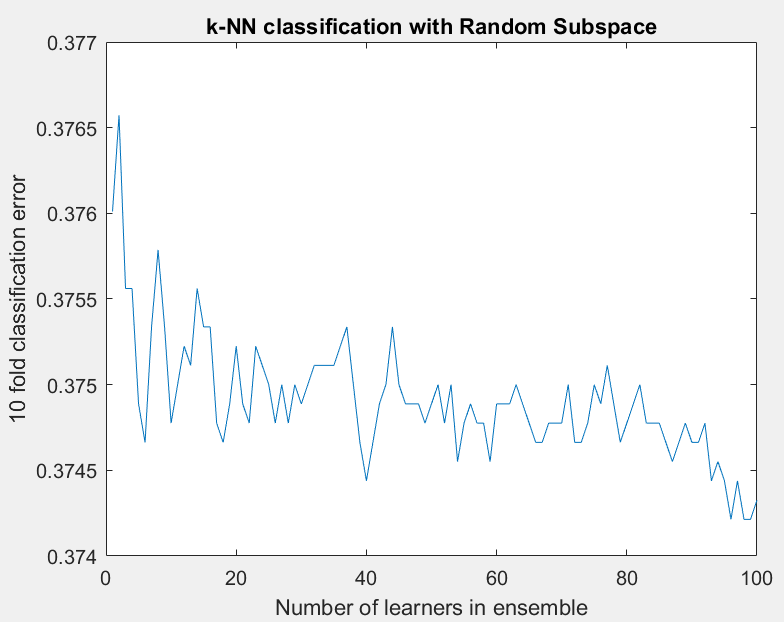
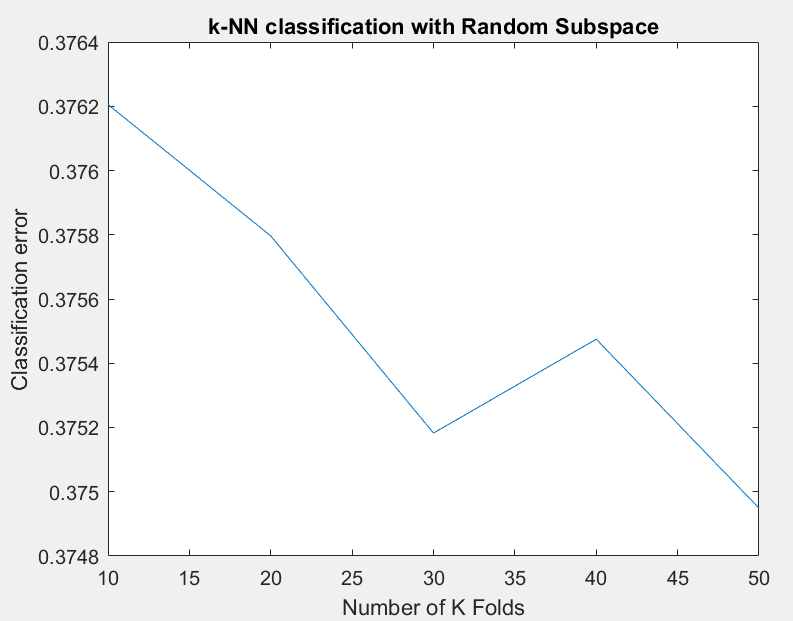
***Figure 4:*** Best number of nearest neighbors ***Figure 5:*** Best Random Subspace Dimension

***Number of Learners***

The smallest number of learners needed to be found to reduce the ensemble model size increase speed of classification. This was done using the ***kfoldLoss()*** function in ***Cumulative Mode***. This result is shown in the graph below and unfortunately, we can see that no more than 95 learners are required for this dataset.

***K-Fold Cross-Validation***

Using the above parameters the ensemble model was run again for different values of K Folds ranging from 10 to 50 over a step of 10. It can be seen from the graph below that a 50 Fold cross-validation yields the best results.

