Predicting County Level Cost Differences for Treating Chronic Obstructive Pulmonary Disease

Jonathan Lin
jolituba@stanford.edu

Michael Smith
msmith11@stanford.edu

Aileen Wang
aileen15@stanford.edu

***Abstract*—This paper aims to predict county-level costs for treating Chronic Obstructive Pulmonary Disease (COPD), and understand which features have the most impact in this prediction. Using datasets derived from the Centers for Medicare and Medicaid Services, we ran a Chi-squared test to find the optimal feature set. Employing various binary and multiclass classification algorithms, including logistic regression, SVMs, and Naïve Bayes, we try to predict COPD cost percentage differences over a single year and a multiple year time span. We find that logistic regression, support vector machines (SVC), AdaBoost, and Naïve Bayes perform the best, with accuracies of 60% (single-year) and 57% (multi-year) for binary classification, and 65% (single-year) and 57% (multi-year) for multiclass classification.**

I. Introduction

A. Motivation

Chronic Obstructive Pulmonary Disease (COPD), is the third leading cause of death in the United States [1]. The National Institutes of Health state that 16 million Americans have been diagnosed with COPD; however, potentially more may have it but are unaware. While there is no current cure for COPD, it can be treated with medication [2].

The CDC estimates that COPD cost the United States $32.1 billion in 2010, with 76% of that being attributable to Medicare and Medicaid billings [3]. By 2020, COPD is projected to cost $49 billion [3]. Given such a rapid cost increase, we seek to explore the year-to-year and multi-year changes in average principal cost (APC) in Medicare & Medicaid spending at the county level, rather than the national level. We define APC as the average cost per beneficiary for hospitalization after COPD diagnosis. By predicting the change in cost annually and over a multi-year range for a given county, we believe that county and state officials could make more informed decisions about how to mitigate cost increases in their future.

B. Problem Definition

Our project aims to perform single-year and multi-year prediction, defined as follows:

- Single-year: We input county-level features from a specific year (ex. 2012), including hospitalization rate, COPD prevalence, and tobacco prevalence. We predict the cost percentage difference from that year to the next (ex. 2013).
- Multi-year: We input county-level features from a specific year range (ex. 2012-2014), including COPD prevalence, unemployment rate, and percent below federal poverty level. We predict the cost percentage difference from that time range to the next (ex. 2012-2015)

We will subdivide single-year and multi-year prediction into three tasks:

- Feature Selection: Find out the optimal feature set for single-year/multi-year prediction
- Binary Classification: Predict single-year/multi-year cost percentage increase or decrease for a given county in the next year/year-range
- Multiclass Classification: Predict the bucket for single-year/multi-year cost percentage difference for a given county. The buckets are varying percentage ranges between -100% and 100%. The buckets represent how much cost changes in the next year/year-range.

We use primarily supervised learning algorithms (Logistic Regression, Support Vector Machines, Naive Bayes, etc.) to make four predictions: single-year binary classification, multi-year binary classification, single-year multiclass classification, and multi-year multiclass classification.

II. Related Work

While there does not seem to be previous work predicting year-to-year and multi-year cost increases at the county level for Medicare and Medicaid, the problem of wanting to predict cost increases and risks associated with healthcare is nothing new. There seem to be two spheres of work most closely related to our project: patient level risk/cost predictions versus hospital level predictions. Moturu et al. [4] fall under the former; they aim to reduce direct and indirect patient costs by identifying high-risk patients. We share three models in common: logistic regression, AdaBoost, and SVMs; they also employ LogitBoost and logistic model trees [4]. Bertsimas et al. [5] also focus on healthcare costs at the patient level. They employ patient claims data from a commercially insured population (people who receive their insurance from an employer) instead of Medicare/Medicaid. Bertsimas et al. extend previous work in the domain by employing nonlinear models such as classification trees and clustering algorithms instead of the heuristic rules and regression methods of the past [5]. Wells et al. [6] make use of well-being data from employees and their dependents to predict individual future healthcare expenditure levels. They utilize a proprietary prediction algorithm, the Multidimensional Adaptive Prediction Process (MAPP), and derive their final results from using 2011 data to predict 2012

costs [6], a methodology akin to ours, but limited given that they only predict over a single year.

