Porto Seguro's Safe Driver Prediction
Project Status Report

Hanlin He, Mingze Xu, Su Yang, Tao Wang
Department of Computer Science, The Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
Email: {hxh160630, mxx160530, sxy161730, txw162630}@utdallas.edu

***Abstract*—This is the project status report of Porto Seguro's Safe Driver Prediction. Some of the content would be part of the final report.**

I. Introduction

Machine learning is emerging in the insurance industry and is being applied across multiple areas including the interpretation of data, business operations and driver safety. One key application is claim prediction. Inaccuracies in car insurance company's claim predictions raise the cost of insurance for good drivers and reduce the price for bad ones. A more accurate prediction will allow insurers to further tailor their prices, and hopefully make auto insurance coverage more accessible to more drivers.

In this report, we based on Kaggle's Featured Prediction Competition *Porto Seguro's Safe Driver Prediction* [1], conducted several experiments, compared different approaches' effectiveness to tackle the claim prediction problem, including *logistic regression*, *tensorflow estimator (neural network)* and *random forest*.

The organization of the following report is as follows. In section II, we will formally define the problem to solve and discuss the theoretical principle of the algorithm we used. Then we will analyze the data feature and our method of feature engineering in section III. After that, the experimental results are shown and analyzed in section IV. Finally, we will discuss related works and conclude the report.

II. Problem Definition and Algorithm

A. Task Definition

A machine learning problem is defined as to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by

P, improves with experience E [13]. The claim prediction problem can be defined as follow:

- E previous year's policy holders' information and whether or not a claim was filed for that policy holder.
- T predicting the possibility that an auto insurance policy holder will file an insurance claim next year.
- P the accuracies and Gini Coefficient was used to measure the effectiveness of models.

B. Algorithm Definition

We used logistic regression, neural network in our model building. In this section, we will discuss the theoretical foundation of these algorithms.

1) Logistic Regression: Logistic Regression is an approach to learning functions of the form $f : X \to Y$ [12] or in our case $P(Y|X)$ where Y is discrete-valued, and $X = \langle X_1 ... X_n \rangle$ is any vector containing discrete and continuous variables. The parametric model assumed by Logistic Regression in the case where Y is boolean is:

$$P(Y = 1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)}$$

$$P(Y = 0|X) = \frac{\exp(w_0 + \sum_{i=1}^{n} w_i X_i)}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)}$$

One reasonable approach to training Logistic Regression is to choose parameter values that maximize the conditional data likelihood. We also used regularization to reduce the overfitting problem. The penalized log likelihood function is as followed:

$$W \leftarrow \arg\max_{W} \sum_{l} \ln P(Y^l|X^l, W) - \frac{\lambda}{2}||W||^2$$

where the last term is a penalty proportional to the squared magnitude of W.

In general, the algorithm used gradient ascent to repeatedly update the weights in the direction of the gradient, on each iteration changing every weight w_i, beginning with initial weights of zero, according to:

$$w_i \leftarrow w_i + \eta \sum_{l} X_i^l (Y^l - \hat{P}(Y^l = 1|X^l, W)) - \eta \lambda w_i$$

where η is a small constant which determines the step size. The actual implementation of scikit learn library includes multiple solvers, such as Stochastic Average Gradient (SAG) descent, SAGA and Broyden-Fletcher-Goldfarb-Shanno (LBFGS).

2) Neural Network: Artificial neural networks (ANNs) [4], a form of connectionism, are computing systems inspired by the biological neural networks that constitute animal brains. Such systems learn (progressively improve performance) to do tasks by considering examples, generally without task-specific programming. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the analytic results to identify cats

in other images. They have found most use in applications difficult to express in a traditional computer algorithm using rule-based programming.

III. Feature Analysis and Engineering

The data comes in the traditional Kaggle form of one training and test file each: `train.csv` and `test.csv`. Each row corresponds to a specific policy holder and the columns describe their features. The target variable is named `target` here and it indicates whether this policy holder made an insurance claim in the past.

A. Data Overview

In total, there are 595212 training data instances and 892816 testing data instance. Since it's a competition, testing data had no `target` label. We cannot test our model accuracy based on testing data. In stead, we would use k-fold and cross validation on the training data to evaluate our model, which is discussed in more detail in section IV.

