CS 229
Stay Alert! The Ford Challenge

Louis Fourrier
Fabien Gaie
Thomas Rolf

1. Problem description

a. Goal

Our final project is a recent Kaggle competition submitted by Ford whose goal is to design a binary classifier to detect whether a driver is alert or not. Driving while being distracted, fatigued or drowsy may lead to accidents. Solving this problem may eventually save lives and is thus a great application of machine learning!

b. Dataset Overview

Ford provided a combination of vehicular, environmental and driver physiological data acquired in real-time while driving. These different features do not have any name indicating what they represent, which adds to the complexity of the problem since we cannot rely on physical intuition.

The dataset consists of a number of "trials", each one representing about 2 minutes of sequential data that are recorded every 100 ms during a driving session. The trials are samples from some 500 drivers and 30 different features are recorded. The overall dataset is a 600,000x30 matrix with the associated labels.

2. Methodology

a. Data Analysis

We started by plotting the probability density functions of the features in both alert and non-alert cases to quickly get a grasp of the impact and the different correlations. We completed this first analysis with some useful metrics:

- mean and variance of each feature distribution
- raw estimation of how far from a Gaussian the distribution is (based on kurtosis and skewness)
- mutual information between each feature and the output labels

In order to run our algorithms we split our data into a training (~75%) and a test set (~25%). It is worth noticing that the data provided are unbalanced since there are more positive labels than negative labels. This is an important point to note since it often affects the learning algorithm behavior.

b. Generation of new features

Need for new features:
The first attempts we made relied on the 30 features provided by the original dataset and our algorithms performed very poorly. Thus we decided to generate new features for algorithms such as Naive Bayes and Logistic regression, which do not handle a Kernel Trick easily.

Method to generate new features:
We generated some additional features by combining existing features into new ones and making some mathematical operations on existing features:

- Added the inverse, the square and the cube of every features (1/xi, xi2, xi3)

- Added all the combination of 2 columns ($x_i * x_j$)
- Generated time intervals variable (mainly $x_t - x_{t-100}$ and $x_t - x_{t-10}$).
- Suppressed 3 original features that proved to be useless since they remain constant for every row.

This resulted in a total of approximately 600 features from an initial 30 feature dataset. It enabled us to capture more patterns when we implemented our algorithms on this enhanced set of features.

Shuffling our data or not:

We considered randomly shuffling our dataset at first. However this proved to be dangerous for our model because it resulted in splitting data from the same user in both the training and test sets, thus artificially improving the accuracy and AUC results. The time sequencing of the data was also lost by this shuffling.

We finally decided to split our users into a test and a training set first, then compute the time varying features and finally shuffle data before running our algorithms.

c. Evaluation criteria

In order to assess our algorithms we mainly used two types of evaluation criteria:

- The accuracy of our algorithm (equivalently the training and testing errors).
- The Area Under the Curve (AUC), computed as the area under the ROC (Receiver operating characteristic) curve. This criterion is insensitive to unbalanced dataset and it does not require defining a specific threshold for the decision boundary. This metric is closely related to the true positive rate (TPR) and false positive rate (FPR).

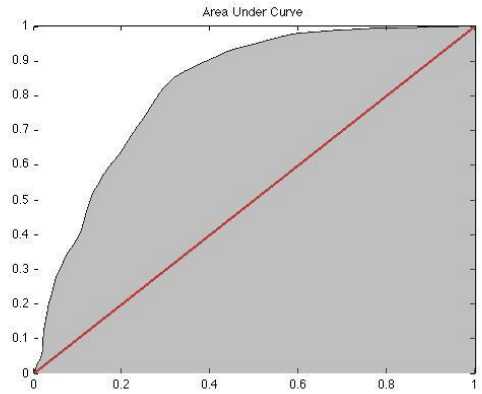


Figure 1: ROC curve obtained by logistic regression. The Area Under the Curve is 0.776.

Throughout our project, we ran algorithms using both evaluation criteria. We mainly present results using the accuracy since it gave us satisfying overall results and it allowed us to compare the performances of all our algorithms. We kept track of the AUC as a control metric and both the accuracy and the AUC improved as our algorithms got better.

