Structure Type Classification Based on Tax Assessor Data

Yue (Major) Zeng
majorzeng@stanford.edu

1. Background and previous work

In regional earthquake simulation, damage occur when the of ground motion demand exceeds the strength capacity of each buildings. It is often difficult to assess the strength of millions of buildings in an area. The common practice is to sort buildings into groups with the same load bearing systems, like wood frame or concrete shear wall systems, which is the key feature that defines the strength of a building. These groups are called structural types. Building in each group have similar structural behavior and can be modeled with a single prototype building [1]. However, the structural type is something that's hidden behind the walls, deep in the design drawings and not openly available for most of the time. The most accurate information of a building's structural type come from in situ assessment of a professional engineer. However, the sheer number of buildings exists in any urban area makes such method infeasible.

The purpose of this project is to explores machine learning tools that would automate the process of assigning the structural type given certain information of the building.

A past studies have attempted the same task using various remote sensing data. The data used in the study are various physical base features, (high, size, texture) extracted from Lidar and high-resolution satellite images. The classification model implement feature select, oversampling and multiclass classification in sequence. [2] This project follows similar procedure and explored various weight balancing, feature selection and classification scheme on a different dataset based on field survey that includes a broader range of building information.

2. Data processing

The data used in this study is tax assessor file of San Mateo county for the year of 2016. It includes 128137 rows and 149 columns, 71 with numerical values and 78 with strings. Amount all columns, 39 columns have less than 1% of examples filled and are dropped due to extremely low completion. 15 columns with string information contains more than 128 (0.1% of number of examples) unique values, thus it's determined to be specific information, not classifications and dropped. The resulting data frame has 51 numerical columns and 44 string columns.

The columns "CONSTRUCTION TYPE" directly indicate the structural type, which is the information this study tries to predict. Thus, the column "'CONSTRUCTION TYPE" is treated as the label. Among 128137 examples, 23143 examples are labeled, and 104994 examples are not labeled.

The set of data that's labeled is used for this study. All columns that contains string are treated as categorical information and converted to numerical values by one hot encoding. The result data includes 23143 examples, each with 16 labels and 531 features. All features are normalized. The labeled date set is randomly shuffled and partitioned to be 70% training set, 15% validation set and 15% test set.

3. Attempts to handle imbalanced data set

For the training data set of 16201 examples, the 16 construction type labels, their definition and number of examples in each category are shown in the *Table 1* below.

Table 1 Label Distribution in Training Set

Structure Type	Training	
	Number of examples	Percentage of examples
WOOD	4686	20.2%
CONCRETE	298	1.3%
TYPE UNKOWN	13	0.1%
FRAME/STEEL	1	0.0%
STEEL/MASONRY	3	0.0%
FRAME	17973	77.7%
MANUFACTURED/MOD	5	0.0%
BRICK	87	0.4%
MASONRY	54	0.2%
CONCRETE BLOCK	12	0.1%
STEEL	1	0.0%
CUSTOM	5	0.0%
METAL	1	0.0%
HEAVY	1	0.0%
ADOBE	2	0.0%
LIGHT	1	0.0%
TOTAL	23143	

It is observed is the majority of the building are either wood or frame. All other types of structure are less than 5% of the examples, which could be considered rarer event. It is because San Mateo county is by majority a residential area and 99% of American single family home are built with wood structure. Such dataset is very imbalanced and its' expected that any model would have low accuracy on prediction the "rare event" classes. However, even if the percentage of building that's concrete, steel or masonry are small, it's very important to capture them correctly since they have very different structural behaviors than wooden buildings, which is important in damage simulation.

Three methods are considered here in the study to overcome the issue of imbalanced data set **1) random**

over sampling 2) random under sampling and 3) training with weighted samples [1].

For the random sampling methods, the sampling goal is to make the number examples in concrete, masonry and steel classes at least 10% of that in wood classes.

A simple binary relevance classifier with logistic regression is used to make prediction and score accuracy [2]. The model builds 16 logistic regression classifier and compute the probability of an examples belong to each category: $f_i(x)$, $i \in \{1,2,\dots,16\}$. Then it predicts the example belongs to the categories with the highest probability:

$$f(x) = \arg\max_i f_i(x)$$

There are totally 69 examples in concrete, masonry or steel class in the validation set. All 531 features are used in the training. The evaluation criteria are test accuracy and number of samples misclassified in concrete, steel and masonry structure classes (both false positive and false negative).