The two subsequent works originate from CS229, and predict cost at the hospital level. Furthermore, both works utilize Medicare data, which the previous three works did not. S. Rosston and S. Steele [7] predict the cost of treatment (hospital prices) for a specific condition: heart failure and shock. They use economic and demographic data from counties to augment their dataset on hospitals. They use four learning methods: linear regression, regression trees, support vector regression, and neural networks. Rosston and Steele find that neural networks produce the lowest RMSE out of their models [7]. J. Louie and A. Wells [8] predict expected Medicare costs in hospital referral regions (HRR) as well as individual hospitals using data from multiple years, similar to our research. They find that classification using logistic regression and linear discriminant analysis yields relatively poor accuracy [8] which coincidentally nearly matches the classification test accuracy found in this report, suggesting patterns in the employment of Medicare data.

III. Dataset and Features

A. Data Processing

For our project, we used the Centers for Medicare and Medicaid Services (CMS) databases [9]. We focused on selecting features related to COPD onset / treatment / prevention and the economic state of counties, all of which aid the prediction of COPD costs. For COPD-specific data, each downloaded dataset corresponded to one feature, and these datasets had five fields in common: year, fips (unique county code identifier), county, state, and analysis value of the selected feature. For county profile data, CMS grouped several features, including unemployment rate, average income, and percent below poverty, into one dataset.

We wrote several scripts to reorganize the data by training example for single-year prediction and multi-year prediction. First, we split the county profile data into separate datasets by feature. Then, we merged the COPD-specific and county profile datasets by the fips and year fields, so each row is a feature vector of a given county in a given year/year-range. Finally, we calculated the $y^{(i)}$ value for the percentage difference in average principal cost from one year/year-range to the next.

We performed some feature engineering to reach our final datasets. We originally included external features from the Center for Disease Control's National Environmental Public Health Tracking Network [3], including pollutant levels, percent forest cover, etc. However, the data for these features remained relatively stagnant from year to year, which we found degraded the performance of classification algorithms. We also removed features with a lot of missing data, as well as features that were similar to one another. For example, average income and unemployment rate are co-dependent, and therefore we removed average income as a feature.

B. Feature Selection

We have two final datasets, one for single-year prediction and one for multi-year prediction. The single-year dataset has 9498 rows, containing county-level information per year from 2012-2014. Each row consists of 13 fields:

- year = year
- fips = county identifier (used to merge data by county)
- county = county in a specified state
- state = state in the US
- urban = urban or rural
- copd_apc = COPD average principal cost
- copd_edv = emergency department visit rate
- copd_hos = hospitalization rate
- copd_pre = COPD prevalence
- copd_pqi = prevention quality indicator
- asthma_pre = asthma prevalence
- toba_pre = tobacco prevalence
- apc_perc_diff = percent difference in APC for treating COPD from current year to the next

For example, row = {year: 2013, fips: 1003, county: Baldwin County, state: ALABAMA, urban: 1, copd_apc: 1384, copd_edv: 518, copd_hos: 6, copd_pre: 10, copd_pqi: 1007, asthma_pre 4, toba_pre: 7, apc_percent_diff: -0.1225}.

The multi-year dataset has 3146 rows, containing county-level information over the time range 2012-2014. Each row consists of 15 fields. The multi-year dataset is very similar to the single-year dataset, but with a few modifications, defined as follows:

- year = year range
- copd_apc = removed as a feature
- copd_prs = preventative services
- below_poverty = percent below federal poverty level (5 year average)
- unemp_rate = unemployment rate (5 year average)
- apc_perc_diff = percent difference in APC for treating COPD from the current year range to the next

C. Choosing Optimal Features

To obtain the optimal subset of features for prediction, we performed a chi-squared test on both datasets to extract the set of features with the highest chi-squared score. We use the fit_transform method of SelectKBest from SKLearn [10] package to select features according to the k highest scores. The subset chosen by this method will boost prediction accuracy.