***Figure 5:*** Best number of learners ***Figure 6:*** Best Number of K Folds

These 4 parameters were chosen and fed to the “***classification learner***” Matlab toolbox to generate an ***Ensemble 30-Dimensional Random Subspace kNN Model with 50 Fold Cross-Validation and 90 Learners***. The toolbox provided a validation accuracy of 68.8 % on the training data set. The trained model (***trainedClassifier***) and a Matlab generated code as a function (***trainedClassifier.m***) were then exported into the workspace for further repetitive testing. This trained classifier in combination with a prediction function (***trainedClassifier.predictFcn()***) were used to get predicted labels on the testing data. These predicted labels were tested with the true labels and an ***F1 score of 0.82*** and an ***accuracy of 82.88%*** was achieved.

***Naïve Bayes Model***

The ***Naïve Bayes Model*** is an inbuilt Matlab function given by ***Md1 = fitcnb()*** and the ***predict(Md1.Trained{i,1},feature\_test\_pp)*** is used to run this model on the testing dataset. Since the Naïve Bayes runs based on the probability density function and the class priors this is not a distribution free classifier. Different distribution methods such as the ***‘kernel’, ‘multinomial’ and ‘normal/Gaussian’*** were used to test the training dataset and it was found that a ‘***normal’*** distribution of all the samples resulted in the best accuracy on the testing dataset. The training was done using 10 Fold cross-validation. The training using 10 fold cross-validation resulted in 10 different training models. These models were then used on the testing data and majority voted predicted label was computed. This classifier resulted in an ***F1 score of 0.56*** and an ***accuracy of 58.72%*** on the testing dataset.

***Perceptron***

The Perceptron classifier is distribution free classifier and is modeled using the ***multiclass()*** function provided in the classification toolbox. The model was run using the ***one vs all (‘OAA’)*** algorithm to classify data between these 3 classes. The maximum number of iterations used to achieve the optimum weight vector was ***2000 iterations***. The Perceptron classifier is not a good classifier in comparison to the above two classifiers as it does not take into consideration the class distributions. This classifier gave the best ***F1 score of 0.52*** and an ***accuracy of 56.48%*** on the testing data set.