All three methods are implemented along a baseline case where no manipulation of examples is done. The results are summarized in the *Table 2* below. It is shown that there is no improvement in both test accuracy and number of misclassified examples in rare categories due to any of the methods. Rather, when attempted to raise the weight of rare events in the data, all three methods misled the model to classify many more wood structure examples into the rare categories. As a result, the false positive examples in the rare categories overwhelms the true positive examples and make the classification less reliable.

Table 2 Summery of Result from Different Data Manipulation Methods

		Overall Accuracy	Number of Misclassified in Rare Classes
Baseline	No Manipulation	98.6%	49
Method (1)	Random Over Exampling	96.3%	104
Method (2)	Random Under Exampling	94.1%	140
Method (3)	Train with Weighted Examples	96.3%	106

4. Feature Selection Attempts

The number of features for each example in this dataset is not trivial (531). Even though it's not large enough to be an issue with computing power available, it is hard to collect these many features in the field. The dataset used here comes from US tax assessor, which is a very intensive survey. It's very hard to collect such detailed data in other occasions. On the other hand, because this data was collected for a different purpose than classifying structure types, there any many features that are repetitive or not very relevant. In this section, the study explores two different ways to rank the importance of features and made a series of experiments to try to find the number of top ranked features needed to produce a relatively accurate classification.

4.1 Feature Ranking

In practice, filter feature selection methods is the easiest to implement, since it doesn't depend on the type of classifier used later in the process. Two filter methods are considered here in this study to rank the importance of features: chi square method and one-off method.

Chi square test is amained to test the independence of two events, in this case one specific feature and one label. The chi square score is calculated:

$$\mathrm{X}^2(\mathbb{D}, t, c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{\left(O_{e_t e_c} - E_{e_t e_c}\right)^2}{E_{e_t e_c}}$$

Where $N_{e_t e_c}$ is observed frequency in $\mathbb{D}$ and E is expected frequency, e_t denotes occurrence of a feature and e_t occurrence of a label [3]. Chi square scoring is implemented with sklearn function where it only takes in one label at a time [4], so the final scoring of a feature is the sum of its score over all 16 labels. The higher the score, the more important such feature is. Note that this method essentially gives the same ranking as mutual information $\mathrm{MI}(\mathrm{U};\mathrm{C}) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} P(U = e_t, C = e_c) \log_2 \frac{P(U=e_t, C=e_c)}{P(U=e_t)P(C=e_c)}$, which has a more intuitive probabilistic interpretation but also less computationally efficient [5].

The one-off methods is the more computational intensive. It takes turns in leaving each of the features out and run the rest of features through the same classification model. The worse a model performs without a feature, the more important that feature is. Such experiment is done and the result of each one-off model is evaluated by the model test accuracy, which is a value from 0 to 1, plus a score function $\exp(-\frac{m-49}{100})$, where m is the number of misclassified example in rare categories. For the penalty term, if the number of misclassified rare examples is 49 which is the same as baseline model with all feature as input, it the additional score function is at value 1; it the new model misclassified 59 rare events, 10 more than baseline, the score is about 0.9; however, if the new model misclassified less than 49 examples, the score could be higher than 1.

The scores and ranks are plotted in *Figure 1* and *Figure 2* below for two methods. For chi-square test, the score exponentially decays as the rank increase. Features ranking higher than 150 is likely to have little relevance. For the one-off score, features ranking from 25 to 450 essentially have the same score. Only the top 25 ranked features demonstrate more importance from the rest of the set.