D. Classes for Binary and Multiclass Classification

For binary classification, we define the target y value as

$$y = \begin{cases} 1 & \text{if apc\_percent\_diff} > 0 \\ 0 & \text{otherwise} \end{cases}$$

For multiclass classification, we will test the performance of three different sets of class buckets. We will try using 6 buckets, 4 buckets, and 3 buckets, defined as follows:

- For 6 class buckets, we divided the apc_percent_diff percentage decrease and increase into the following intervals: [-100%, -20%], [-20%, -10%], [-10%, 0%], [0, 10%], [10%, 20%], [20%, 100%].
- For 4 class buckets, we divided the apc_percent_diff into the following intervals: [-100%, negative average], [negative average, 0%], [0%, positive average], [positive average, 100%]. Negative average is the average of the negative percentages, while positive average is the average of the positive percentages.
- For 3 class buckets, we divided it into the following intervals: [-100%, negative average], [negative average, positive average], [positive average, 100%].

IV. Methods

All classification algorithms described below were implemented in binary and multiclass classification, as described in the feature selection section. Binary classification predicted whether the APC of each county for treating COPD increases or decreases, while multiclass predicts the bucket of APC percentage increase or decrease. We began by implementing many different algorithms to see which performed well and which didn't. This allowed us to learn more about the data based on the performance of individual algorithms.

The train and test errors and binary confusion matrices are included for each algorithm. The binary confusion matrix is

$$C = [C_{00}, C_{01}, C_{10}, C_{11}]$$

Where C_{00} = true negatives, C_{10} = false negatives, C_{11} = true positives, and C_{01} = false positives. Multiclass confusion matrices have been omitted in the interest of space.

A. Logistic Regression

We used the logistic regression method in the linear model class of SKLearn [11], with the parameters of regularization $C = 1e5$ and Solver = 'lbfgs'. While logistic regression is primarily binary, we used the multiclass variant of the method with an additional parameter of multi_class = 'multinomial' for predicting the percentage increase or decrease bucket. Logistic regression performs gradient descent on θ to maximize the log-likelihood:

$$\ell(\theta) = \sum_i y^{(i)} log\left(g\left(z^{(i)}\right)\right) + \left(1 - y^{(i)}\right) log\left(1 - g\left(z^{(i)}\right)\right)$$

Where $z^{(i)} = \theta^T x^{(i)}$ and $g(z)$ is the sigmoid function. A prediction is then made by evaluating $p = g\left(\theta^T x^{(i)}\right)$ and predicting 1 if $p \geq 0.5$ else 0. For multiclass, the algorithm instead uses cross-entropy loss:

$$CE(y, \hat{y}) = -\sum_k y_k log \hat{y}_k$$

Logistic regression assumes that features are roughly linear and the data is well-separated. We varied the regularization level, finding that less regularization - a larger C-value - yielded slightly higher test accuracy. We observed the train and test accuracy of logistic regression for both binary and multiclass, single year (SY) and multi-year (MY):

Accuracy	SY Train	SY Test	MY Train	MY Test
Binary	0.6055	0.5908	0.6084	0.5731
Multiclass	0.6477	0.6484	0.6047	0.5720

The binary confusion matrices for single/multi year:
$C_{SY}, C_{MY} = [966, 500, 662, 722], [416, 94, 309, 125]$

B. Multi-layer Neural Network

We used the MLPClassifier (multi-layer perceptron) class in the neural_network package of SKLearn [12] with a quasi-Newton solver 'lbfgs'. Neural networks use multiple hidden layers to learn a non-linear function through backpropagation. This neural network in particular uses the cross-entropy loss as described above to find the weights $W^{[i]}$ and $b^{[i]}$ through gradient descent. We observed the train and test accuracy for MLP as follows:

Accuracy	SY Train	SY Test	MY Train	MY Test
Binary	0.6917	0.5673	0.8914	0.5286
Multiclass	0.6978	0.6066	0.8905	0.5635