In the train and test data, features that belong to similar groupings are tagged as such in the feature names (e.g., `ind`, `reg`, `car`, `calc`). In addition, feature names include the postfix `bin` to indicate binary features and `cat` to indicate categorical features. Features without these designations are either continuous or ordinal. Feature count in each category and type are shown in table I.

TABLE I
Feature Counts in Each Category and Type

Type	Binary	Categorical	Numeric	Total
`ind`	11	3	4	18
`reg`	0	0	3	3
`car`	0	11	5	16
`calc`	6	0	14	20

Although feature's categories are provided, the meaning of each feature remains unknown. Some participants have guessed the meaning of several features, for example the *binary* variables `ps_ind_06-10` are *one-hot encoded*, and `ps_car_13` might be car's mileage. Our experiment show that some of the assumptions are plausible, however, we did not rely on these information to build our model.

B. Distribution Analysis

Features' distribution for `ind`, `reg`, `car`, `calc` are shown as histograms in fig. 1, fig. 2, fig. 3 and fig. 4 respectfully.

From histograms, we may conclude the following:

- Some binary features are dominated by `True` (`False`), for example `ps_ind_06-10`.
- Some features' distribution are fairly even, for example `ps_calc_01`, `ps_calc_02` and `ps_calc_03`.

It might seem plausible to discard these features in training phase. But the deletion

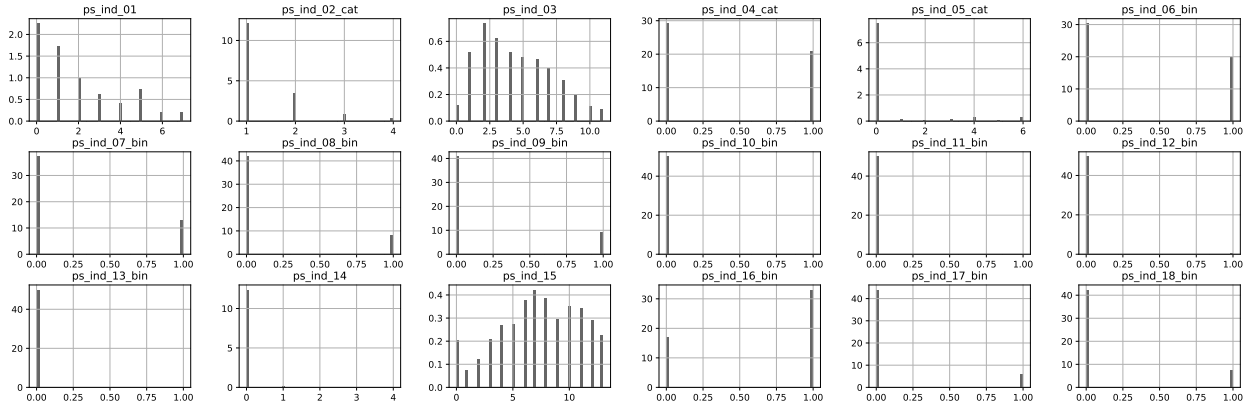

Fig. 1. Histogram for the `ind` Attributes.

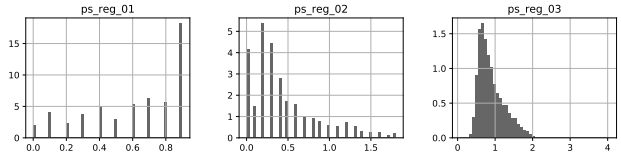

Fig. 2. Histogram for the `reg` Attributes.

requires more careful consideration. As mentioned above, `ps_ind_06-10` might be *one-hot encoded*, thus, all these four features would be naturally exclusive of each other.

C. Correlation Analysis

Features' Correlation within each category, i.e., `ind`, `reg`, `car`, `calc` are shown as histograms in fig. 5, fig. 6, fig. 7 and fig. 8 respectfully.

From correlation plot, we can conclude that strong correlation did not exist. It it not possible to conduct any dimension reduction based on correlations between features.

