EE 559 Project(Mathematical Pattern recognition)

Evaluating and testing the best classifier for Bank Marketing campaign data

Date:04/29/2018

Author: **Ruchin Patel**
email: ruchinpa@usc.edu
Usc id: **9520665364**

ABSTRACT

This project uses a dataset that contains information on people targeted by a bank marketing campaign. The goal would be to find the best classifier that predicts if a client would subscribe to a long term deposit depending on the new data. The first thing that would be done in the project would be creating a baseline model, and then using cross-validation for error estimation and tuning the hyper parameters. Four classifiers would be trained, namely **Perceptron**, **SVM**, **Minimum Risk Bayes Classifier** and **Neural Networks**. All of these classifiers would be compared with the baseline model as well as with themselves in order to find the best classifier suited for the job. At the end of the report we conclude that the data about bank tele marketing that we processed and tried to classify using different classifiers was not linearly separable and as a result of this non-linear techniques like SVM with 'rbf' kernel and Artificial Neural networks worked well for the classification of data.

Description

- Data-set: The data-set is related to direct marketing campaigns through phone calls of a Portuguese banking institution. The data set contains 4119 examples and has 19 features in it. The classes corresponding to every example in the dataset has one of the two unique values('yes' or 'no'). This dataset is inherently unbalanced and is highly biased towards the 'no' responses. The ratio of the number of examples with the two responses 'yes' and 'no' is approximately 0.12. i.e(#('yes')/#('no') = 0.12). This means that the examples with 'no' responses are almost 8 times the examples with 'yes' response.

- Imbalance: This imbalance should be taken care off by novel techniques which in the case of this project is handled by varying class weights for Perceptron and SVM classifiers and varying the loss for Minimum risk Bayes classifier. Another method of oversampling the data set is also used in the project for perceptron, SVM and neural Network classifiers to combat the imbalance in the data. The performance of all the classifiers will be compared to decide the best classifier for the task at hand.

- Performance metric: The main metric of performance would be F1 score and AUC(area under the curve of the Receiver Operating characteristics) instead of accuracy. This is because since the data is highly biased towards one of the class('no'), our classifier might decide to classify all the examples of the test set to be in the 'no' class, and we will still get a very high accuracy, but such a classifier would be of no use for a new data. However when we use F1 score and AUC we get the insights of the True Positives and False Positives through F1 score and The True Positive rate and False Positive rate through the ROC and AUC.

- Computer Language: The computer language used is Python.

Preprocessing

Reviewing the data set, out of the 19 features for every example 9 features are numeric and 10 features are categorical data. Also there are a lot of unknown values in the dataset , and all of those unknown values are in the categorical features. Also some features of the categorical type are ordered where as some of them are unordered. The preprocessing steps like removing the unknown values, defining the ordered and unordered categorical features and standardizing the numeric and ordered categorical data will be described below. We will be handling the unknown values in two ways, one is by assigning class wise means value to unknown values of ordered categorical variable and removing all the examples corresponding to unknown values of unordered categorical variable and second is through KNN imputation in which we consider 3 nearest neighbor without any unknown value to an example containing unknown values and impute values based on maximum occurrence of the values in other examples. It is also important to note that the order specified below is the order in which preprocessing is done.

- Toolbox used for parsing: **pandas**
- Toolbox used for KNN imputation: **fancyimpute**(KNN)
- Toolbox used for Standardizing: **sklearn**
- Toolbox for oversampling: **imblearn.over_sampling**(SMOTE)

A. Categorical features and Unknown values:

- The default feature has 803 unknown values and as a result of this the default feature is entirely discarded from the data.
- The ordered categorical features' levels and the values assigned to them are as follows:

Num Vals	job	education	Month	day_of_week
1	unemployed	illiterate	Mar	Mon
2	student	basic.4y	April	Tue
3	housemaid	basic.6y	May	Wed
4	retired	basic.9y	Jun	Thu
5	self-employed	high.school	Jul	Fri
6	admin.	professional.course	Aug	-
7	services	university.degree	Sep	-
8	technician	-	Oct	-
9	blue-collar	-	Nov	-
10	management	-	Dec	-
11	entrepreneur	-	-	-

These values assigned above to every level of an ordered categorical variable is in context with the social standing. For example (an entrepreneur has a higher social standing than an unemployed and is more likely to open a long term deposit account) and (a university degree graduate is more likely to open a bank account as compared to an illiterate). Also month and day of week can be considered ordered in context of their natural order i.e December cannot occur before November and Friday cannot occur before Thursday.