The binary confusion matrices for single/multi year:
$C_{SY}, C_{MY} = [856, 610, 584, 800], [294, 216, 240, 194]$

C. K Nearest Neighbors

We used the KNeighborsClassifier class in the neighbors package of SKLearn. K nearest neighbors is a simple classifier that examines the k nearest neighbors to every training example and assigns it to the class shared by the most training examples out of the k nearest neighbors [16]. We observed the train and test accuracy for k nearest neighbors as follows:

Accuracy	SY Train	SY Test	MY Train	MY Test
Binary	0.7718	0.5238	0.7660	0.5529
Multiclass	0.7060	0.5992	0.7124	0.5688

The binary confusion matrices for single/multi year:
$C_{SY}, C_{MY} = [1108, 358, 994, 390], [426, 84, 338, 96]$

D. Support Vector Machines

We used the SVC class in the svm package of SKLearn. SVMs treats feature vectors as high dimensional points in space, which it tries to separate with a hyperplane in order to create the largest margin between the points and the decision boundary [13]. The classifier uses the hinge loss, which is given by:

$$L = max(0, 1 - y^{(i)}(w^T x^{(i)} + b))$$

This loss function means that a training example only contributes to the loss function if it is misclassified. We observed the train and test accuracy for SVM as follows:

Accuracy	SY Train	SY Test	MY Train	MY Test
Binary	0.5964	0.6086	0.5570	0.5402
Multiclass	0.6441	0.6378	0.5520	0.5360

The binary confusion matrices for single/multi year:
$C_{SY}, C_{MY} = [990, 476, 675, 709], [510, 0, 434, 0]$

E. Decision Tree

We used the DecisionTreeClassifier class in the tree package of SKLearn. Decision trees start from a root and follow a path down multiple nodes based on the values of each feature. It then calculates the majority class for a given path and always predicts that class for any unseen data that follows a given path [21], [22]. We observed the train and test accuracy for the decision tree as follows:

Accuracy	SY Train	SY Test	MY Train	MY Test
Binary	1.0	0.5245	1.0	0.5211
Multiclass	1.0	0.5119	1.0	0.5233

The binary confusion matrices for single/multi year:
$C_{SY}, C_{MY} = [782, 684, 640, 744], [282, 228, 221, 213]$

F. Random Forest

We used the RandomForestClassifier class in the ensemble package of SKLearn. Random forests are very similar to decision trees in that they use multiple decision trees on various subsets of the data, and then aggregate all of the trees' predictions to make a final prediction [23]. We observed the train and test accuracy for the random forest as follows:

Accuracy	SY Train	SY Test	MY Train	MY Test
Binary	0.9831	0.5624	0.9813	0.5497
Multiclass	0.9822	0.6164	0.9800	0.5286

The binary confusion matrices for single/multi year:
$C_{SY}, C_{MY} = [988, 478, 740, 644], [344, 166, 285, 149]$

G. AdaBoost

We used the AdaBoostClassifier class in the ensemble package of SKLearn. AdaBoost, or adaptive boosting creates k "weak classifiers," or classifiers that do slightly better than random. The algorithm then runs each weak classifier on the data, with each successive weak classifier weighting more heavily on the training examples that were missed by the previous weak classifiers [17]–[19]. This is represented as

$$H_{final}(x) = \textbf{sign} \sum_k \alpha_k h_k(x).$$

We observed the train and test accuracy for AdaBoost as follows:

Accuracy	SY Train	SY Test	MY Train	MY Test
Binary	0.6278	0.5905	0.6510	0.5677
Multiclass	0.6493	0.6459	0.5711	0.5497

The binary confusion matrices for single/multi year:
$C_{SY}, C_{MY} = [923, 543, 620, 764], [360, 150, 258, 176]$