D. Missing Value Mechanism and Data Imputation

Values of -1 indicate that the feature was missing from the observation. Missing values count in each feature is shown in table II:

There are several approaches to handle missing values [14]:

- Deletion methods: cases with missing values are discarded, the analyses are restricted to cases that have complete data.
- Single imputation methods: imputes (i.e., *fills in*) the missing data with seemingly suitable replacement values. Mean, median and mode are the most popular averaging techniques to use [10].
- Multiple imputation methods: creates several copies of the data set, each containing different imputed values. These method has greater performance but is harder to implemented.

From the table we can see that, most fea-

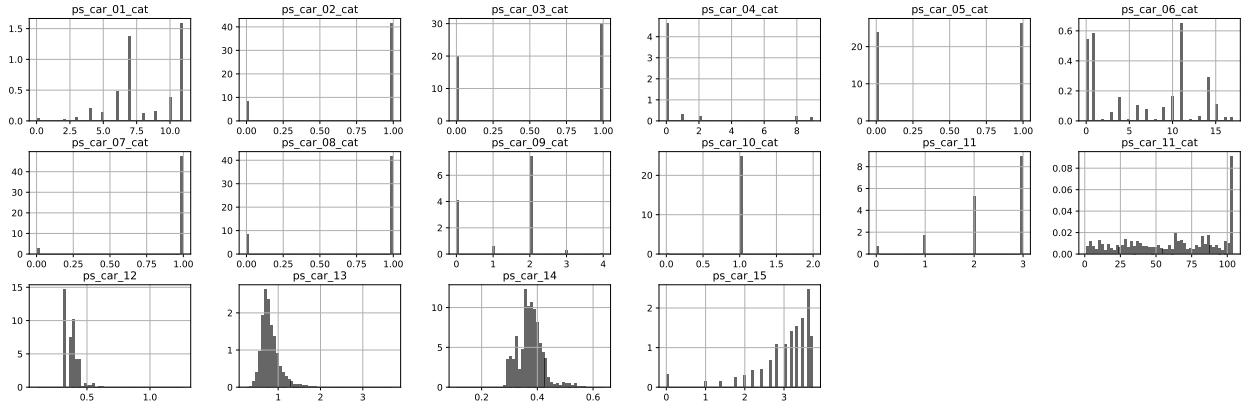

Fig. 3. Histogram for the `car` Attributes.

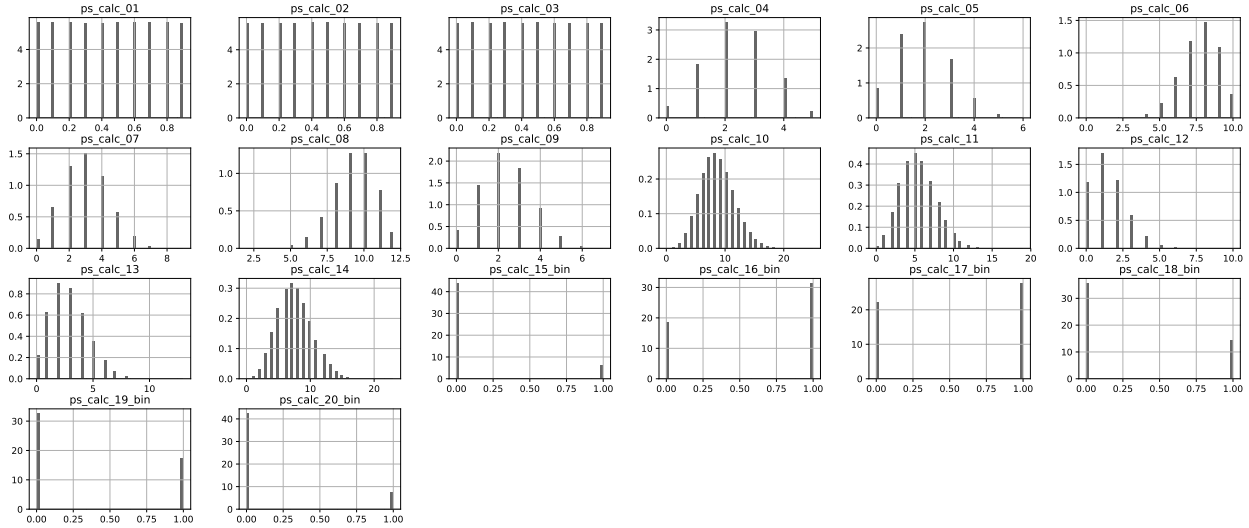

Fig. 4. Histogram for the `calc` Attributes.

tures have no missing value (only 13 features out of 51 have). Some features only have a small amount of instances with missing value (`ps_car_02_cat` and `ps_car_11` have 5 missing value and `ps_car_12` have only 1).