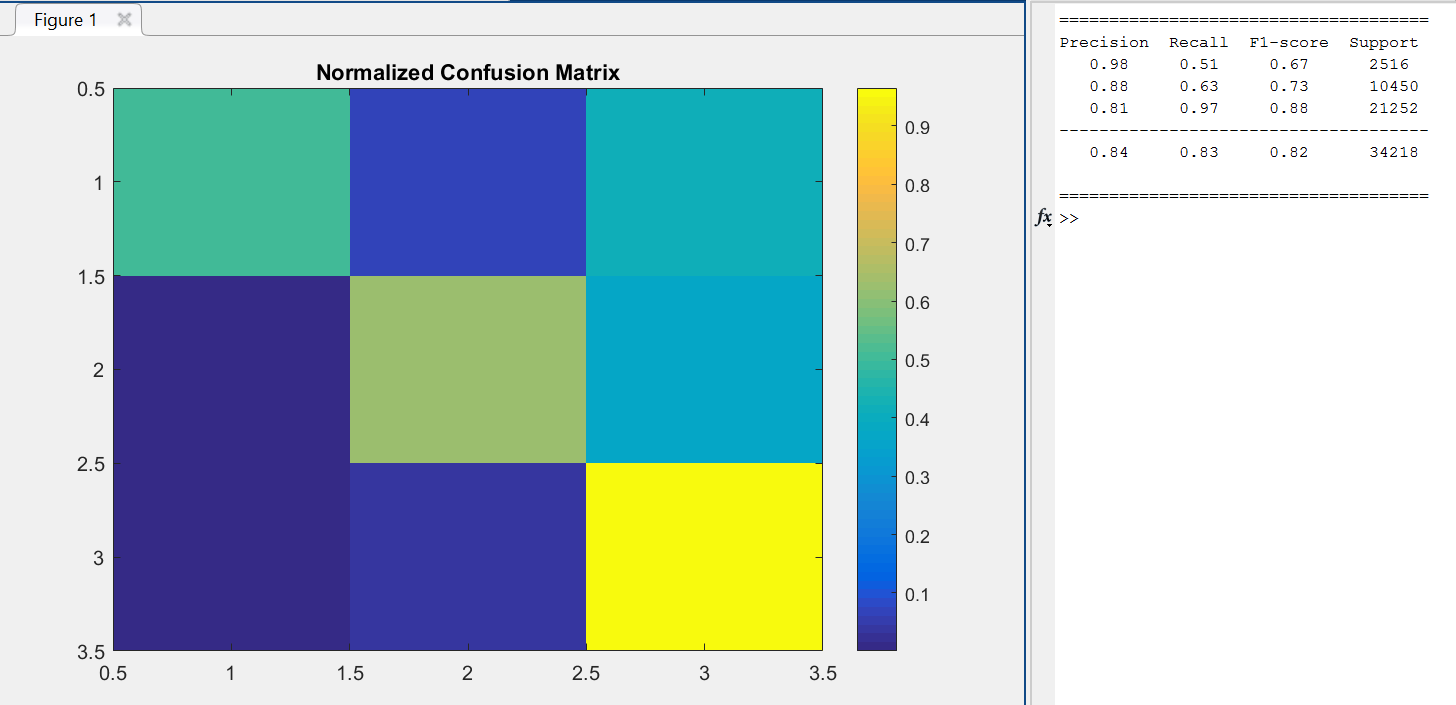
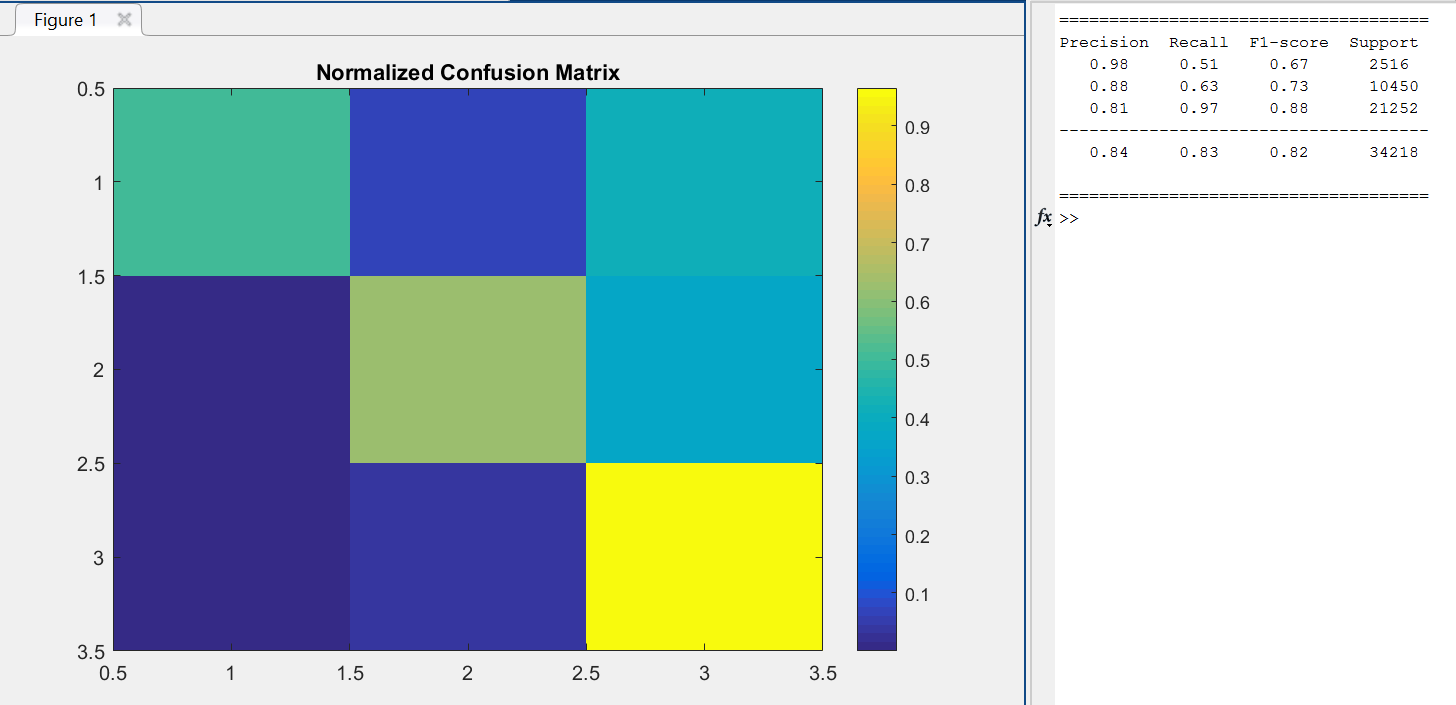
***Performance Evaluation Techniques***