H. Naive Bayes

We used the GaussianNB class in the naive_bayes package of SKLearn. Naive Bayes looks at each feature independently and assigns probabilities to each feature value based on the y value [14]. It classifies into a class k such that:

$$k = argmax_k \left(P(y = k) \prod_{i=1}^{n} P(x_i | y = k) \right)$$

This means that it classifies into the class k that has the highest posterior probability, as given above. However, this model makes the strong assumption that all of the x_i's are conditionally independent given y. We observed the train and test accuracy for Naive Bayes as follows:

Accuracy	SY Train	SY Test	MY Train	MY Test
Binary	0.5857	0.5919	0.5906	0.5720
Multiclass	0.6389	0.6336	0.5870	0.5635

The binary confusion matrices for single/multi year:
$C_{SY}, C_{MY} = [1079, 387, 762, 622], [419, 91, 313, 121]$

I. Evaluation Metrics

We are using score and accuracy_score methods (see equations (1) and (2) below) in SKLearn package to measure the training and test accuracy of the above classification algorithms, which are calculated by dividing the number of correct predictions by the total number of train and test data examples respectively.

$$Train\_Accuracy = Fit.score(X\_train, Y\_train)$$

$$Test\_Accuracy = accuracy\_score(pred, Y\_test)$$

We also measure the confusion matrix using

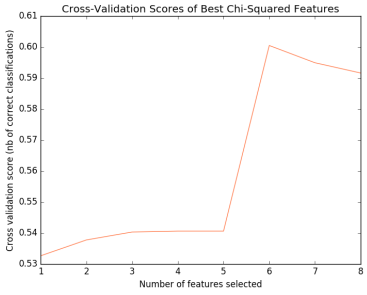
$$[C_{00}, C_{01}, C_{10}, C_{11}] = confusion\_matrix(Y\_test, pred)$$

from the SKLearn package.

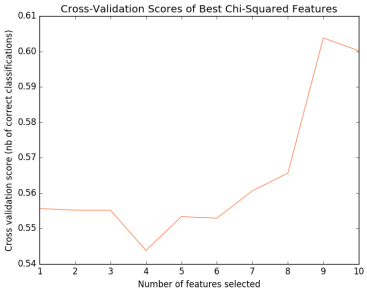
V. Results and Analysis

We first use the train_test_split method in the SKLearn package to randomly select 70% for the train dataset and 30% for the test dataset from the dataset with 9498 rows for single year and 3145 rows for multi-year defined in Section VI. Then we apply the following data preprocessing functions from Sklearn package:

- **StandardScaler**: Feature scaling through standardization (or Z-score normalization)
- **PCA**: Linear dimensionality reduction using Singular Value Decomposition of the data to project it to a lower dimensional space.



(a) Single Year Binary Prediction



(b) Multi-Year Binary Prediction

Fig. 1: Chi-Squared Score

Finally, we run the classification algorithms in Section VI on this preprocessed dataset configuration and obtained the following results:

A. Optimal Features Selection

After running Chi-squared as the scoring function, we determined that the optimal number of features to be used for binary prediction were 6 for single year and 9 for multi-year (see Figure 1). Similarly, for multiclass prediction, 7 features and 10 features are need for single year and multi-year respectively.

B. Classifier Accuracy Comparison

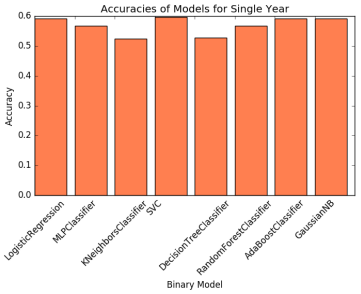
- After running our binary and multiclass classifiers on the selected features, we observed that logistic regression, SVM, AdaBoost, and GaussianNB are the top performers with test accuracy around 60% (See Figure 2) for single year prediction. Since logistic regression and SVM expect linear features or features that interact linearly, it implies that the selected features in our model are more linearly correlated.
- For single year multiclass prediction, the best test accuracy is 65% for three classes (see target y in Section III). However, when we tested it on ≥ 4 classes, accuracy dropped below 40%.
- For multi-year prediction, the top four models: logistic regression, KNeighbors, AdaBoost, and GaussianNB performed well both for binary and multiclass classifications (See Figure 3). These models have 57% test accuracy in both binary and multiclass predictions.
- The accuracy of the algorithms above have benefited from data preprocessing. The SVM algorithm's accuracy increased about 10% after applying data preprocessing, but other algorithms just increased about 1%.
- We also tried to add more COPD related features, such as the percentage of forest cover and pollutant levels per county, but the data for these features were not available each year. Since the features were unchanging year to year, they degraded classification performance both for binary and multiclass. That was the reason to remove them from consideration; albeit, had the data changed year-to-year, yearly pollutants and forest cover might have been salient features.
- While the performance of algorithms such as random forest and AdaBoost is comparable to the other algorithms, the level that they are performing at seems low for these particular algorithms, which tend to be strong algorithms. This implies an issue with the data - too noisy, or too many outliers, which are both known issues with these algorithms, particularly AdaBoost [18].

