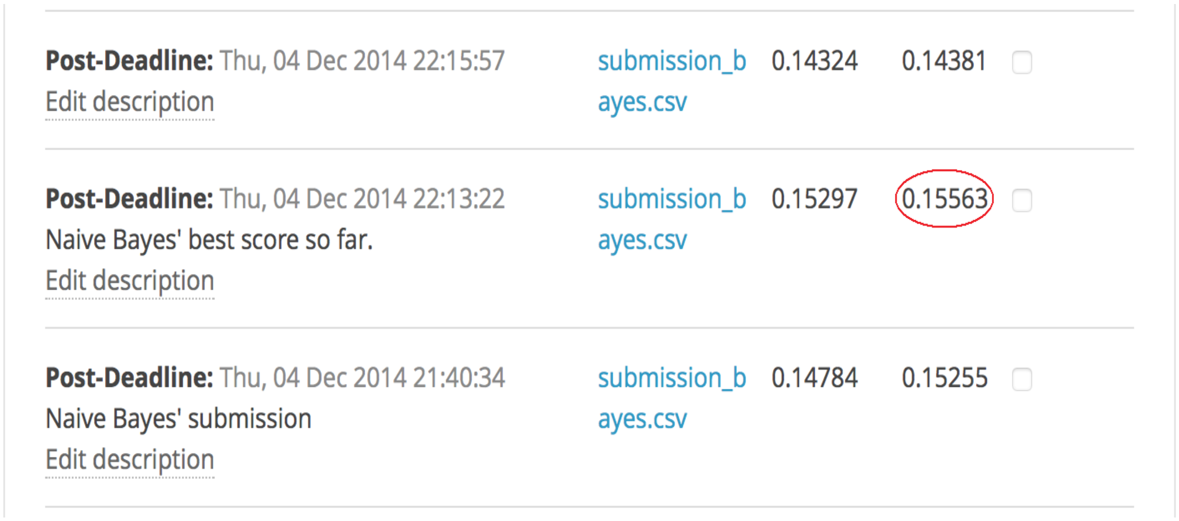
Prakyath Jaladhar (jaladha2) | Jatin Ganhotra (ganhotr2)

Project Report – CS 412 Fall 2014

Team Information:

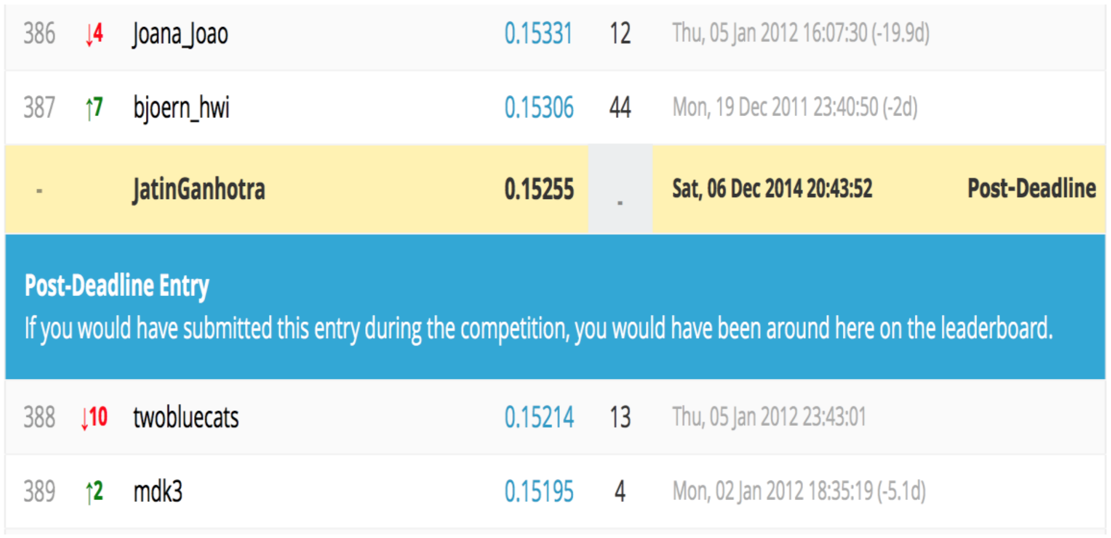
1. Prakyath Jaladhar (jaladha2)
2. Jatin Ganhotra (ganhotr2)