3. Feature selection:

Seeing the initial results and after generating new features, we had to filter the best features out to reduce the noise and implement more efficient algorithms.

a. Mutual information

Our first method to select features was to use mutual information on the initial features. The five best features were E_5, E_9, V_5, E_3 and V_{10}.

We were surprised by the fact that none of the physiological features was selected so we decided to implement forward searches on different algorithms.

b. Forward search

We performed two different forward search procedures maximizing our two different evaluation criteria - the accuracy and the

AUC. The selected features varied with the algorithm and the evaluation criteria.

	Allowed features (x_i, x_i^2, x_i^3)	**Allowed features** $(x_i, x_i x_j)$
Naive Bayes	-	$V_{11}, E_9V_{11}, P_5P_6, P_1E_5, E_1^2$
Logistic Regression	$V_1, P_7^3, E_{10}^2, V_8, V_6$	$P_6V_1, E_3E_9, E_{10}^2, P_7E_5, P_5V_{11}$
SVM	$E_9, P_1, V_1^2, P_7^2, E_{10}^2$	$E_7E_9, E_9V_1, E_8V_2, E_{11}V_5, E_4V_4$

Figure 2: Five best features obtained from forward search in Naive Bayes, Logistic Regression and SVM algorithms.

The selected best features by the forward search procedures varied with the algorithm so that there is no ultimate feature set. Note that some features that we already captured with the Mutual Information like E_9 and E_5 appear several times in the table. When checking against the distribution of such features, we can see that they are indeed helpful separating the data into both labels.

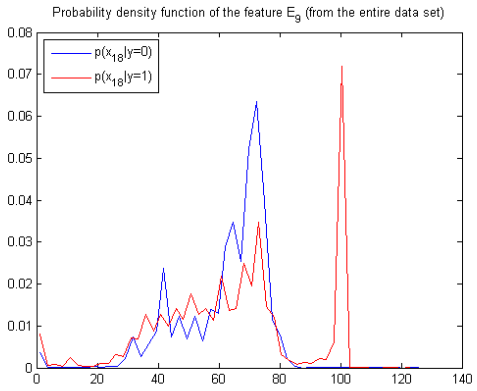


Figure 3: Plot of the distribution of feature E_9 in both alert and non-alert cases.

4. Learning Algorithms

a. Naïve Bayes

We implemented Naive Bayes algorithms with two different underlying models assuming that features were conditionally independent given the alertness of the driver.

Multinomial Naïve Bayes with Laplace smoothing:

We discretized all the input data by dividing each feature into a predetermined number of buckets before computing the multinomial probabilities with Laplace smoothing. The number of buckets had a big impact on the performance of the algorithm with a maximum around 150 buckets. Too many buckets proved to be inefficient since it could not capture the distribution properly and led to random probabilities.

Naive Bayes with Gaussian Distribution fitting:

We modeled each distribution as a Gaussian distribution which is a strong assumption for some features but gave surprisingly good results thanks to the forward search procedure.

These good results can be explained by the fact that some of the feature distributions are close to Gaussian distributions.

b. Logistic Regression

To begin with, we implemented logistic regression algorithms with both batch and stochastic gradient descent. While the batch gradient descent always converged, the stochastic one sometimes did not. We then leveraged Matlab optimized version of the logistic regression to have faster algorithms, after insuring that the results were consistent with our own algorithms results.

The logistic regression model enabled us to easily compute the AUC since we could output the probability of the label being 1 or 0 and sample the ROC curve by varying the decision threshold. We used this method to optimize on the AUC for one of our Forward Search experiment.

c. SVM

We implemented a SVM algorithm with both linear and nonlinear Kernels using liblinear and libsvm libraries for Matlab.

First, the SVM with linear Kernel applied to the 30 initial features gave a coherent accuracy of 64%. In order to work in a higher dimensional space, we implemented a SVM with Gaussian Kernel but it did not converge in a reasonable time. We then tried to implement the linear Kernel on the enhanced dataset with our additional features but faced the same issue. Therefore, we decided to implement a forward search to limit the number of features and prevent the algorithm to diverge. It turned out that choosing more than 10 features did not provide significant improvements.