Based on these observation, we handled the missing values as follow:

- For features with a few missing values, delete those data instances.
- For categorical features, we treat the missing values a new category.
- For numeric features, we use the mean of the population as missing value.

IV. Experimental Evaluation

A. *Hyperparameter Tuning*

Hyper-parameters are parameters that are not directly learnt within estimators. Hyperparam-

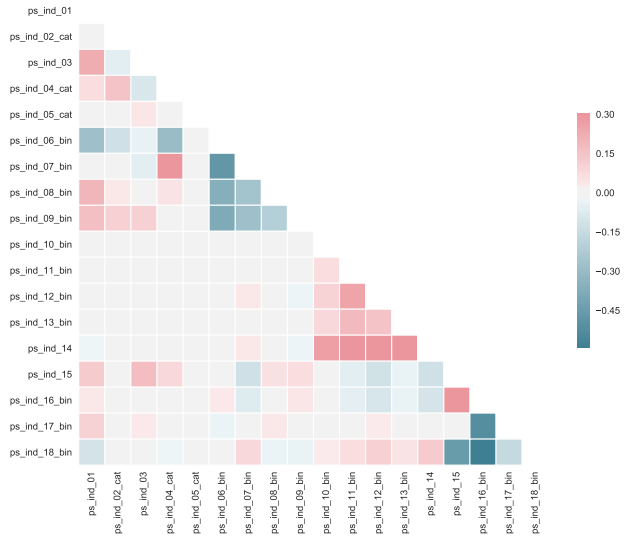

Fig. 5. Correlation for the `ind` Attributes.

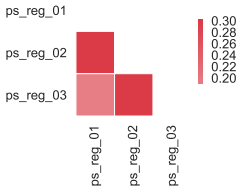

Fig. 6. Histogram for the `reg` Attributes.

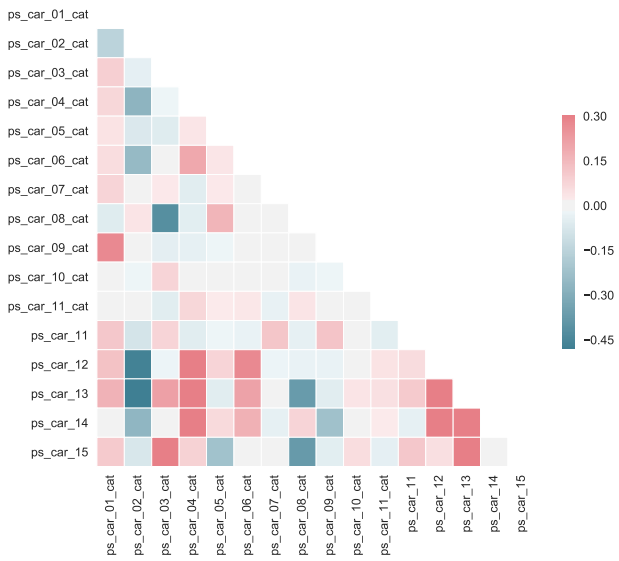

Fig. 7. Histogram for the `car` Attributes.

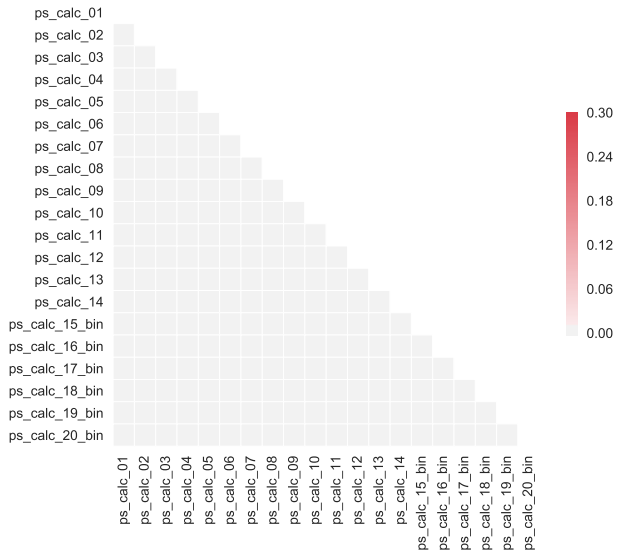

Fig. 8. Histogram for the `calc` Attributes.

eter settings could have a big impact on the prediction accuracy of the trained model. Optimal hyperparameter settings often differ for different datasets [16].