Best Score Screenshot:



Post-Deadline: Thu, 04 Dec 2014 22:15:57 Edit description	submission_bayes.csv	0.14324	0.14381	☐
Post-Deadline: Thu, 04 Dec 2014 22:13:22 Naive Bayes' best score so far. Edit description	submission_bayes.csv	0.15297	0.15563	☐
Post-Deadline: Thu, 04 Dec 2014 21:40:34 Naive Bayes' submission Edit description	submission_bayes.csv	0.14784	0.15255	☐

We got the best score of **0.15563** when we were trying out different combinations of features, but we missed to take a screenshot of the score. The above screenshot is from the list of all submissions. Currently, we get the maximum score of **0.15255**



386	↓4	Joana_Joao	0.15331	12	Thu, 05 Jan 2012 16:07:30 (-19.9d)	
387	↑7	bjoern_hwi	0.15306	44	Mon, 19 Dec 2011 23:40:50 (-2d)	
-		**JatinGanhotra**	**0.15255**	-	**Sat, 06 Dec 2014 20:43:52**	**Post-Deadline**

Post-Deadline Entry
If you would have submitted this entry during the competition, you would have been around here on the leaderboard.

388	↓10	twobluecats	0.15214	13	Thu, 05 Jan 2012 23:43:01
389	↑2	mdk3	0.15195	4	Mon, 02 Jan 2012 18:35:19 (-5.1d)

Project Description

The project is to predict if a car purchased at an auction is a lemon or not. The data provided has the following properties:

1. Each tuple/ instance has a lot of features
2. The data is highly skewed, i.e. contains a lot of tuples which have class label as 0
3. There are NULL values in the data for many attributes

These issues are addressed by pre-processing the data according to the classification techniques applied. The details for data preprocessing are explained below.

Data pre-processing / Feature engineering

Data pre-processing was an important step. The data in consideration was noisy. It contained many missing values and NULL values. Especially AUCGUART and PRIMEUINT attributes had almost 85% NULL values.

We decided to implement k-NN and Naïve Bayes classification algorithms. For k-NN to work best, we had to convert attributes to numeric data. For this to be done, we considered dropping instances with missing values or replacing missing values with the mean.

As we decided to use LIBSVM format data for the Naive Bayes classifier, so we replaced all the missing values by '-1' as it would also be considered as a value of an attribute in naive bayes. This also helped us in case of preprocessing test data because test data also had missing values.

The categorical data was also replaced by the index of the value. The numeric attributes in the data had various values, to tackle this we chose to smooth the data. There were 14 numeric attributes:

VehYear *VehicleAge*
VehOdo *MMRAcquisitionAuctionAveragePrice*
MMRAcquisitionAuctionCleanPrice *MMRAcquisitionRetailAveragePrice*
MMRAcquisitonRetailCleanPrice *MMRCurrentAuctionAveragePrice*
MMRCurrentAuctionCleanPrice *MMRCurrentRetailAveragePrice*
MRCurrentRetailCleanPrice *VehBCost*
WarrantyCost *PurchDate*

The next step was to discretize the numeric variables; we decided to do equal frequency binning and discretized those variables which had more than 25 unique values. Equal frequency binning splits the continuous variables into equally sized groups based on their values. We did this by using *histc* function that is available in *MATLAB.* We first read the 'training.csv' and 'test.csv' file into the memory and saved the data for future use. Later we used an external program that helped us to store the preprocessed data back to CSV format.

The Naive Bayes classifier takes LIBSVM files as input. So after preprocessing, we converted pre-processed CSV files to LIBSVM format and made sure there were no missing values or NULL values. In order to easily parse the data we inserted 'IsBadBuy' column with dummy values in the pre-processed test CSV file before converting it to the LIBSVM format. We used these LIBSVM files as input for the Naïve Bayes classifier.