We maximized the accuracy while varying the SVM parameter C between 0.1 and 100. We noticed that C had little impact on the SVM accuracy and finally let C be 1.

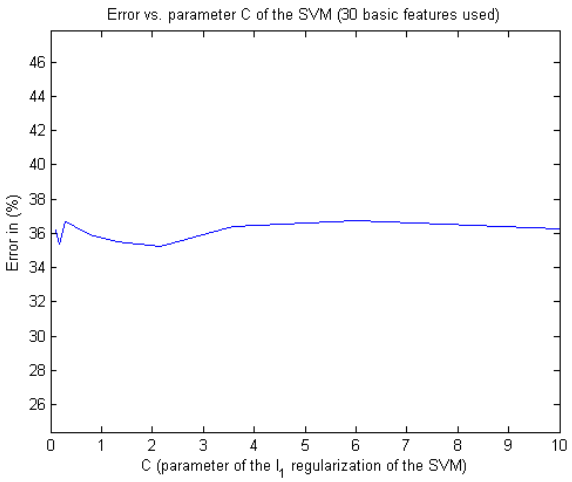


Figure 4: Influence of the parameter C of the SVM on the error.

d. Results

Results with basic 30 features:

Our first step was to run our three algorithms on the initial dataset of 30 features.

	Accuracy	TPR	FPR	AUC
Multinomial Naïve Bayes	53.1%	15.7%	9.2%	0.596
Gaussian Naïve Bayes	63.5%	49.0%	21.8%	0.759
Logistic Regression	73.1%	99.7%	74.2%	0.772
SVM	63.9%	*N/A*	*N/A*	*N/A*

Figure 5: Performances of algorithm on initial dataset.

These insufficient results were the reason for both the generation of additional features to capture better correlations and the selection of best features among this enhanced dataset.

Results with Mutual Information Features:

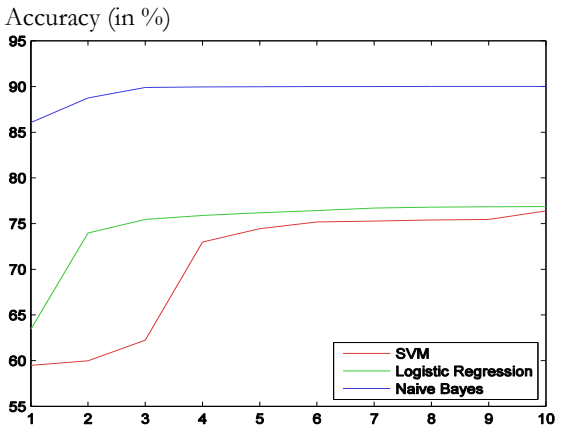
Selected features	Accuracy	AUC
$E_5, E_9, V_5, E_3, V_{10}$	56% - 60% depending on the algorithm	0.620

Figure 6: Performances of algorithms applied to the five best features from the Mutual Information.

The performances displayed above were a good start given the small number of features but encouraged us to implement forward search algorithms.

Results with best selected features from Forward Search:

The forward search procedures we used with each algorithm gave us the best results with the features described in the previous section.



	Accuracy	TPR	FPR	AUC
Naïve Bayes	90%	99.7%	19.7%	0.844
Logistic Regression	76.9%	93.6%	40.0%	0.776
SVM (linear Kernel)	76.4%	97.9%	45.3%	N/A

Figure 7: Accuracy evolution during the forward search (plot) and final performances of the algorithms (table).

Out of the algorithm we implemented, the Naïve Bayes with Gaussian Distribution fitting is clearly the most accurate one. The Naïve Bayes and the SVM have lower performances. The unsatisfying result of the SVM is due to the only linear Kernel.

From an initial dataset of 30 features, we created around 600 features before selecting only ten of them while significantly increasing the accuracy. A few features only therefore provided a good predictive power.

5. Conclusion

The best features strongly depend on the algorithm used to predict the alertness of the driver. The set of selected features is always a combination of vehicular, physiological and environmental features, which seems reasonable to capture the maximum information about the driving situation.

Our best results were obtained with a Naïve Bayes algorithm and such a set of features. To further improve these results, we are considering a Random Tree Forest algorithm.