The performance evaluation was done on the testing data set that which would go through the same pre-processing steps as the training data did. The testing data was then fed into different classification models and predicted labels were obtained from testing the test data on the trained classifiers. This predicted data was then sent to a function ***classification\_report()*** to test the predicted labels with the true test labels. This function resulted in a ***confusion matrix*** and an ***F1 score*** for the testing dataset. A confusion matrix is a table that is used to describe the performance of classification model on a set of testing data set whose true label test is known. The classification performance of each classifier is shown below.

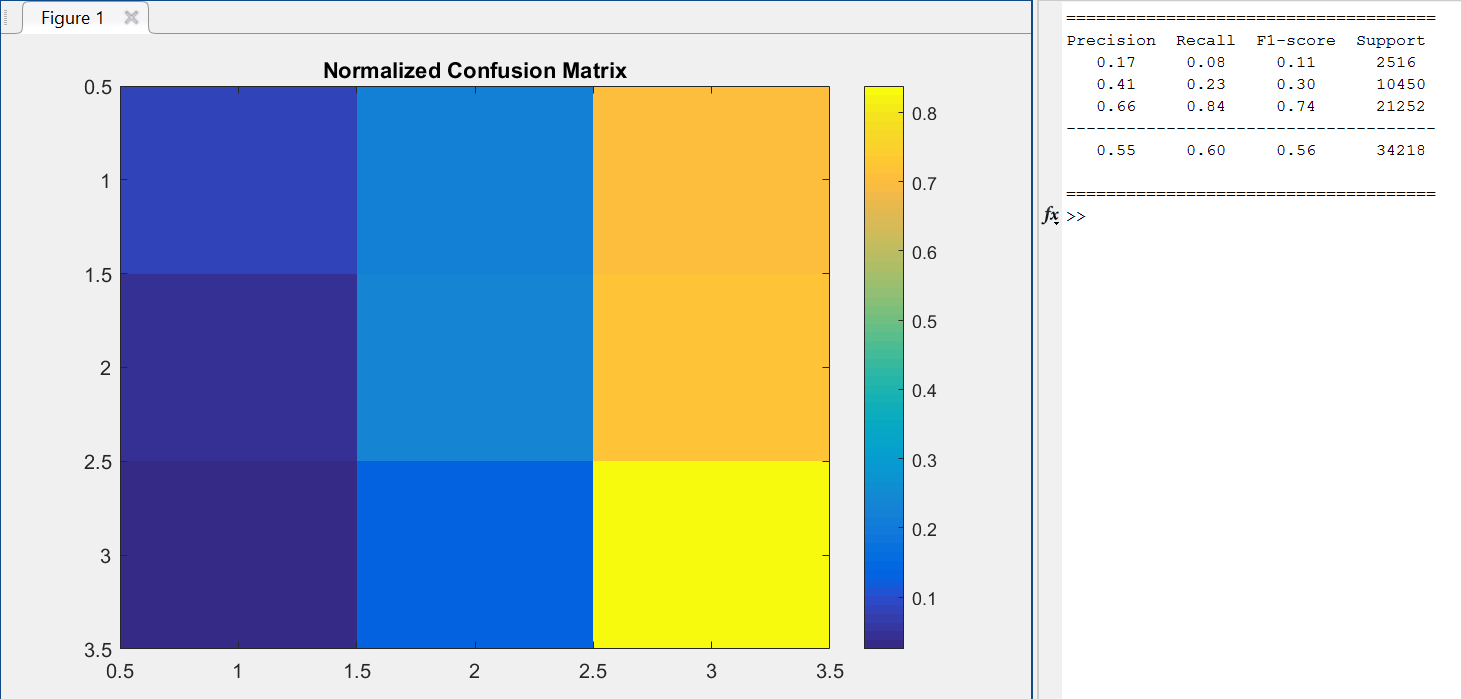
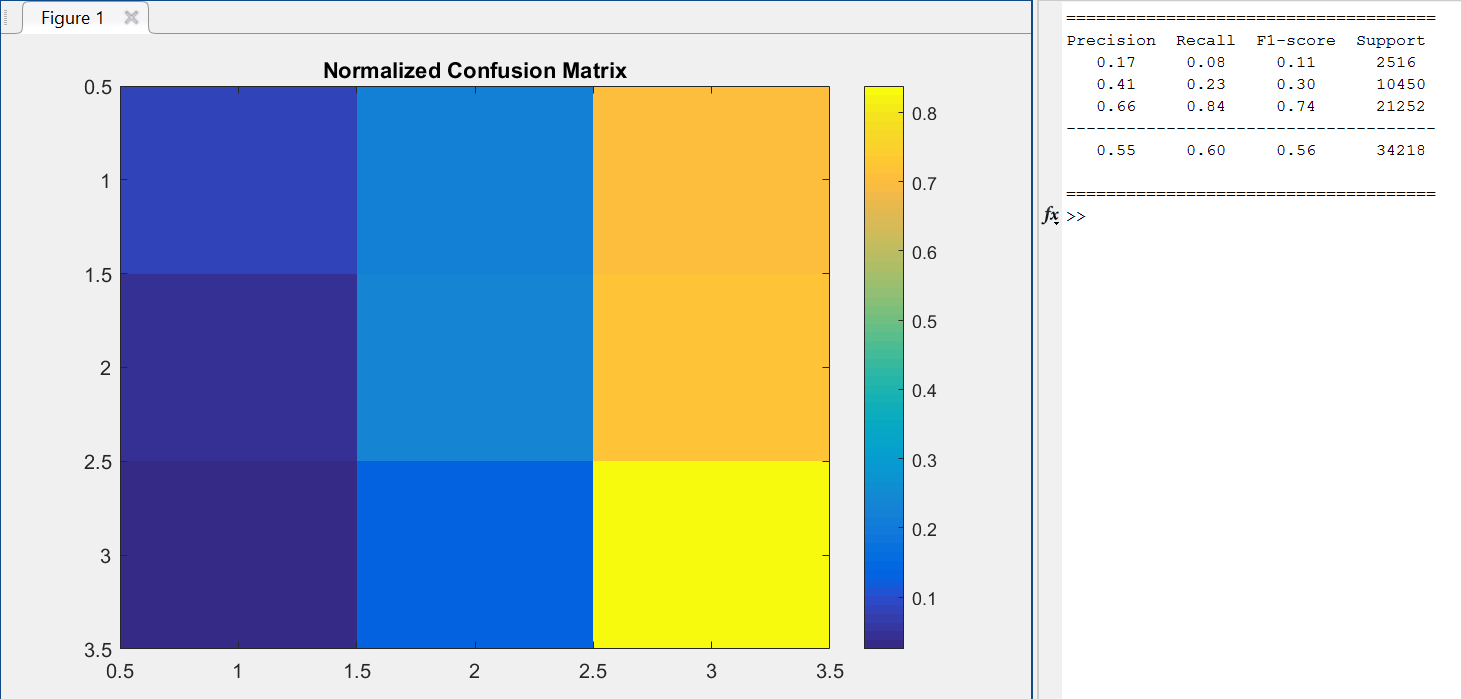
***Results***