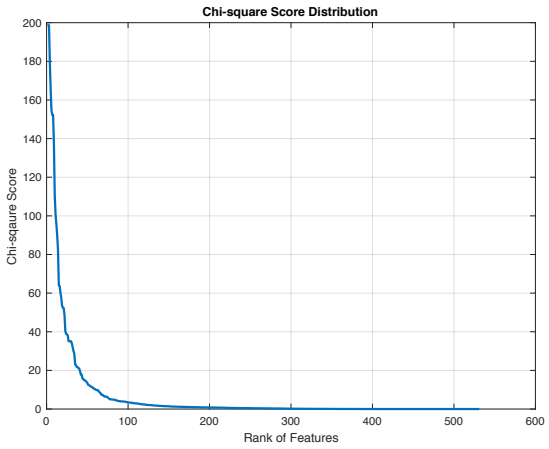

Figure 1 Chi-square score vs. rank

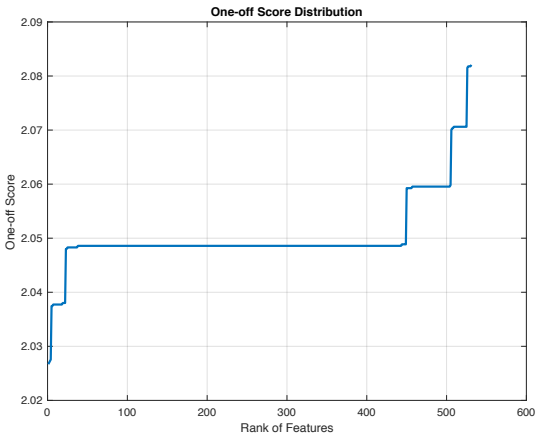

Figure 2 One-off score vs. rank

To understand the results of these ranking experiments, all features are sorted into 10 groups according to type of information they provide. These categories, total features in each category and the number of features ranked top 50 by the 2 ranking methods are summered in *Table 3* and *Figure 1* below.

Table 3 Summery of Top 50 Features

Feature Category	All Features	Chi-square Selection		One-off Selection	
		Number of Features	Percent of all features	Number of Features	Percent of all features
Owner Status	47	3	6%	13	28%
Building and Lot Size	21	5	24%	0	0%
General Location	251	9	4%	14	6%
Building Value	11	1	9%	1	9%
MISC	6	1	17%	0	0%
Architectural Layout	60	9	15%	8	13%
Code and General Condition	52	6	12%	7	13%
Building Occupany	25	4	16%	2	8%
Non-structural Component	21	5	24%	1	5%
Structural Component	37	7	19%	4	11%

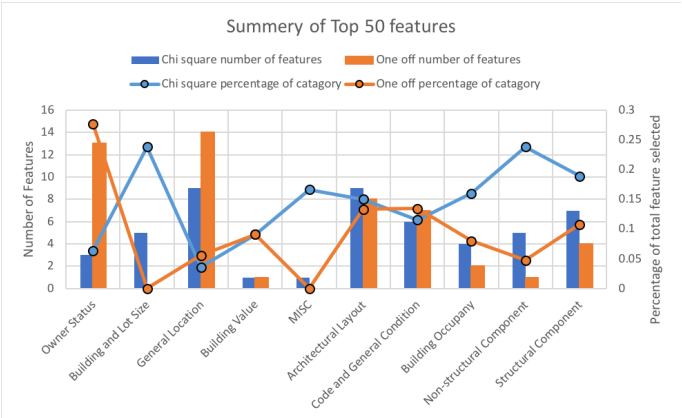

Figure 3 Summery of Top 50 Features

Chi-square test ranked many features regarding location of building, architecture layout (number of bedrooms, basement and garage) or structure component (frame, wall and foundation) in top 50, indicating that those are relevant information for prediction structural type. Also, a large percent of features describing non-structural components (air conditioning, fire place and sewer) or size of the building are selected, indicating that those categories are important as well. One-off rest ranked many features of owner status or building location in top 50, indicating that those are important information category. Ranking from these two methods disagrees on the importance of many feature categories. However, both ranked only 1 feature about the value of the building into top 50, indicating that the value of the building is not very related to its' structural type and thus the strength of the building.

5.2 Comparing of Ranking Methods

3 different feature selection methods: 1) chi-square scoring, 2) one-off testing and 3) manual selection from domain knowledge are tested in the following experiments. The same number of top ranked features are draw by each method and used as input of the same baseline binary relevance classifier with logistic regression model. The result of their accuracy and number of misclassified rare events are summarized in *Table 4*, *Figure 4* and *Figure 5* below.

Table 4 Accuracy and Misclassified Examples with Number of Features

Nume of Features	Manual Selection		Chi-square Ranking		One-off Ranking	
	Accuracy	Misclassified in Rare Classification	Accuracy	Misclassified in Rare Classification	Accuracy	Misclassified in Rare Classification
151	98.9%	36	98.8%	39	98.8%	35
111	98.8%	42	98.8%	38	98.6%	39
70	98.8%	42	99.0%	32	96.6%	57
62	98.8%	42	98.9%	40	96.7%	57
49	98.8%	42	98.9%	41	95.3%	63
38	98.8%	42	98.8%	44	95.4%	62
30	98.5%	53	98.9%	41	88.9%	62
16	98.2%	58	98.9%	41	79.2%	76
7	97.7%	75	98.6%	46	76.6%	75