VI. Conclusion and Future Work

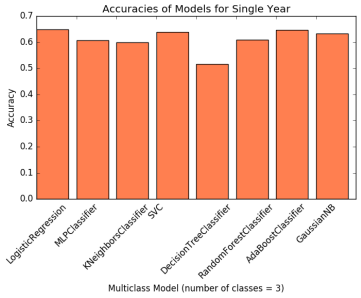
Overall, the results of the algorithms that we ran were decent. The best performing algorithms are logistic regression, support vector machines (SVC), AdaBoost, and Naïve Bayes, with accuracies of 60% (single-year) and 57% (multi-year) for binary classification, and 65% (single-year) and 57% (multi-year) for multiclass classification. While the error rate for multiclass classification was initially low, completely reworking the data and features brought the accuracy much higher. Eventually, the accuracy for 3-class classification became better than binary classification through careful feature selection.

However, based on the analysis of our results, it seems as if there is still work to be done on the data. Most of the performance of our algorithms are limited by the amount of available COPD-related data. The CMS database only provided COPD and county profile data from 2012-2015, which may impact our models' ability to predict cost differences for other years. Furthermore, analysis of several algorithms indicates that the data is too noisy.

In the future, we would like to find more data from past years to better generalize our cost predictions. Additional work could also be done in finding features that are more strongly indicative of the changes in cost, due to the noisiness discussed previously. Finally, another interesting application would be to explore county level cost differences for treating other chronic illnesses, such as cancer, dementia, and heart disease.

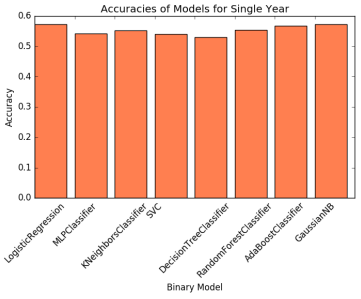


(a) Single Year Binary Prediction

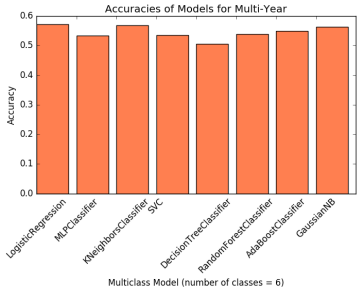


(b) Single Year Three Classes Prediction

Fig. 2: Single Year Test Accuracy Comparison



(a) Multi-year Binary Prediction



(b) Multi-year Six Classes Prediction

Fig. 3: Multi-Year Test Accuracy Comparison

VII. Contributions

Everyone was involved in extracting features from different sources and merging the datasets together. Jonathan focused on implementing a linear least squares classification (which

didn't make it into the final report due to space), writing about the algorithms we used, and error analysis/future work. Aileen focused on reworking the data to be more usable/less error prone, feature analysis, and the binary classification algorithms. Michael focused on the multiclass classification algorithms, as well as researching past related works. All of us equally contributed to different parts of the report.