The results of various classifiers are shown below using the ***classification\_report()*** function provided to get the ***confusion matrix and the F1 score.***

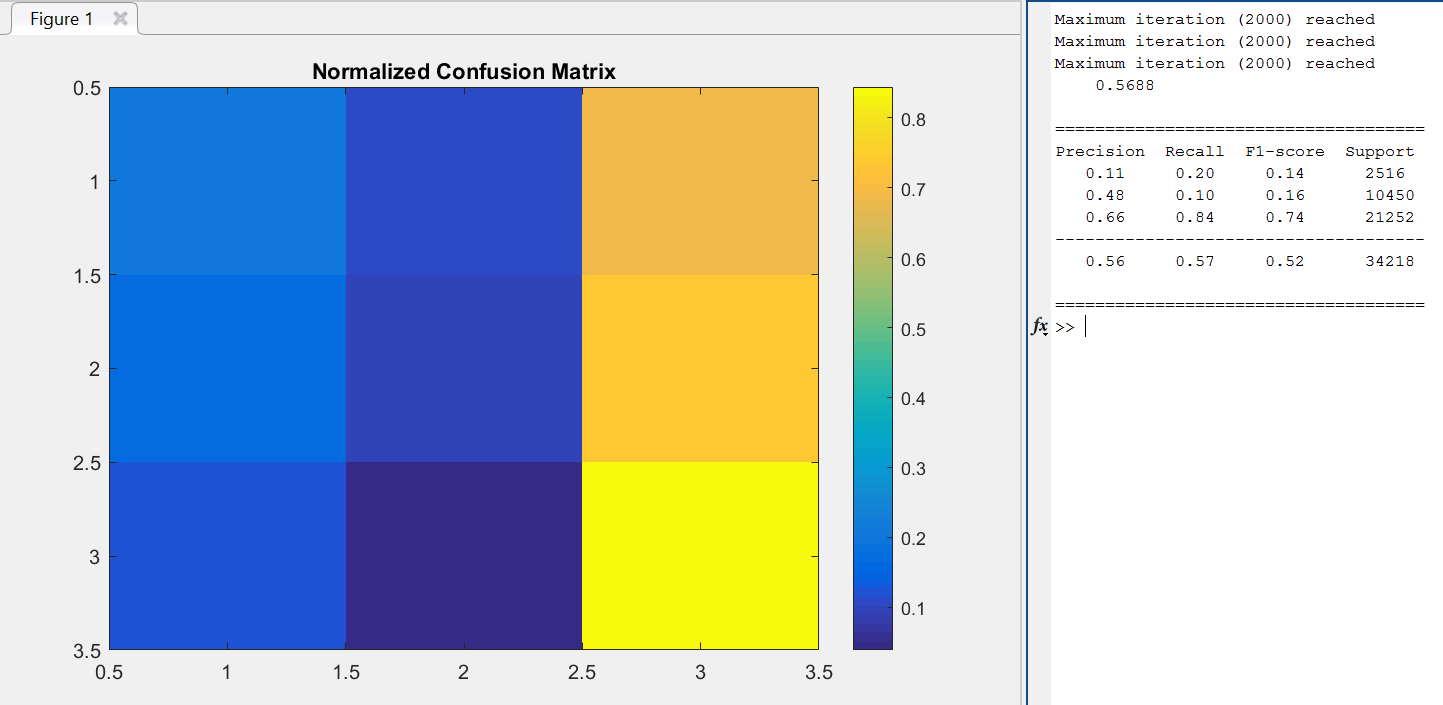
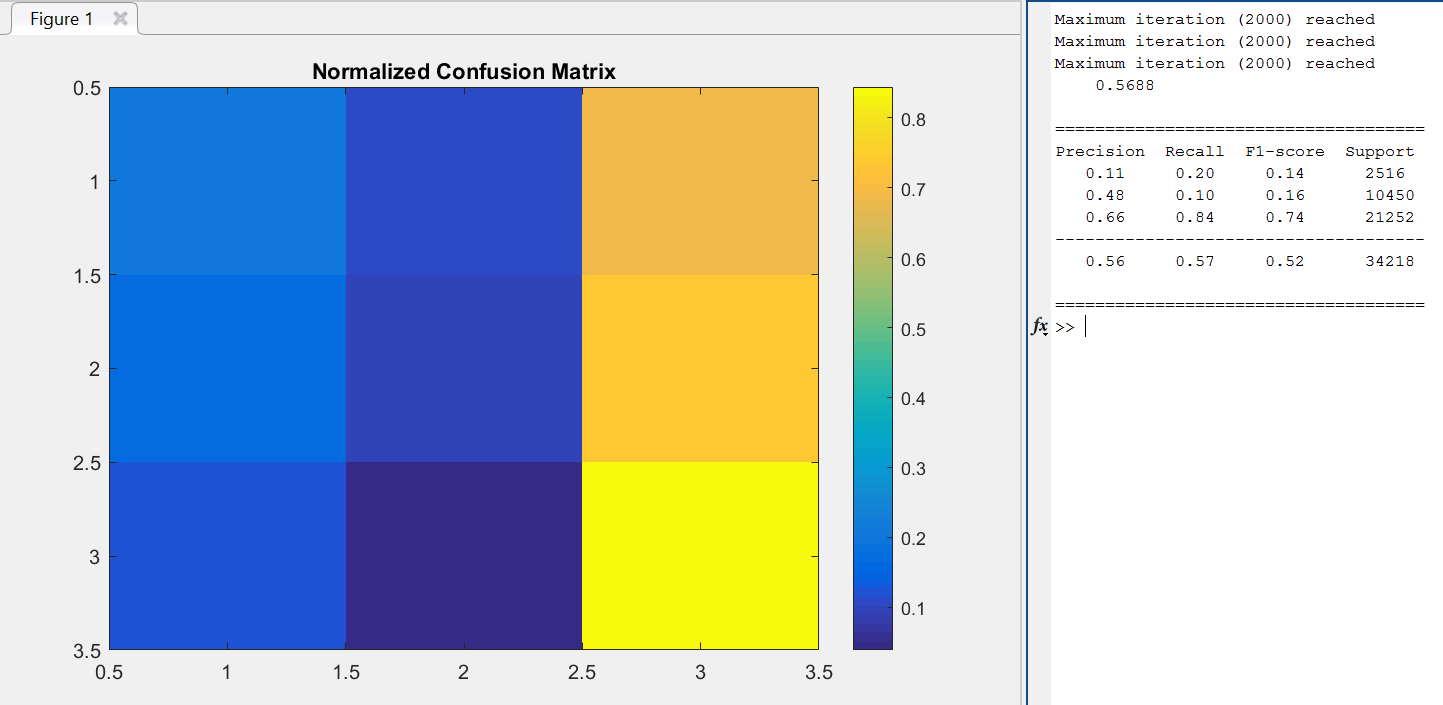
***Ensemble Random Subspace KNN Model – BEST RESULT***