- After assigning values to ordered categorical features we take care of unknown values in 2 ways and consider both of them to be a different dataset.

    1. X_NA_removed : The data in which unknown values in ordered categorical variable are imputed as per class wise means of that particular feature and removing unknown values by deleting rows in which unknown values occur.

2. X_KNN: The data in which the unknown values are imputed by considering 3 nearest neighbors. In this data there is no under sampling.

- After doing the above process the unordered categorical features are converted to one hot vectors and substituted in place of the original unordered categorical features. The unordered categorical features are as follows:

  [marital, housing, loan, contact, poutcome]

B. Numeric features:

- The only processing on the numeric data to be done is on the feature 'pdays'. The value 999 in that feature means that the client was never contacted before, however the other values in that feature range from 0 to 9. Since 999 is a very high numeric number it might be bad for the classifier and hence 999 is changed to -1 in order to make sense of the values of the feature 'pdays'. It is to be noted that this value is changed in both the data mentioned above (X_NA_removed) and (X_KNN).

C. Standardizing:

- All the numeric and ordered categorical features are normalized. The unordered categorical data are kept as it is since it does not make any sense normalizing one hot vectors who only have 0 and 1 as their unique values.

D. Class Vector:

- Finally the class vector y which contains the responses of example ('yes', 'no') is modified so that 'no' is now a numeric 0 and 'yes' is a numeric 1.

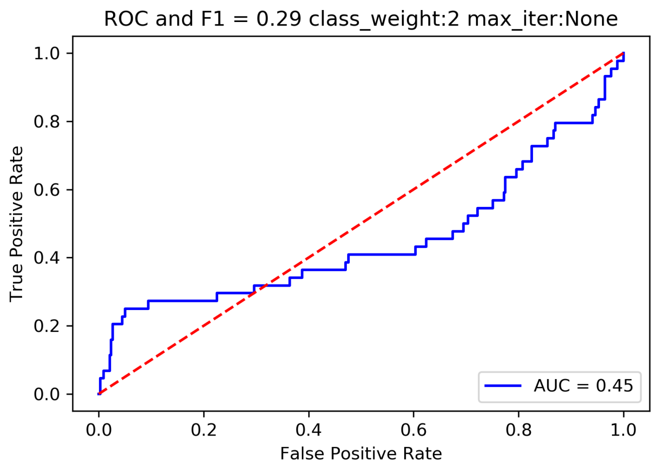
****

Baseline model:

The baseline model is created to compare it with other classifiers. The description is as follows:

1. Perceptron classifier.
2. No processing of unknown values and all rows corresponding to unknown values are removed.
3. All categorical variables are converted to one hot vectors.
4. No Standardizing is performed.
5. No dimensionality reduction is performed.
6. Class weight is kept to be 2 since the data is highly unbalanced.

Baseline F1: 0.29 **Baseline AUC: 0.45**



- As we can see the ROC is almost as bad as a random classifier and the f1 score is also very less.

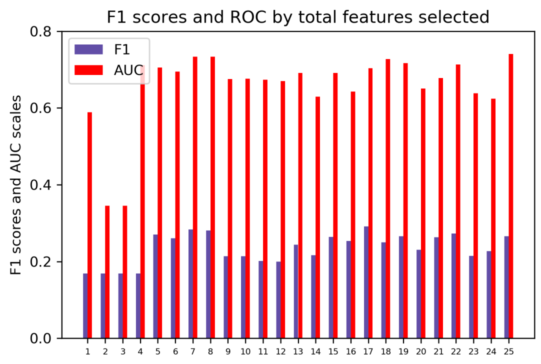
Feature Selection/Dimensionality Reduction