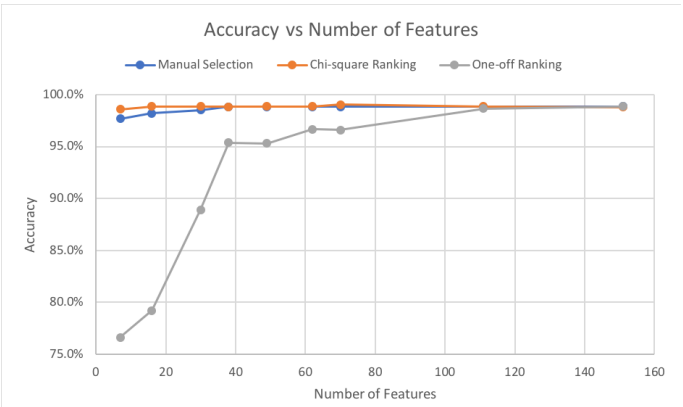

Figure 4 Accuracy with number of features

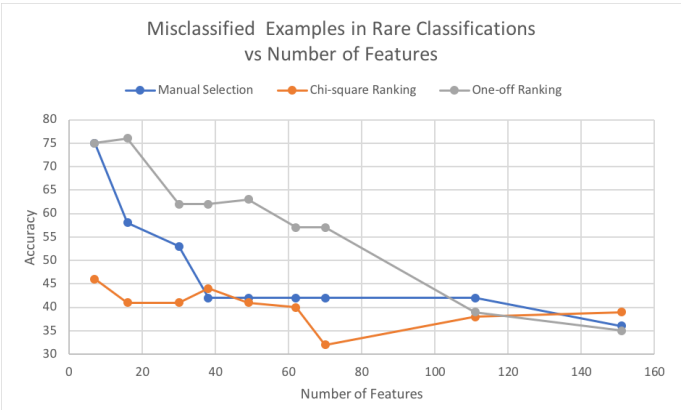

Figure 5 Misclassified Examples in Rare Classifications with number of features

The result shows that all three have the same high accuracy when using to 150 features. However, as the number of feature decrease, the accuracy of the one-off selected model decreases significantly, while the chi-square model and manual selection model remains at

high performance. For the number of misclassified rare events, increase significantly as the number of feature decrease, for the manual selection model and one-off test model, while the performance of chi-square model remains strong. Over all, chi-square is the strongest feature selection methods among the three and thus is carries on to the next section.

6. Prediction Methods

The next section explores the performance of 5 different multilabel classification algorithms on this dataset: 1) binary relevance classifier with logistic regression, 2) k-nearest neighbor classifier, 3) decision tree classifier, 4) random forest classifier and 5) 2 layer neural network.

K-nearest neighbor classifier: for each test example x_i with p features, the algorithm search for its k nearest neighbors $(x_{j1}, x_{j2}, \dots, x_{jk})$ form the training set, with the distance defined by:

$$d(x_i, x_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{ip} - x_{jp})^2}$$

The model predicts the label of test examples x_i the most frequent true label among k nearest training examples [6].

It's implemented with sklearn library, function k-nearest neighbor classifier [7]. Through iteration, the best hyper parameter k is determined to be 5.

Decision Tree Classifier: The algorithm repetitively search for attribute with the highest information gain: $H = -\sum_i p_i \log p_i$ and partition the data into subsets [8], demonstrated in *Figure 6* below [9]. The algorithm is implemented with sklearn library, function decision tree classifier [8].

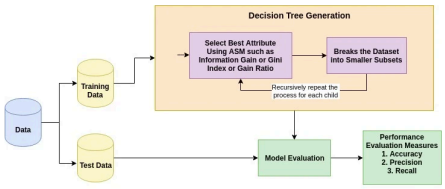

Figure 6 Decision Tree Classification

Random Forest Classifier: it's a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The algorithm is implemented with library, sklearn function random forest classifier [9]

Multilayer perceptron neutral network: the model is a single 2-layer neural network with sigmoid function as activation function of the hidden layer and soft max for output. The loss function in optimization is the sum of cross entropy loss for all labels [10], $L = -\sum_{c=1}^{16} y_c \log p_c$. The algorithm is implemented through library, sklearn NN-MLP function. Through iteration, the hyperparameter number of neurons in hidden layer is chosen to be 40. This method takes noticeably longer time than the previous ones.