***Naïve Bayes Classification***



***Perceptron Classification***



***Interpretation/Conclusion***

* Pre-processing of data is the most important part of the entire project on improving the accuracy of the classification models. It has been inferred that some features contain more relevant data as compared to others and the pre-processing steps helped in identifying such features.
* Important factors that lead to readmission of patients were “***time in hospital***”, “***number of lab procedures***”, “***admission source id***”, ”***admission type***” and ”***discharge disposition***”. These features are directly related to the type of management services provided by the hospital staff and it could be seen that a poor hospital management leads to readmission of diabetic patients back to the hospital.
* ***Early discharge*** of patients to their ***homes*** resulted in patients returning back to the ***within 30 days.*** It was also noticed that discharge of patients to various facilities also affected their readmission rate and it is advisable for patients to be treated in a suitable environment.
* The more amount “***time in hospital***” and “***number of lab procedures***” resulted in better health of the patients and directly affect the readmission rate of those patients. Hence, these features played a crucial role in the classification model. Primary diagnosis of the patients in accordance with the ***“List of ICD-9 Codes”*** was also considered as an important feature during classification.
* ***Categorical data merging*** and ***feature expansion*** helped in ***increasing the accuracy by 10%*** of the classification a lot as there were many data point values representing the same information.
* It was also seen that testing data sets in small bundles would results in better classification on the testing dataset and this led to choosing ***10 to 50 fold cross-validation*** techniques to prevent data overfitting. Using cross-validation ***increased the performance*** of the kNN classifier giving an ***F1 score of 0.82***

***References***

* ***Diabetes patient Readmission Prediction using Big Data Analytic tools*** - Xing Yifan, Jai Sharma <http://www.jimxingyf.com/pdf/CSE4095.pdf>
* ***Multi-Labeled Classification of Demographic Attributes of Patients: a case study of diabetic patients*** - Naveen Kumar, Parachur Cotha1 and Marina Sokolova <https://arxiv.org/ftp/arxiv/papers/1503/1503.07795.pdf>
* ***Predicting Readmission of Diabetic Patients using the high performance Support Vector Machine algorithm of SAS® Enterprise Miner™*** - Hephzibah Munnangi, MS, Dr. Goutam Chakraborty, Oklahoma State University, Stillwater, Ok <http://support.sas.com/resources/papers/proceedings15/3254-2015.pdf>
* ***Data Mining for Diabetes Readmission Prediction*** – Yi Chun Chien, Xiayu Zeng, Hong Zhang, Yixi Zhang <http://www.slideshare.net/NancyChien/data-mining-for-diabetes-readmission>
* ***Distribution of variable values and readmissions*** (population size is 69,984) <http://www.hindawi.com/journals/bmri/2014/781670/tab3/>
* ***Identifying Diabetic Patients with High Risk of Readmission*** - Bhuvan M S, Ankit Kumar, Adil Zafar, Vinith Kishore, National Institute of Technology Karnataka, Surathkal, India <http://arxiv.org/pdf/1602.04257.pdf>
* ***Large-Scale Support Vector Machines: Algorithms and Theory*** - Aditya Krishna Menon <https://cseweb.ucsd.edu/~akmenon/ResearchExam.pdf>
* ***Predicting the treatment effect in diabetes patients using classification models*** - Haoting Zhang, Lingyun Shao, Chengyu Xi, Guanghua International School of Shanghai, Shanghai 200040, China <http://www.aicit.org/JDCTA/ppl/JDCTA3689PPL.pdf>
* ***Random Subspace Classification*** <https://www.mathworks.com/examples/statistics/mw/stats-ex98983766-random-subspace-classification>
* ***Nearest Neighbor Ensemble*** - Carlotta Domeniconi, Bojun Ya, Information & Software Engg Dept, George Mason University <https://cs.gmu.edu/~carlotta/publications/NNensemble.pdf>
* ***A Comprehensive Guide to Data Exploration –*** Sunil Ray<http://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/#three>
* ***Outliers -*** John M. Cimbala, Penn State University <http://www.mne.psu.edu/cimbala/me345/Lectures/Outliers.pdf>
* ***Remove Outliers –*** M Sohrabinia <http://www.mathworks.com/matlabcentral/fileexchange/37211-remove-outliers>