- **Technique:** The selection of features is done by a method called Recursive feature elimination(RFE) method. This method works by recursively considering lesser and lesser number of features depending on the weights assigned to the features by an external estimator. In our case I have used the SVM linear kernel estimator from sklearn to assign weights to the features.
- **Method of selecting best feature:** However since for every total number of feature that I select , those features will give me a different f1 score, and as a result I am iterating a loop starting from selecting just one feature and up-to all the features with the help of RFE technique, and at the same time using those features selected to cross validate over a classifier to generate a mean f1 score. Since I have 25 features after the initial preprocessing of data, the above mentioned algorithm will give me 25 f1 scores. I am selecting the features corresponding to the best f1 score and then using it for further training of data to predict the test accuracies and plotting the roc curve.
  The features selected and used for different classifiers will be as follows:

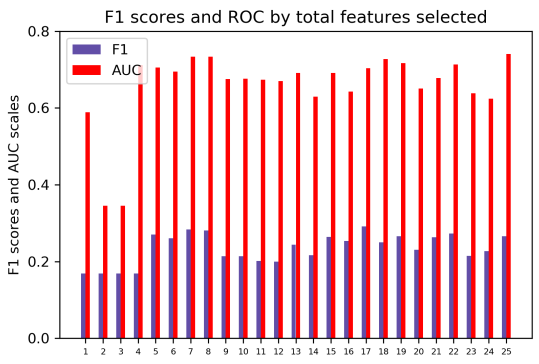
A. *Perceptron and SVM:*

- Shown below are the mean f1 scores for each selected total number of features using the above mentioned method for feature selection with a perceptron classifier for cross validating on both the data ,(i.e) the data with class wise means imputation(X_NA_removed) and with KNN imputation(X_KNN).

X_NA_removed



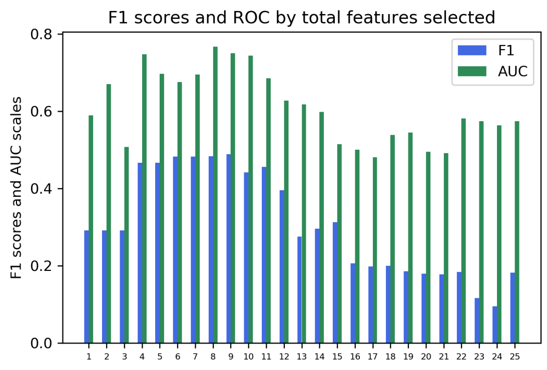
X_KNN



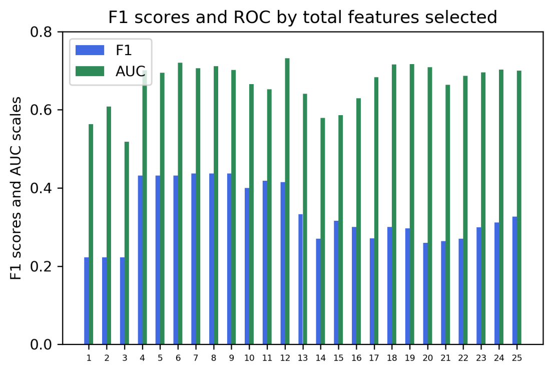
As we can see the f1 scores are almost as bad as the base line model hence we take another approach for selecting the best features. This also gives us an insight that the data is not linearly separable and using an SVM classifier with 'rbf ' kernel might help.

- Using the same method as above but changing the CV classifier to SVM gives us the following results.

X_NA_removed



X_KNN



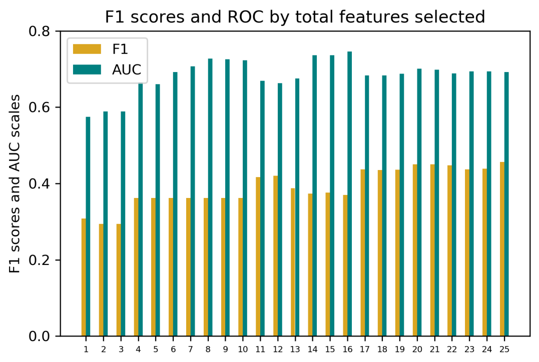
As we can the f1 scores are significantly higher and they come out to be the maximum when 9 of the highest ranking features are considered for both data (NA_removed and KNN). Those 9 features are selected as per the RFE technique and these are the features that we will be using for the SVM and perceptron classifier.