For k-NN, we used the same pre-processed data but tried different measures for distance. We did min-max normalization on this data. Initially, we tried with Euclidean distance for numeric attributes and Hamming distance for categorical attributes. But, later after pre-processing, we used the Euclidean distance for all attributes. We tried to find an optimal distance measure, and based our experiments around it to optimize the accuracy for our k-NN classifier.

To convert the given training and test CSV files to libsvm format, we used the python code provided by Zygmunt Zając at the link below:
https://github.com/zygmuntz/phraug2

Classification techniques

We applied two techniques – ***Naïve Bayes*** and ***k-nearest-neighbors*** for classification. These techniques are explained briefly as follows:

Naïve Bayes:

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features. Given a class variable y and a dependent feature vector x_1 through x_n, Bayes' theorem states the following relationship:

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) P(x_1, \dots x_n \mid y)}{P(x_1, \dots, x_n)}$$

Using the naive independence assumption that,

$$P(x_i | y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i | y),$$

for all i, this relationship is simplified to

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

Since $P(x_1, \dots, x_n)$ is constant given the input, we can use the following classification rule:

$$P(y \mid x_1, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

we can use Maximum APosteriori (MAP) estimation to estimate $P(y)$ and $P(x_i \mid y)$; the former is then the relative frequency of class y in the training set.

k-Nearest Neighbors (k-NN):

The ***k*-Nearest Neighbors algorithm** (or ***k*-NN** for short) is a non-parametric method used for classification.

The input consists of the k closest training examples in the feature space. The output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

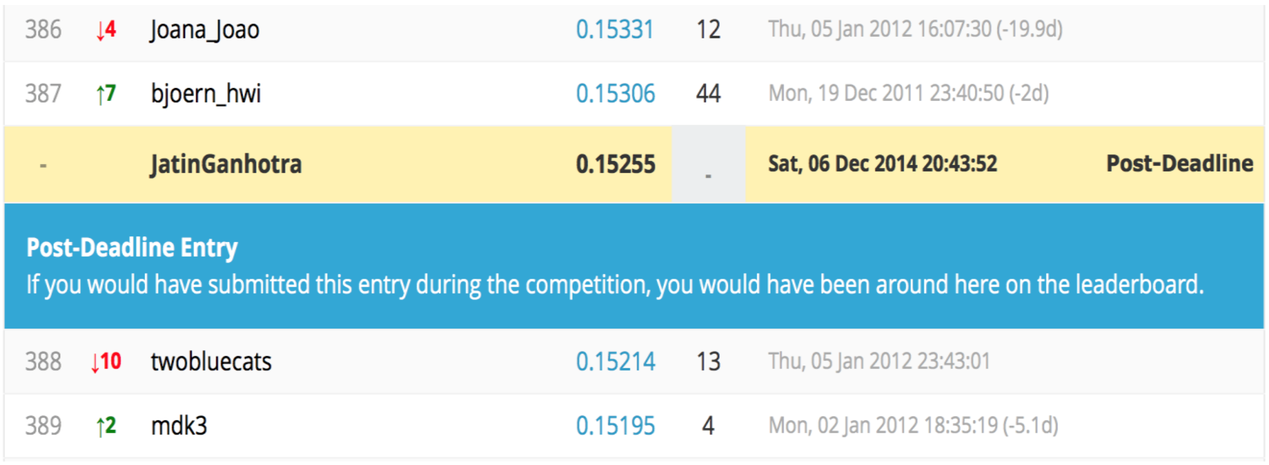
k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification.