Each classifier model is run with 10, 20 until 70 features with top relevance according to chi-square ranking. There test accuracy and number of misclassified rare event are summarized in *Table 5, Figure 8 and Figure 9* below.

The results show that neural network, k-nearest neighbor and random forest methods performs consistently better than baseline binary relevance classification. Decision tree methods performs well with the minimum number of features; however, it's performs deteriorates as the number of features increase. Random forest performs better than baseline but slightly less than k-earnest neighbor or neural network. It's is also less stable.

Over all, K-nearest neighbor and neural network are the better methods among testes, while neural network required more higher computing power.

Table 5 Comparison of classification methods

	Number of Features	10	20	30	40	50	60	70
Baseline OnevsRest	Accuracy	98.3%	98.5%	98.5%	98.5%	98.5%	98.6%	98.6%
	Misclassified Rare Events	57	59	59	59	51	50	46
KNearest Neighbor	Accuracy	98.5%	98.7%	98.8%	98.8%	98.9%	98.8%	98.8%
	Misclassified Rare Events	49	45	42	42	38	38	38
Decision Tree	Accuracy	98.6%	98.6%	98.2%	98.2%	98.0%	97.8%	97.8%
	Misclassified Rare Events	46	44	65	66	70	77	73
Random Forest	Accuracy	98.6%	98.7%	98.5%	98.6%	98.8%	98.8%	98.7%
	Misclassified Rare Events	46	46	48	46	41	39	38
Neural Network MLP	Accuracy	98.6%	98.8%	98.9%	98.8%	98.8%	98.8%	98.7%
	Misclassified Rare Events	46	41	40	43	39	40	41

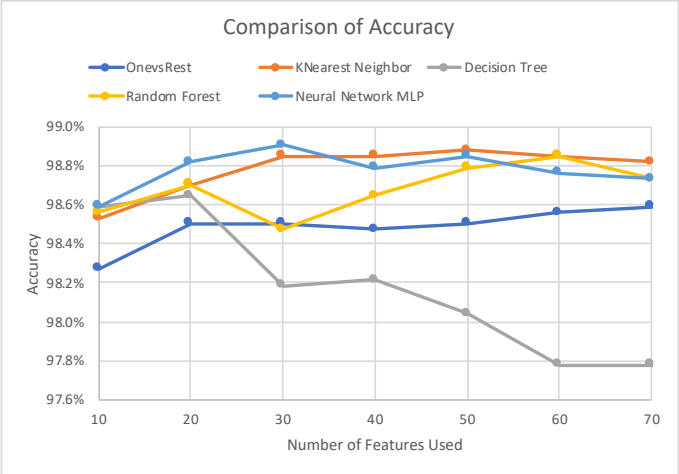

Figure 7 Accuracy Comparison

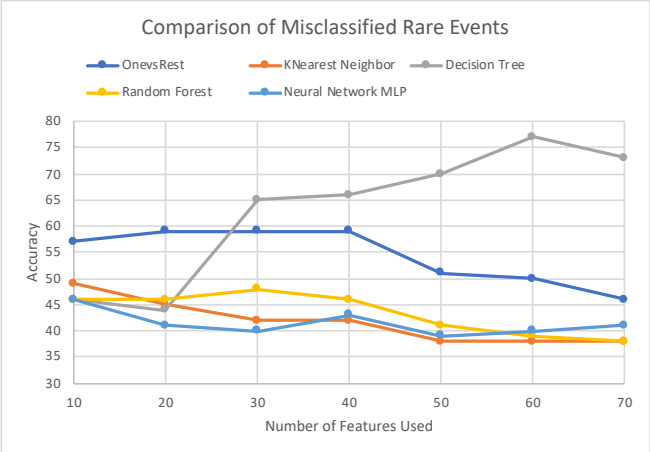

Figure 8 Campare of Mis

7. Prediction for the full dataset

The k-nearest neighbor model and neural network model are retrained with all 23143 labeled features and

used to predict the labels for the rest 104994 unlabeled examples. The result of prediction is shown in *Table 6* below.