REFERENCES

[1] What Is COPD? - NHLBI, NIH, Nhlbi.nih.gov, https://www.nhlbi.nih.gov/health/health-topics/topics/copd/

[2] COPD - NHLBI, NIH, Nhlbi.nih.gov, https://www.nhlbi.nih.gov/health-topics/copd

[3] CDC Features - Increase expected in medical care costs for COPD, Cdc.gov, https://www.cdc.gov/features/ds-copd-costs/index.html

[4] S. Moturu, W. Johnson and H. Liu, "Predictive risk modelling for forecasting high-cost patients: a real-world application using Medicaid data", International Journal of Biomedical Engineering and Technology, vol. 3, no. 12, pp. 114-132, 2010.

[5] D. Bertsimas, M. Bjarnadóttir, M. Kane, J. Kryder, R. Pandey, S. Vempala and G. Wang, "Algorithmic Prediction of Health-Care Costs", Operations Research, vol. 56, no. 6, pp. 1382-1392, 2008.

[6]]A. Wells, X. Guo, C. Coberley and J. Pope, "Integrating Well-Being Information and the Multidimensional Adaptive Prediction Process to Estimate Individual-Level Future Health Care Expenditure Levels", Population Health Management, vol. 19, no. 6, pp. 429-438, 2016.

[7] S. Rosston and S. Steele, "The Price is Right? Estimating Medicare Costs with Machine Learning", CS229 Final Project, Stanford Univ., Stanford, CA, United States, 2015. Available:

[8] J. Louie and A. Wells, "Predicting Medicare Costs Using Non-Traditional Metrics", CS229 Final Project, Stanford Univ., Stanford, CA, United States, 2016. Available: http://cs229.stanford.edu/proj2016spr/report/029.pdf

[9] "Mapping Medicare Disparities", https://data.cms.gov/mapping-medicare-disparities, 2017

[10] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[11] "sklearn.linear_model.LogisticRegression — scikit-learn 0.19.1 documentation",Scikit-learn.org, 2017. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.

[12] "1.17. Neural network models (supervised) — scikit-learn 0.19.1 documentation",Scikit-learn.org, 2017. [Online]. Available: http://scikit-learn.org/stable/modules/neural_networks_supervised.html.

[13] "1.4. Support Vector Machines — scikit-learn 0.19.1 documentation", Scikit-learn.org, 2017. [Online]. Available: http://scikit-learn.org/stable/modules/svm.html.

[14] "1.9. Naive Bayes — scikit-learn 0.19.1 documentation", Scikit-learn.org, 2017. [Online]. Available: http://scikit-learn.org/stable/modules/naive_bayes.html.

[15] P. Cunningham and S. Delany, "k-Nearest Neighbour Classifiers", ResearchGate, 2007.

[16] "sklearn.neighbors.KNeighborsClassifier — scikit-learn 0.19.1 documentation", Scikit-learn.org, 2017. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.

[17] J. Duchi, "CS 229 Supplemental Lecture Notes - Boosting", 2017. [Online]. Available: http://cs229.stanford.edu/extra-notes/boosting.pdf.

[18] "Explaining AdaBoost", 2017. [Online]. Available: http://rob.schapire.net/papers/explaining-adaboost.pdf.

[19] "sklearn.ensemble.AdaBoostClassifier — scikit-learn 0.19.1 documentation", Scikit- learn.org, 2017. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html.

[20] E. Emer, "Boosting (AdaBoost Algorithm)", MIT, 2017. [Online]. Available: http://math.mit.edu/ rothvoss/18.304.3PM/Presentations/1-Eric-Boosting304FinalRpdf.pdf.

[21] "sklearn.tree.DecisionTreeClassifier — scikit-learn 0.19.1 documentation", Scikit- learn.org, 2017. [Online]. Available: http://scikit- learn.org/stable/modules/generated/ sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.

[22] A. Moore, "Decision Trees", Carnegie Mellon University, 2017. [Online]. Available: https://www.autonlab.org/_media/tutorials/dtree18.pdf.

[23] "sklearn.ensemble.RandomForestClassifier — scikit-learn 0.19.1 documentation", Scikit-learn.org, 2017. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.