Strategic Identification of Health Professional Shortage Areas (HPSA) Using Supervised Learning

Arul Bharthi
aarul@sfu.ca

Kanika Sanduja
ksanduja@sfu.ca

Namita Shah
namitas@sfu.ca

Supreet Takkar
stakkar@sfu.ca

Vishal Shukla
vshukla@sfu.ca

Abstract

Health Professional Shortage Areas (HPSAs) are designations that indicate health care provider shortages in Primary care; Dental health; or Mental health. HPSA identification is one of the ancillary process of the United States government to maintain sustained health condition throughout the country. The proposed model will classify whether a given county is a health professional shortage area or not provided the health indicator features of that locality.

1 Data Understanding

The dataset was taken from Community Health Status Indicators from the healthdata.gov. We corralled several CSV files and extracted the meaningful features that would be helpful in classifying a given county as HPSAs or not. The dataset contains records of 3007 Counties, 64 Parishes, 18 organized boroughs, 11 census areas, 41 independent cities and the District of Columbia for a total of 3142 counties and county equivalent. This model can be used to reevaluate a newly formed county's HPSA status in the future.

2 Data Preprocessing

2.1 Feature Extraction

The features were extracted from various datasets that included demography, measures of health, leading causes of death, infant mortality, vulnerable population, environmental health, preventive service use, and health risk factors information. The master data set was made by joining all these details into a single file that consists of 42 features.

2.2 Feature Engineering

While merging the features from several datasets, there were several features that needed to be engineered into a single primary attribute: For percentage variables such as Obesity (Percentage of obese people in the county), only the min and max confidence interval values were given. So, the feature Obesity was engineered by taking the average of the min and the max confidence intervals. This was done for other features also. For many diseases such as Measles, Hepatitis, Flu, etc., 3 different counts were given: Number of Reported Cases, No. of Cases with indication of that disease and the Number of expected cases. One feature for one particular disease was engineered and a single variable was created with the formula **Number of Reported Cases + MAX (Number of people with indicators, Number of expected people)**, thereby creating a single continuous variable which could reflect the actual impact of a disease.

2.3 Data Preparation and Cleaning

2.3.1 Missing Value Imputation Using Random Forest

The dataset consisted of several missing values in most of the continuous columns. Since the dataset contained all the county's features as a single row, it became essential that all the rows be present and therefore, missing values were to be imputed properly to generate a good model. Instead of imputing the missing values with mean value, we predicted the values and imputed them random forest. The values for the missing values were predicted with a error of 11 %.

2.3.2 Outlier Treatment using Capping

Outliers were identified in the few of the continuous variables in the dataset. Variables such as Elderly Medicare, Primary Care Rate, % of Uninsured people consisted of outliers and they were capped using the upper whisker value of the box chart.