Table 6 Prediction Results

Structure Type	Prediction			
	K-nearest Neighbor		Neural Network	
	Number of examples	Percentage of examples	Number of examples	Percentage of examples
WOOD	289	0.3%	630	0.6%
CONCRETE	12	0.0%	1433	1.4%
TYPE UNKOWN	0	0.0%	0	0.0%
FRAME/STEEL	0	0.0%	0	0.0%
STEEL/MASONRY	0	0.0%	0	0.0%
FRAME	103106	98.2%	100456	95.7%
MANUFACTURED/MOD	0	0.0%	0	0.0%
BRICK	1428	1.4%	2396	2.3%
MASONRY	157	0.1%	47	0.0%
CONCRETE BLOCK	0	0.0%	29	0.0%
STEEL	0	0.0%	0	0.0%
CUSTOM	0	0.0%	0	0.0%
METAL	0	0.0%	0	0.0%
HEAVY	0	0.0%	0	0.0%
ADOBE	0	0.0%	1	0.0%
LIGHT	0	0.0%	0	0.0%
TOTAL	104994		104994	

Prediction from both methods are highly skewed towards to common class "frame" with k-nearest neighbor prediction 98.2% and neural network predicts 95.7% belongs to that category. Neural network predicts slightly higher number of buildings in concrete and masonry structural type than k-nearest neighbor. Both model seems not able to classify rare events well. However, since there is no label available, it's not possible to evaluate accuracy such prediction. Even the prediction result has class distribution different from the the labeled set, and such skewed class composition does not inspire too much confidence, this prediction is still possible. Since the labeled are manually logged and its data is sparse, it's possible that building in certain rare structural types are more likely to be placed a label than commonly seem ones.

8. Conclusion and future work

In conclusion, this study explored various methods in feature selection, sample balancing and multiclass classification aiming to predict structural type from a tax assessor building information. It shows that: none of weight rebalancing schemes improves prediction accuracy; chi-square ranking is a better method for feature selection among tested ones; k-nearest neighbor and neural network are the better classification algorithm. However, none of the model truly solve the problem of imbalanced classes and do not give satisfactory performance in prediction rare classes.

In the future, a search for statistical information on structural type distribution of the region would serve as a validation for the prediction result of the full building stock. More data about concrete, steel and masonry building in residential dominant areas could eb collected to improve classification in those structure types. More advanced wrapper type of feature selection could be used integrated into the classification model and more advanced weight penalty functions would also be embedded in the optimization process to yield better performed classifiers.

Project Code

https://github.com/MajorZengatStanfordedu/CS229-Project

Works Cited

[1] C. A. R. V. W. a. W. T. H. Kircher, " HAZUS earthquake loss estimation methods," *Natural Hazards Review,* vol. 7, no. 2, pp. 45-59, 2006.

[2] C. P. P. A. M. M. S. W. E. M. L. T. &. T. H. Geiß, "Estimation of seismic building structural types using multi-sensor remote sensing and machine learning techniques," *ISPRS journal of photogrammetry and remote sensing,* no. 104, pp. 175-188., 2015.

[3] V. Kumar, "Use sample weight in multi-label classification," [Online]. Available: https://stackoverflow.com/questions/49534490/use-sample-weight-in-multi-label-classification.

[4] scikit-learn, "sklearn.multiclass.OneVsRestClassifier," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html. [Accessed 01 06 2019].

[5] N. Stanford, "Chi2 Feature selection," [Online]. Available: https://nlp.stanford.edu/IR-book/html/htmledition/feature-selectionchi2-feature-selection-1.html.

[6] scikit.learn, "sklearn.feature_selection.chi2," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html#sklearn.feature_selection.chi2.

[7] S. NLP, "Mutual-Information," [Online]. Available: https://nlp.stanford.edu/IR-book/html/htmledition/mutual-information-1.html#mifeatsel.

[8] L. E. Peterson, "K-nearest neighbor," Scholarpedia 4.2.

[9] scikit-learn, "class sklearn.neighbors.KNeighborsClassifier," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier.

[10] S. R. a. D. L. Safavian, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics* , no. 21.3 , pp. 660-674, 1991.

[11] A. Navlani, "Decision Tree in Python," [Online]. Available: https://www.datacamp.com/community/tutorials/decision-tree-classification-python.

[12] scikit-learn, "sklearn.tree.DecisionTreeClassifier," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.

[13] scikit-learn, "sklearn.ensemble.RandomForestClassifier," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier.

[14] scikit-learn, "sklearn.neural_network.MLPClassifier," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html.

[15] scikit-learn, "sklearn.feature_selection.mutual_info_classif," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html#sklearn.feature_selection.mutual_info_classif.