- The features selected for SVM and perceptron for both X_NA_removed and X_KNN are as follows: ***[month, euribor3m, nr.employed, loan_yes, contact_cellular, contact_telephone, poutcome_failure, poutcome_nonexistant, poutcome_success]***

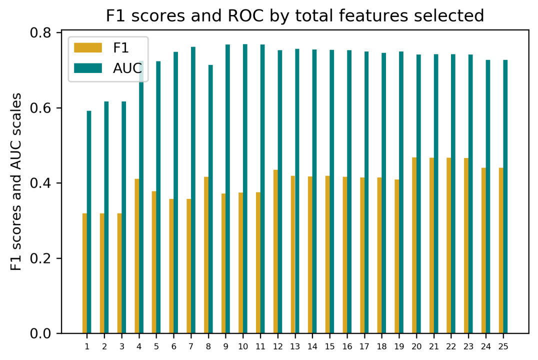
B. *Minimum risk Bayes*

- The above mentioned method for feature selection using RFE is used, the only difference is that instead of using the SVM classifier for cross validating we use our kernel density classifier. The mean f1 scores generated for both the data for every total number of feature is as follows:

X_NA_removed



X_KNN



- As we can see from the above graph for X_NA_removed the f1 score when all the features are selected is maximum and in the case of X_KNN the maximum f1 score occurs when 20 of the highest ranking features are selected. The features are as follows
  X_NA_removed : **All features are used**
  X_KNN : **All features except 'day_of_week','previous','marital_single','housing_no','housing_yes'**

Classifiers/DatasetUsage/Performance/Result/Interpretation/toolbox/libraries

A. *Perceptron:*

- Data: The perceptron classifier is run on both the data sets mentioned above i.e X_NA_removed and X_KNN in two ways.
  - i. The error estimation is done by cross validating and tuning the hyper parameters (class_weight) and maximum iteration) for an unbalanced dataset.
  - ii. First the testing set is separated and then the training set is oversampled using SMOTE and again error estimation is done by cross validating and tuning only the max_iter hyper parameter.

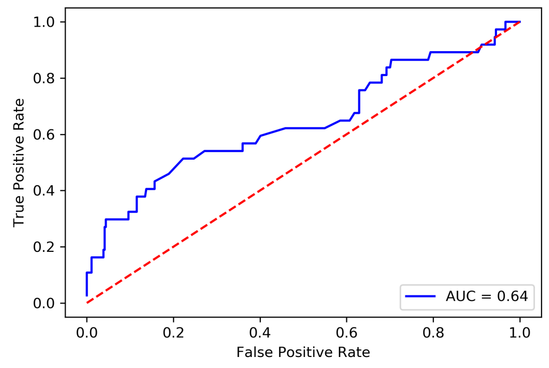
- Cross Validation: In the case of perceptron, for the two unbalanced data the cross validation takes care of tuning class_weight and the maximum iteration parameters while for the balanced dataset, maximum iteration is the only parameter that is tuned in cross validation. The tuned parameter are then used to calculate the f1 score and AUC of the test data.

- Toolboxes and Libraries:
  - Toolbox and library for cv: No toolbox used, Cross validation code was written by me.
  - Toolbox and library for Density Estimation: **sklearn**(Perceptron)

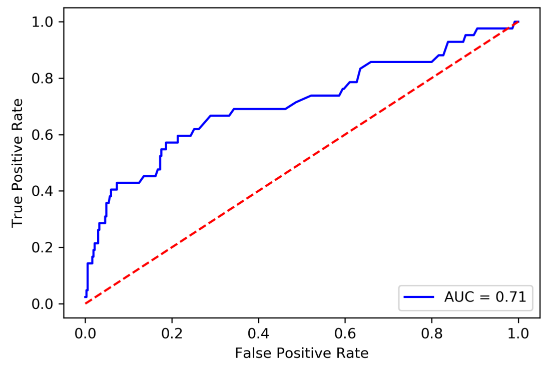
- Mean CV F1,CV stdev, test f1 and selected hyper parameters for the 4 tests conducted:

	Mean F1(CV)	St.dev	Test F1	AUC	**weight**	**Max_iter**
X_NA_removed_unbalanced	0.4067	0.0659	0.30	0.64	3	33
X_KNN_unbalanced	0.408	0.04	0.35	0.71	2	13
X_NA_removed_balanced	0.35	0.06	0.29	0.74	-	17
X_KNN_balanced	0.36	0.05	0.26	0.69	-	47

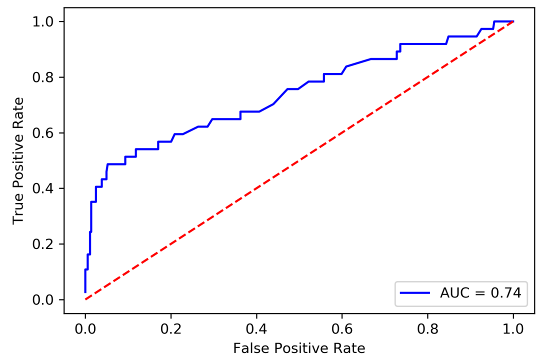
X_NA_removed_Unbalanced



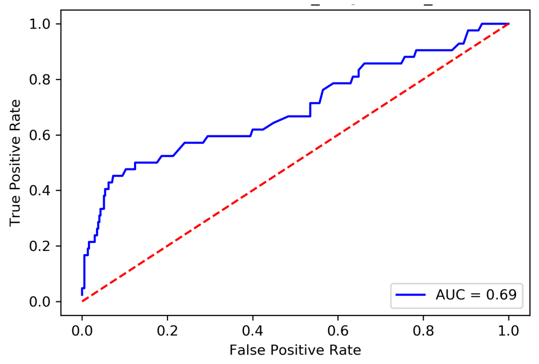
X_KNN_Unbalanced



X_NA_removed_balanced



X_KNN_balanced



- Interpretation: The f1 scores are more than the baseline model but they are almost very close to the baseline scores. However the AUC is large as compared to that of the baseline model. This is because our dataset is highly imbalanced and even though the AUC is high the f1 score remains low as f1 is a measure of both precision and recall. The high value of AUC might be because of large false positives that occur in the classification due to the skewness of the dataset. Possible reasons for this are that the data is not linearly separable and perceptron being a linear classifier is not able to classify properly.

B. ***SVM***:

- Data: The data used is the same as that of perceptron as the feature selected for both the classifiers are the same. It is to be noted that for unbalanced data the class weight given to class 'yes' is 8 since Class 'no' is approximately 8 times more in the dataset.

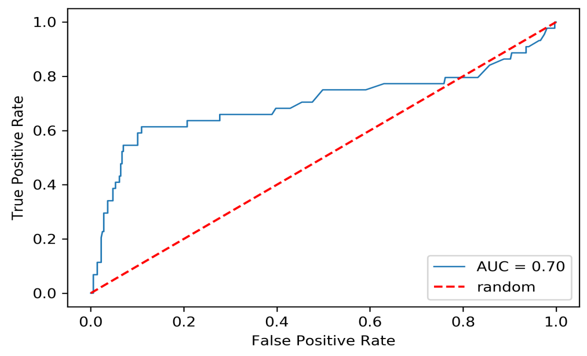
- Cross Validation: The parameters tuned during the cross validation of SVM are the C and gamma of 'rbf' kernel. After tuning the parameters the final parameters are used on the test data separated from the original data initially. It is to be noted that this tuning of parameters are done on two unbalanced data, with unknown values removed and KNN imputation, and also on the two balanced data, where the data are oversampled with the help of SMOTE. The details of which are given below.

- Toolboxes and Libraries:
    - ➢ Toolbox and library for cv: **sklearn**(GridsearchCV)
    - ➢ Toolbox and library for classification: **sklearn**(SVC)

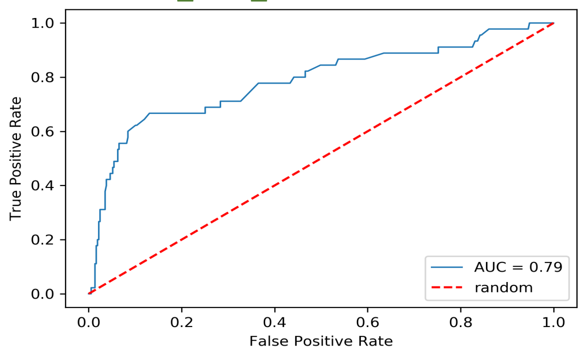
- Mean CV F1, test f1 and selected parameters for the 4 tests conducted:

	Mean F1(CV)	Test F1	AUC	**C**	**gamma**
X_NA_removed_unbalanced	0.53	0.51	0.70	0.464	2.154
X_KNN_unbalanced	0.54	0.49	0.79	0.8	1.7
X_NA_removed_balanced	0.70	0.67	0.65	215.44	1000
X_KNN_balanced	0.70	0.69	0.63	143.69	10.77

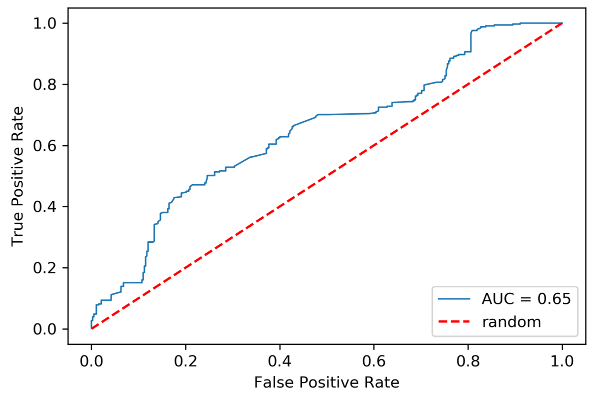
X_NA_removed_unbalanced



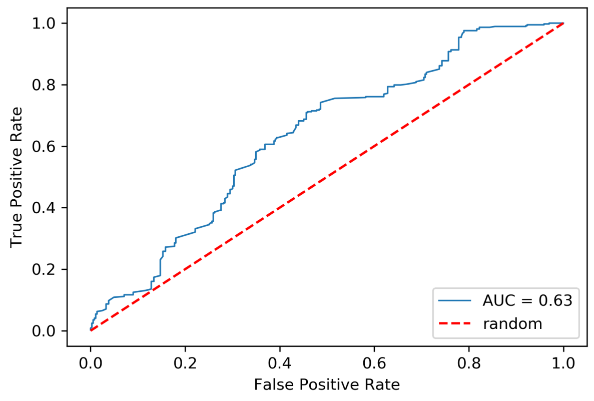
X_KNN_unblanced



X_NA_removed_balanced



X_KNN_balanced



- Interpretation: As mentioned earlier while discussing the results of perceptron, we concluded that the data is not linearly separable. Thus an SVM classifier with ‘rbf’ kernel would prove much better in such cases and this is evident from the table shown above. Even though the AUC is almost the same for both the classifiers there is a clear difference in the f1 scores. The f1 scores for unbalanced data is significantly higher that the perceptron and for the balanced dataset the f1 score is even higher. Thus we can say that SVM is a better classifier than perceptron for this dataset of bank marketing.

C. ***Minimum Risk Bayes Classifier***:

- Data: The minimum risk Bayes classifier considers only the unbalanced dataset and varies the loss function, and bandwidth of the kernel density estimator while cross validating.

- Density estimation: Class conditional Density estimation is done using unsupervised nearest neighbor learning with 'gaussian' kernel, using the KernelDensity() function of sklearn. The 'gaussian' kernel is used so that the class conditional probability density has a smooth curve.

- Loss Functions λ_{ij}: The loss of deciding class i when the actual class is j.

λ_{00}	λ_{01}	λ_{10}	λ_{11}
0	Variable	Variable	0

  - The parameters λ_{00} and λ_{11} are always kept to be zero as we consider that there is no loss when the classifier predicts 'no' when the actual class is 'no' and 'yes' when the actual class is 'yes'.
  - The parameters λ_{01} and λ_{10} are variable and chosen through cross validation.