- Grid Search, which is commonly accepted as naive because the number of required samples grows exponentially in dimension, assuming the number of points per axis is held fixed. Though most expensive in terms of total computation time, if run in parallel, it is fast in terms of wall clock time. [11] has provided some insights into the relationship between classical grid search and probabilistic roadmaps.
- Random Search, which is a slightly variation on grid search. In stead of searching over the entire grid, random search only

TABLE II

MISSING VALUE COUNTS IN EACH FEATURE

Feature Name	Count	Percentage
ps_ind_02_cat	216	0.0363%
ps_ind_04_cat	83	0.0139%
ps_ind_05_cat	5809	0.9760%
ps_reg_03	107772	18.1065%
ps_car_01_cat	107	0.0180%
ps_car_02_cat	5	0.0008%
ps_car_03_cat	411231	69.0898%
ps_car_05_cat	266551	44.7825%
ps_car_07_cat	11489	1.9302%
ps_car_09_cat	569	0.0956%
ps_car_11	5	0.0008%
ps_car_12	1	0.0002%
ps_car_14	42620	7.1605%

evaluates a random sample of points on the grid. [6] has shown that random search is more efficient than grid search for hyper-parameter optimization in the case of several learning algorithms on several data sets.

Scikit-learn provides both Exhaustive Grid Search [2] and Randomized Parameter Optimization [3] class. To gain more insight about the hyperparameter tuning process (though might be useless), we used `ParameterGrid` in our implementations. Test results of all parameter sets are included in log files.

B. Evaluation Metrics

We used per-class accuracy, AUC and Normalized Gini Coefficient to evaluate out model.

1) Per-Class Accuracy: Accuracy simply measures how often the classifier makes the correct prediction. Its the ratio between the number of correct predictions and the total number of predictions (the number of data points in the test set) [16]:

$$accuracy = \frac{\#\text{ correct predictions}}{\#\text{ total data points}}$$

Per-class accuracy is the average of the accuracy for each class. By using per-class accuracy, we can have a better understanding of the model if the target class was dominated by one label.

2) Normalized Gini Coefficient and AUC: The Normalized Gini coefficient, (named for the similar Gini coefficient/index used in Economics, which originally developed by Italian statistician and sociologist Corrado Gini [9]), measures the inequality among values of a frequency distribution (for example, levels of income) [5]. It is most commonly defined as twice the area between the ROC curve and the diagonal (with this area being taken as negative in the rare event that the curve lies below the diagonal) [7].

Figure 9 shows that the Gini coefficient is equal to the area marked A divided by the sum of the areas marked A and B, that is, $Gini = A/(A+B)$. It is also equal to $2A$ and to $1-2B$

due to the fact that $A+B=0.5$ (since the axes scale from 0 to 1).

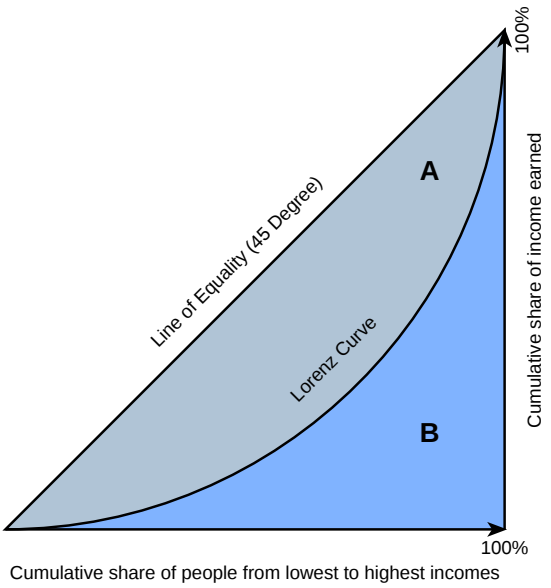

Fig. 9. Graphical Representation of the Gini Coefficient

The normalized Gini coefficient and AUC are closely related. AUC stands for area under the curve. When using normalized units, the AUC is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one (assuming 'positive' ranks higher than 'negative') [8]. Elementary geometry shows that

$$Gini = 2 \times \text{AUC} - 1. \tag{1}$$

The competition used normalized Gini coefficient to measure participants' submission performance. In our experiment, we worked in terms of both Gini coefficient and AUC.