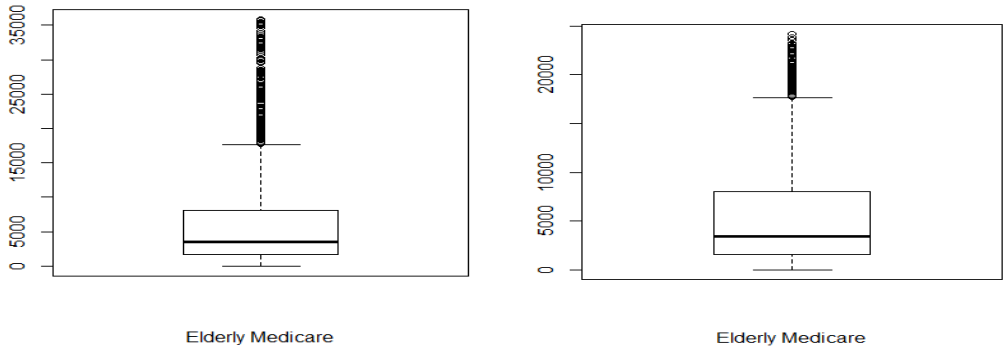
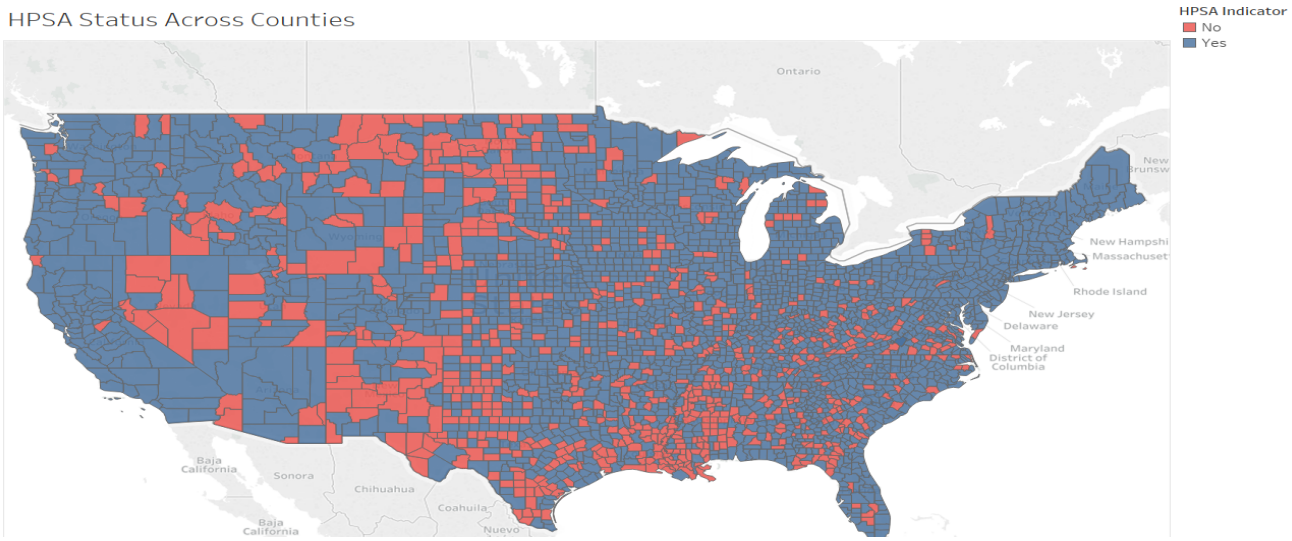


Figure 1: Elderly Medicare - Before and Outliers Removal

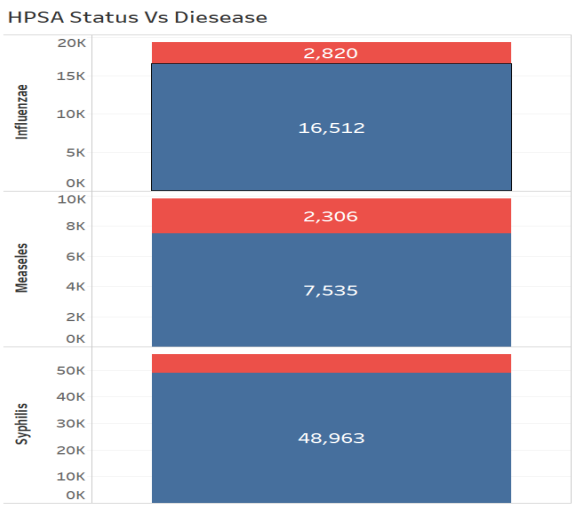
3 Exploratory Data Analysis

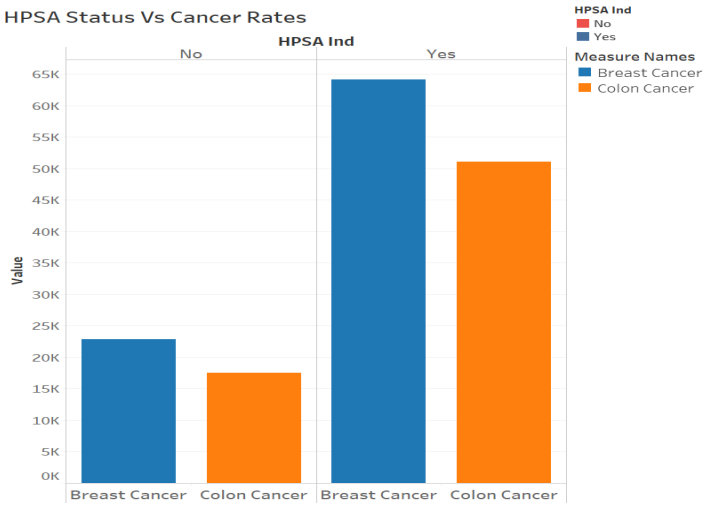
3.1 Target Variable - Geographical Analysis