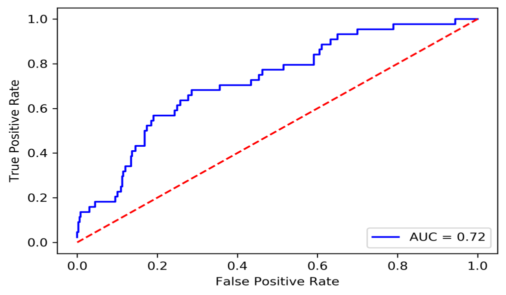
- Classification Rule:
  - If $(\lambda_{10} - \lambda_{00})P(\underline{X}|S_0)P(S_0) > (\lambda_{01} - \lambda_{11})P(\underline{X}|S_1)P(S_1)$ then ($\underline{X} \in S_0$ else $\underline{X} \in S_1$)
    Since we are considering λ_{00} and λ_{11} to be zero we can write the final equation for classification using minimum risk Bayes classifier as follows.
  - If $(\lambda_{10})P(\underline{X}|S_0)P(S_0) > (\lambda_{01})P(\underline{X}|S_1)P(S_1)$ then ($\underline{X} \in S_0$ else $\underline{X} \in S_1$)

- Cross Validation: The parameters that are tuned through cross validation are the bandwidth of gaussian kernel, and the loss functions λ_{01} and λ_{10}. The error estimation is performed on 10% of the data that is separated as test set initially all along maintaining the ratio of classes in the test set.
- Toolboxes and Libraries:
  - Toolbox and library for cv: **sklearn**(GridsearchCV)
  - Toolbox and library for Density Estimation: **sklearn**(KernelDensity)
  - The code for minimum risk Bayes classification is written by me.

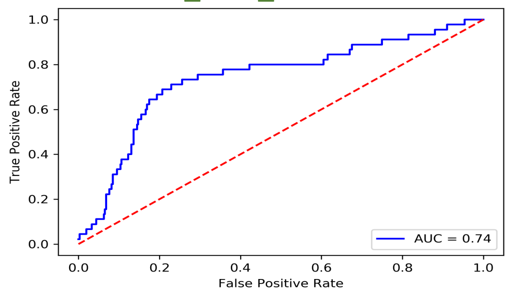
- Mean CV F1, test f1 and selected parameters for the 2 tests conducted:

	Mean F1(CV)	Test F1	AUC	λ_{10}	λ_{01}	**bandwidth**
X_NA_removed_unbalanced	0.539	0.53	0.72	1	7	2
X_KNN_unbalanced	0.52	0.47	0.74	1	8	3.8

X_NA_removed_unbalanced



X_KNN_unbalanced



- Interpretation: As we can see the Minimum risk Bayes classifier gives us results that are as good as the unbalanced data used for SVM. It should also be noted that the loss function λ_{01} is 7 and 8 for both the unbalanced unknown values removed data and the KNN imputed data respectively. Also from previous sections we know that the unbalanced data has the examples with 'no' class which are almost 8 times as compared to examples of 'yes' class. All this points to the fact that the loss function is giving an importance to the under-represented class which is in accordance with the ratio of it with the over represented class.

D. ***Neural Networks***:

- Data: The neural network classifier considers only the balanced data(NA_balanced and KNN_balanced), trains on it and makes the predictions on the previously separated test set.

- Cross Validation: The cross validation in case of neural network is done for tuning the total number of hidden layers and the total neurons in it, as well as for alpha(L2 regularization parameter) and the maximum number of iterations. These final tuned parameters are then used to find f1 score and AUC on the test set previously separated.

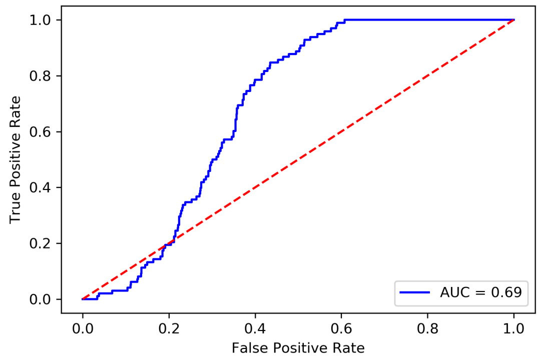
- Toolbox and libraries:
  - Toolbox for cv: **sklearn**(GridsearchCV)
  - Toolbox for classification: **sklearn**(MLPclassifier)