For classification, k-NN can be useful to weight the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. The neighbors are taken from a set of objects for which the class (for k-NN classification) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

Screen-shots of both algorithms:

After trying out various pre-processing techniques, we got different results. The screenshots for our both approaches i.e. Naïve Bayes and k-NN are as follows:

Naïve Bayes screenshot:

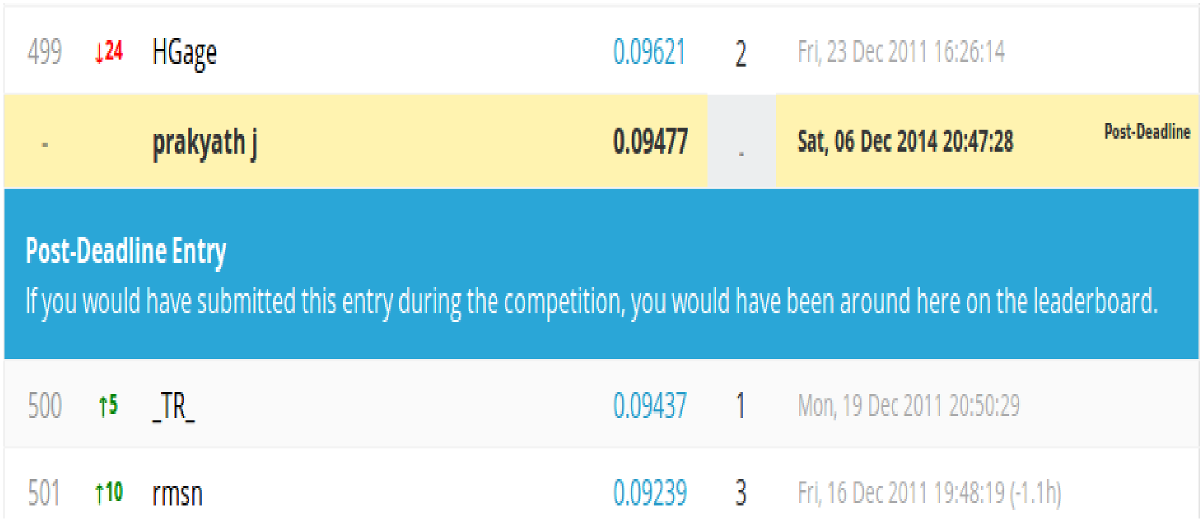


386	↓4	Joana_Joao	0.15331	12	Thu, 05 Jan 2012 16:07:30 (-19.9d)	
387	↑7	bjoern_hwi	0.15306	44	Mon, 19 Dec 2011 23:40:50 (-2d)	
-		**JatinGanhotra**	**0.15255**	-	**Sat, 06 Dec 2014 20:43:52**	**Post-Deadline**

Post-Deadline Entry
If you would have submitted this entry during the competition, you would have been around here on the leaderboard.

388	↓10	twobluecats	0.15214	13	Thu, 05 Jan 2012 23:43:01
389	↑2	mdk3	0.15195	4	Mon, 02 Jan 2012 18:35:19 (-5.1d)

k-NN result screenshot:



499	↓24	HGage	0.09621	2	Fri, 23 Dec 2011 16:26:14	
-		**prakyath j**	**0.09477**	-	**Sat, 06 Dec 2014 20:47:28**	**Post-Deadline**

Post-Deadline Entry
If you would have submitted this entry during the competition, you would have been around here on the leaderboard.

500	↑5	_TR_	0.09437	1	Mon, 19 Dec 2011 20:50:29
501	↑10	rmsn	0.09239	3	Fri, 16 Dec 2011 19:48:19 (-1.1h)

Parameter settings

1. For Naïve Bayes, we have one parameter setting, which is the likelihood of a new tuple for a class. Since, we are doing binary classification, we have 2 class labels and each label should have equal likelihood probability, i.e. 0.5 each. If we calculate the class probabilities from the training data, then these numbers are very skewed because the training data is skewed as discussed before in the pre-processing step.
2. For k-NN, the value of k neighbors is an important parameter, especially when the training data is skewed. We tried different values for k and got different results for our experiments.

Distribution of work

Staring from the beginning, we planned to meet once a week and discuss on the progress made. We were always actively discussing pre-processing strategies and trying out different experiments. For classification strategies, we would discuss our approach, try it and re-visit our approach for better accuracy.

However, we implemented our own algorithms separately. Once we had the initial results for our approaches, i.e. Naïve Bayes and k-NN, we decided to stop continuing work on k-NN and tried collectively to improve the score for Naïve Bayes.

NOTE: The details about the source code, data and execution of the source code are explained in the README.txt file.