C. Evaluation Mechanisms

We use k-fold cross-validation as our validation technique.

In k-fold cross-validation, we first divide the training dataset into k folds. For a given hyperparameter setting, each of the k folds takes turns being the hold-out validation set; a model is trained on the rest of the $k1$ folds and measured on the held-out fold. The overall performance is taken to be the average of the performance on all k folds. Repeat this procedure for all of the hyperparameter settings that need to be evaluated, then pick the hyperparameters that resulted in the highest k-fold average.

Scikit learn provides multiple variations of k-fold [15], for example,

- `KFold`, which divides all the samples in k groups of samples, called folds of equal sizes (if possible). The prediction function is learned using $k-1$ folds, and the fold left out is used for test.
- `LeaveOneOut`, which is basically special simple case when $k=n$ in `KFold`.
- `StratifiedKFold`, which returns *stratified* folds, i.e., each set contains approximately the *same* percentage of samples of each target class as the complete set.

We chose `StratifiedKFold` in out implementations during training phase.

D. Results

Present the quantitative results of your experiments. Graphical data presentation such as graphs and histograms are frequently better than tables. What are the basic differences revealed in the data. Are they statistically significant?

E. Discussion

Is your hypothesis supported? What conclusions do the results support about the strengths and weaknesses of your method compared to other methods? How can the results be explained in terms of the underlying properties of the algorithm and/or the data.

V. Related Work

Answer the following questions for each piece of related work that addresses the same or a similar problem. What is their problem and method? How is your problem and method different? Why is your problem and method better?

VI. Future Work

What are the major shortcomings of your current method? For each shortcoming, propose additions or enhancements that would help overcome it.

VII. Conclusion

Briefly summarize the important results and conclusions presented in the paper. What are the most important points illustrated by your work? How will your results improve future research and applications in the area?

VIII. Bilbiography

Be sure to include a standard, well-formated, comprehensive bibliography with citations from the text referring to previously published papers in the scientific literature that you utilized or are related to your work.

References

[1] Porto Seguros Safe Driver Prediction, 2017. [Online; accessed November 27, 2017].

[2] Scikit-learn api. 3.2.1. Exhaustive Grid Search, 2017. [Online; accessed November 27, 2017].

[3] Scikit-learn api. 3.2.2. Randomized Parameter Optimization, 2017. [Online; accessed November 27, 2017].

[4] Wikipedia: Artificial neural network. https://en.wikipedia.org/wiki/Artificial$_n eural_n etworkt$, 2017. [Online; accessed November 27, 2017].

[5] Wikipedia: Gini coefficient. https://en.wikipedia.org/wiki/Gini_coefficient, 2017. [Online; accessed November 27, 2017].

[6] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[7] Robert Dorfman. A formula for the gini coefficient. *The Review of Economics and Statistics*, 61(1):146–149, 1979.

[8] Tom Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, June 2006.

[9] Corrado Gini. *Variabilitá e Mutuabilitá.* 1912. Contributo allo Studio delle Distribuzioni e delle Relazioni Statistiche. C. Cuppini, Bologna.

[10] Jacob Joseph. How to treat missing values in your data : Part I, 2016. [Online; accessed November 27, 2017].

[11] Steven M LaValle, Michael S Branicky, and Stephen R Lindemann. On the relationship between classical grid search and probabilistic roadmaps. *The International Journal of Robotics Research*, 23(7-8):673–692, 2004.

[12] Thomas M. Mitchell. *Generative and Discriminative Classifiers: Naive Bayes and Logistic Regression*. draft of February, 2016.

[13] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[14] Amanda N Baraldi and Craig K Enders. An introduction to modern missing data analyses. 48:5–37, 02 2010.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-

learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[16] Alice Zheng. *Evaluating Machine Learning Models*. OReilly Media, Inc., 1 edition, 2015.