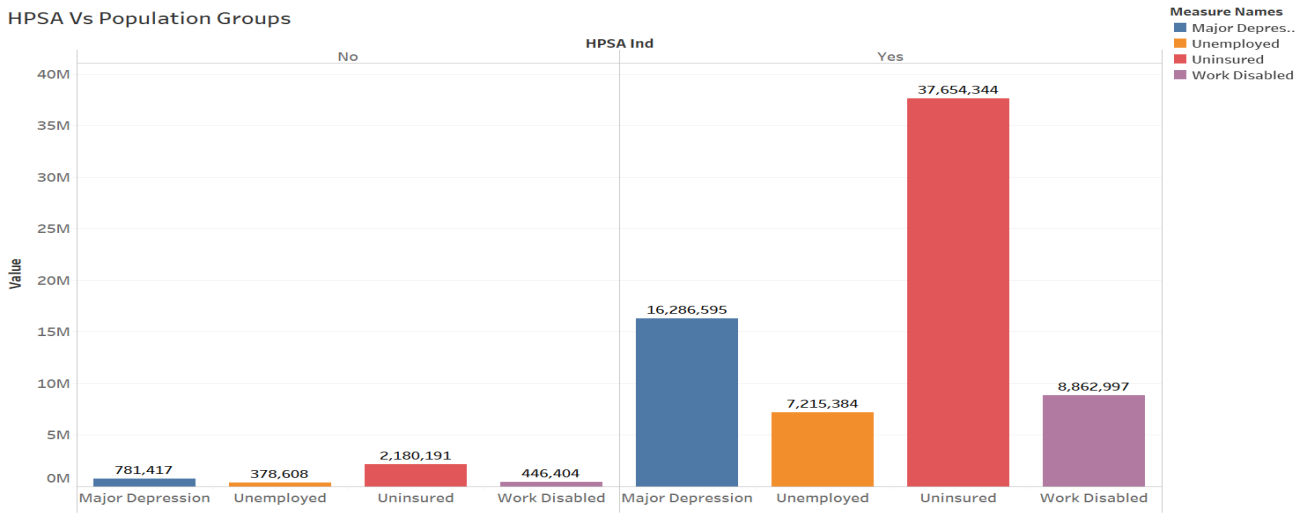
3.2 Multivariate Analysis on Target Variable

The presence of diseases and cancer seems to have a direct impact on the status of HPSA. The counties which are HPSAs seem to have high number of disease reported counts and cancer counts. Also, the number of population with major





depression, unemployment, uninsured life, severe work disablement seem to affect significantly the HPSA status of a county. The increase of number of these group of population is a significant marker for HPSA status of that county.



4 Model Preparation

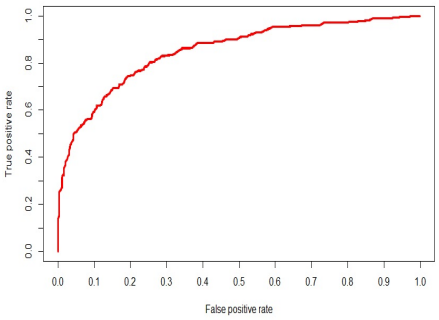
4.1 Logistic Regression

4.1.1 Model Building

After getting the initial idea about the significance of variables using exploratory data analysis, the information value of each feature was calculated using weight of evidence analysis. We built an initial generalized linear model was built with binomial family using all the features in the dataset.The step() function was used to perform variable selection in the initial

model. This function works by comparing the AIC (Akaike information criterion) improvements. It drops/adds the variable, thereby leading to the best AIC improvement.
Post this, the insignificant variables were removed from the model using the p-value as the deciding factor. If p-value is greater than 0.05 then the variable is ignored. Multicollinearity between variables is removed using the Variance Inflation Factor, VIF() function. Since the inputs given to the model may not be independent of each other, they have the effect of increasing the variance of model coefficients. The variance was needed to be decreased to make a model robust. The VIF for a predictor variable can be calculated by taking a linear regression of that predictor variable on all other predictor variables in the model. From that fit we pulled the R^2 value.
$VIF_i = \frac{1}{(1-R_i^2)}$, where VIF is the Variance Inflation Factor for variable i, R_i^2 is the Coefficient of Determination for a model where the i^{th} variable is fit against all other predictor variables in the model. The idea was to remove the variables with maximum VIF and which are insignificant (denoted by high p-values). This process is iterative. Then we re-created the model and repeated the process again till all variables in the model had a VIF value below a certain threshold value (in our case 10).
Final List of Significant Variables - % of People with Obesity, % of People Uninsured, Primary Medical Care Rate, Dentist Care Rate, Recent Drug Use, E.coli Infection, Elderly Medicare Rate.