- Mean CV F1, test f1 and selected parameters for the 2 tests conducted

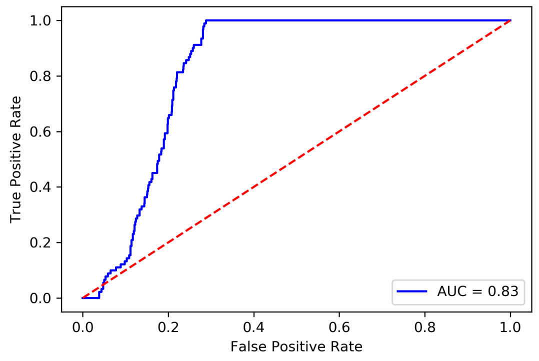
	Mean F1(CV)	Test F1	AUC	**alpha**	**Max_iter**	**Hidden_layers**
X_NA_removed_balanced	0.74	0.69	0.69	0.0005	70	(25,100,20,10,5)
X_KNN_balanced	0.76	0.72	0.83	0.0004	58	(25,100,40,20,10,5)

This means that the neural network that classified the test set after training on X_NA_removed_balanced has 5 hidden layers with 25,100,20,10 and 5 neuron units in it respectively.

X_NA_removed_balanced



X_KNN_balanced



- Interpretation: As we can see from the above graph and the test f1 scores and AUC, neural network is as good as SVM with ‘rbf’ kernel for classifying the data points of the current data. We can also say that it is a fraction better than SVM. The reason I can think of for this is that the data is not linearly separable, and hence linear classifiers like perceptron failed to give us a good result. Bayes minimum risk classifier gave us a fairly good result by estimating the class conditional density. However the best results were given to us by nonlinear classifiers like SVM and neural networks. High f1 score and AUC for the neural network also tell us that the amount of true positives in the classification is more as compared to perceptron and Minimum risk Bayes classifier.

Comparison and Conclusion:

➢ Comparison:

Unbalanced data

	F1_NA	F1_KNN	AUC_NA	AUC_KNN
Perceptron	0.30	0.35	0.64	0.71
SVM (rbf kernel)	0.51	0.49	0.70	0.79
Min risk Bayes	0.53	0.47	0.72	0.74
Baseline	**0.29**	**0.29**	**0.45**	**0.45**

balanced data

	F1_NA	F1_KNN	AUC_NA	AUC_KNN
Perceptron	0.29	0.26	0.74	0.69
SVM (rbf kernel)	0.67	0.69	0.65	0.63
Neural Network	0.69	0.72	0.69	0.72
Baseline	**0.29**	**0.29**	**0.45**	**0.45**

- Even though the perceptron model with cross validation gives good AUC values, its f1 score is still very less and almost equal to the baseline model for both the balanced and unbalanced data. Hence we can conclude that our data is not linearly separable and linear classifiers like perceptron are not good for this scenario. Thus we concentrate on non-linear classifiers and probabilistic models.
- As seen from the above table for unbalanced data, SVM with rbf kernel which is a non-linear classifier and minimum risk Bayes classifier give us the best result and their AUC is also pretty high. The reason for this is that since our data is not linearly separable, non-linear classifiers and probabilistic density estimation techniques fit the data in a better way and give us better results.
- Now if we look at the final stats. of balanced data, SVM and Neural Networks give us the best result. Not only is their f1 score significantly high as compared to other classifiers but the overall f1 score is a good value which tells us that precision of the classifier is good. It also gives us an insight that the roc curve has a higher value not because of higher false positive rates but because of higher True positive rates.
- We can also see that Neural networks has slightly better results as compared to SVM.

➢ Conclusion: Thus we can conclude that if we want to make predictions of future clients who would like to open a long term deposit, more accurately, then our best bet is to take the following steps:

- Take all the steps of preprocessing the data mentioned above, but this time remove all the unknown values with the help of KNN imputation.
- Separate 10 % of the data as test data from the original preprocessed data, all the while maintaining the ratio of the two classes 'yes' and 'no' in training and testing data.
- Oversample the Training data using the SMOTE to increase the ratio of under-represented class.
- Reduce dimensionality by selecting the best features through methods discussed above.
- Train SVM or Neural Network classifier on the training data and then use it for future predictions.

****
THE END