4.1.2 Model Evaluation



Evaluation Metric	Values (%)
Accuracy	81.53
False Positive Rate	38.5
False Negative Rate	11.9
Sensitivity	87.5
Specificity	62.5
Area Under Curve	85

Figure 2 & Table 1: Model Evaluation: Logistic Regression

4.2 Support Vector Machine

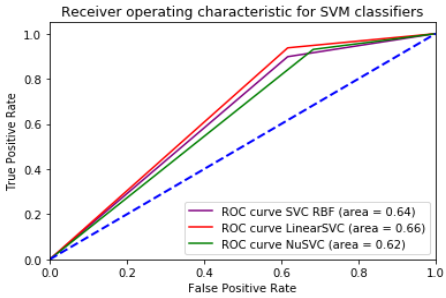
4.2.1 Model Building

We tried three SVM models:
The first model was prepared using SVC with kernel as Radial basis function. The parameters, C and gamma were tuned to train the model. C finds a hyperplane that correctly separates as many instances as possible and gamma is the 'spread' of the kernel. When gamma is low, the 'curve' of the decision boundary is very low and thus the decision region is very broad. When gamma is high, the 'curve' of the decision boundary is high, which creates islands of decision-boundaries around data points.
The second model was prepared using LinearSVM having kernel as 'linear'. The parameters, loss and penalty were tuned to train the model. Loss specifies the loss function. We used 'squared_hinge' as the square of the hinge loss. Penalty was used to penalise the parameter. We used L2-ridge loss.
The third model is trained using Nu-SVM with kernel as polynomial . The parameters, nu and degree are tuned to train the model. Nu parameter sets the number of support vectors. Nu was set to 0.399 with third degree polynomial kernel.

4.2.2 Model Evaluation

SVC with Radial Basis Function
Parameters: SVC(C=8, gamma=0.035625 , kernel='rbf')
Linear SVC
Parameters: LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True, intercept_scaling=1, loss='squared_hinge', max_iter=10000, penalty='l2', random_state=0, tol=0.0001,verbose=0)
Nu SVC

Parameters: NuSVC(cache_size=200, class_weight =None, coef0 = 0.0, decision_function_shape='ovr', degree=2, gamma=0.1, kernel='poly', max_iter=-1, nu=0.399, probability=False, random_state=None, shrinking=True, tol=0.001, verbose=False)



Evaluation Metric	SVC RBF (%)	Linear SVC (%)	Nu SVC (%)
Accuracy	80.38	80.38	78.261
False Positive Rate	61.67	61.67	59.01
False Negative Rate	62.84	62.84	68.28
Sensitivity	93.71	93.71	93.01
Specificity	38.32	38.32	31.71
Area under curve	64	66	62

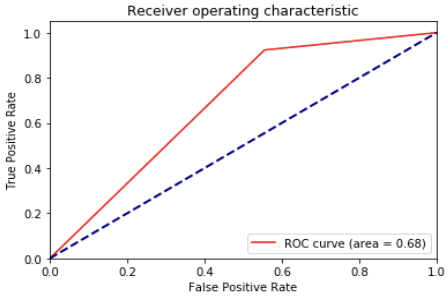
Figure 3 & Table 2: Model Evaluation: A comparison of SVM Techniques

4.3 Neural Network

4.3.1 Model Building

Implemented a feed-forward Neural Network using the Keras library with one input layer, one output layer and one intermediate layer. The parameters like activation function, loss function, epochs, optimiser were tuned to get the best training model with maximized accuracy and precision on the held out test set. Used only one hidden layer with 19 neurons, the input layer with 38 neuron and the output layer with 1 neuron. 'Sigmoid' activation function was used for the hidden layer and output layer. For loss function, applied 'binary cross entropy' and the optimizer used was 'adadelta'. Used adadelta as it is an adaptive learning rate method which uses exponentially decaying average of gradients. Epochs were kept as low as 160 with a batch size of five per iteration.

4.3.2 Model Evaluation



Evaluation Metric	Values (%)
Accuracy	80.80
False Positive Rate	51.10
False Negative Rate	10.05
Sensitivity	89.94
Specificity	48.89
Precision	84.73
Area under curve	68

Figure 4 & Table 3: Model evaluation for neural network

4.4 Boosting

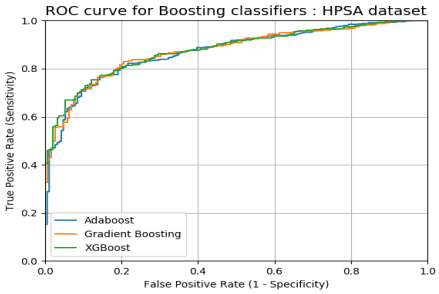
The technique of Boosting, which works on the Ensemble Principle, was applied to our dataset using three boosting algorithms: **Adaptive Boosting** or **Adaboost**, **Gradient Boosting Machines** or **GBMs** and **'XGBoost'**, the details of which are discussed below.

4.4.1 Model Building

The following parameters were taken up after a grid-search:
Adaboost :
num_trees = 40 (Number of trees), learning_rate=0.5 (between 0 and 1) and algorithm = 'SAMME.R' (of the two available algorithms: 'SAMME.R','SAMME')

GBMS :
n_estimators = 50 (number of estimators were tried between 30-250), learning_rate=0.2 (default is 0.1), max_depth = 3, presort='auto' (Presort options available were 'auto', 1 and 0), loss = 'deviance' (available options were 'deviance', 'exponential')
XGBoost
The best model obtained was with the implementation of XGBoost with these hyperparametes: max_depth = 3, learning_rate = 0.1, n_estimators = 60,
objective = 'binary:logistic', booster = 'gbtree' (after trials with gbtree gblinear and dart), max_delta_step = 1 (Maximum delta step we allow each tree's weight estimation to be), subsample = 1, reg_alpha = 1 (the L1 regularization term on weights), reg_lambda = 0 (the L2 regularization term on weights)

4.4.2 Model Evaluation



Evaluation Metric (%)	Adaboost	Gradient Boosting	XGBoost
Accuracy	83.21	81.30	83.84
False Positive Rate	46.56	44.97	40.21
False Negative Rate	7.37	10.39	8.54
Sensitivity	92.63	89.61	91.46
Specificity	53.44	55.03	59.79
Area Under the Curve	87.28	88.05	88.57

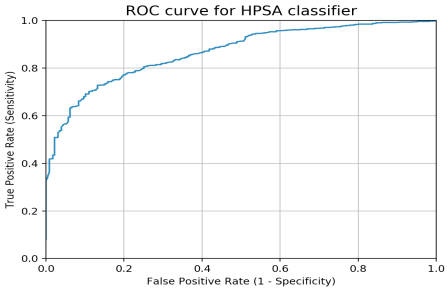
Figure 5 & Table 4: Model Evaluation: A comparison of boosting techniques

4.5 Random Forest

4.5.1 Model Building

After initial preprocessing of steps, the model was built using grid search for finding best parameters. Used 10 K fold cross validation for getting best accuracy. Different parameters were given for grid search. The following are best parameters
1. max_features = 'auto': This automatically takes all the features which make sense in every tree.
2. n_estimators = 300 : This is number of trees, it gives good performance, doesn't make the code slow and makes predictions stronger and more stable.
3. max depth = 10 : Max depth of the tree
4. criterion = 'gini': It measure the quality of a split. Gini for gini impurity.

4.5.2 Model Evaluation



Evaluation Metric	Values (%)
Accuracy	84.94
False Positive Rate	44.49
False Negative Rate	5.73
Sensitivity	94.27
Specificity	55.50
Precision	86.98
Area under curve	87.8

Figure 6 & Table 5: Model Evaluation: Random Forests

4.6 Other Models

4.6.1 KNN

Initial preprocessing and standardization of the variables were made to the dataset so that K-Nearest Neighbors algorithm could be executed. The KNN classification model was built by calculating the optimal value of K using 10-fold cross validation. The optimal value of K was found out to be 22. The model was then trained using the optimal K value and was used to predict the target indicators in the test dataset. The classification accuracy and the sensitivity of the model was found to be 76% and 98% respectively. But the model gave poor specificity.

4.6.2 Naive Bayes

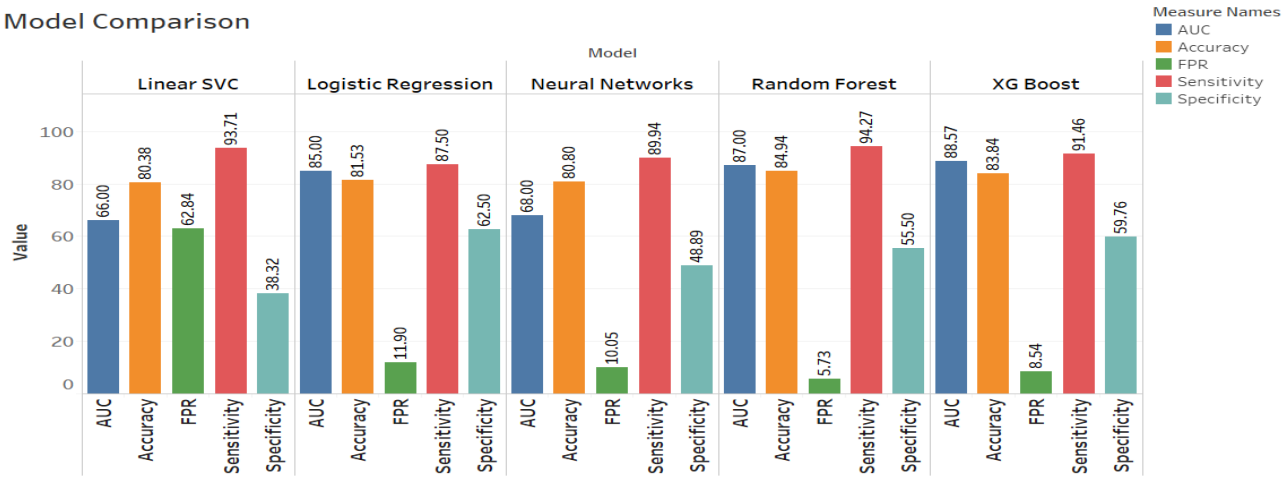
Used Naive Bayes model from scratch to calculate probability. Also tested the same model using scikit-learn to build mutinomialNB model for discrete data and gaussianNB model for continuous data. Then, probability was calculated for each model to get the accuracy. The accuracy obtained was 46.77% and Precision as 98.20%. Naive Baye's is simple classifier known for doing well when only a small number of observations are available. Since our data is big, Naive Bayes could not perform well even though it obtained a decent precision.

5 Model Comparison and Analysis

Though there are standard parameters to evaluate model performance, the perspective of our evaluation was based on the domain knowledge and problem statement involved in the analysis. In our case, it was more essential that an area with absolute medical shortage be identified as HPSA than a non-HPSA area be identified as non-HPSA, since medical assistance should be accurately provided to an area where there is critical and absolute need. Therefore, we factored having a 'Low False Negative Rate' as an important parameter along with Accuracy, Sensitivity, Specificity and Area Under the Curve(AUC).

Best Model(in terms of overall efficacy) - Random Forest (Bagging) & XG Boost(Boosting)

Best Model (in terms of a stable efficacy) - Logistic Regression (Generalized Linear Model)



6 Contributions

6.1 Group Contributions

Data Preprocessing - Feature Extraction & Engineering, Outlier Treatment, Missing Value Imputation, Standardization
Exploratory Data Analysis - Univariate Analysis, Multivariate Analysis, Data Visualization

6.2 Individual Contributions

Arul Bharathi - Logistic Regression, K-NN
Kanika Sanduja - Support Vector Machines
Namita Shah - Random Forest, Naive Bayes
Supreet Kaur Takkar - Boosting Algorithms
Vishal Shukla - Neural Networks

7 Links and References

Data-set: The HPSA dataset was obtained from **here.**

Code Repository: The link to our code repository is **here.**