IDENTIFYING POVERTY-DRIVEN NEED BY AUGMENTING CENSUS AND COMMUNITY SURVEY DATA

by

KEERTHI KORIVI

B.E. , Osmania University, India, 2012

A REPORT

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department Of Computer Science
College Of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2016

Approved by:

Major Professor
Dr. William H. Hsu

Copyright

Abstract

Need is a function of both individual household's ability to meet basic requirements such as food, shelter, clothing, medical care, and transportation, and latent exogenous factors such as the cost of living and available community support for such requirements. Identifying this need driven poverty helps in understanding the socioeconomic status of individuals and to identify the areas of development. This work aims at using georeferenced data from the American Community Survey (ACS) to estimate baseline need based on aggregated socioeconomic variables indicating absolute and relative poverty. In this project, I implement and compare the results of several machine learning classification algorithms such as Random Forest, Support Vector Machine, and Logistic Regression to identify poverty for different block groups in the United States.

Table of Contents

Table of Contents viii

List of Figures x

List of Tables xi

Acknowledgements xi

1 Introduction 1

1.1 Problem Definition 1

1.2 Goal and Technical Objective 2

1.3 Overview 2

2 Background and Related Work 4

2.1 Literature Survey 4

2.2 Established Methods and Evaluation Approaches 5

2.2.1 Classification Algorithms 5

3 Implementation 13

3.1 Dataset Preparation 13

3.1.1 Data Preprocessing 13

3.1.2 Data Overview 14

3.2 Implementation steps 15

4 Experiments **17**

4.1 Training and Test Datasets 17

4.2 Experimental Design 18

4.2.1 Support Vector Machines 18

4.2.2 Logistic Regression 18

4.2.3 Random Forests 18

5 Results **20**

5.1 Performance Metrics 20

5.1.1 Confusion Matrix 20

5.1.2 Precision and Recall 20

5.1.3 F1-score 21

5.1.4 ROC Curve 21

5.2 Experimental Results 21

5.2.1 Support Vector Machines 21

5.2.2 Logistic Regression 23

5.2.3 Random Forests 24

5.2.4 SVM Vs Logistic Regression Vs Random Forests 25

5.2.5 Support Vector Machines 25

6 Conclusion and Future Work **26**

6.1 Conclusion 26

6.2 Future Work 27

Bibliography **28**

A Technologies **30**

List of Figures

2.1 Illustration of the binary classification technique 5

2.2 Illustration of Random Forest classification technique 7

2.3 Illustration of an optimal hyperplane 9

2.4 Illustration of graphical representation of Logistic Regression 11

3.1 Illustration of implementation of poverty prediction system 15

5.1 Receiver Operating Curve for Linear SVM 22

5.2 Receiver Operating Curve for Logistic Regression 23

5.3 Receiver Operating Curve for Bagged Random Forest 24

5.4 Illustration of Number of Trees in Random Forest Vs F1-Score 25

5.5 Illustration of Learning Algorithm Vs F1-score 25

A.1 List of algorithms in scikit-learn package 30

List of Tables

5.1 Performance metrics for Linear SVM 22
5.2 Confusion Matrix for Linear SVM 22
5.3 Performance metrics for Logistic Regression 23
5.4 Confusion Matrix for Logistic Regression 23
5.5 Performance metrics for Bagged Random Forest 24
5.6 Confusion Matrix for Bagged Random Forest 24

Acknowledgments

I would like to express my deepest gratitude to my major professor Dr.William H.Hsu for his constant inputs, comments, and feedback while working on this project and also for his constant support during my Master's at Kansas State University. I would like to thank Dr.Katherine Nesse, Assistant Professor in the department of Landscape Architecture and Regional Community Planning for her domain expertise and initiation without which this project might not have happened. I would also like to thank Dr.Mitch Neilsen and Dr.Torben Amtoft for serving on my M.S. committee. Finally, I thank my family and friends for their constant love and support.

Chapter 1

Introduction

This chapter discusses the problem statement, goals, and overview of the work done in this project.

1.1 Problem Definition

Identifying poverty-driven need helps in understanding the socioeconomic conditions of individuals living in a neighbourhood. It is believed that crime or disorder in a neighbourhood is correlated with poverty level in that neighbourhood. Identifying poverty in the neighbourhood is a challenging task because it involves lot of manual effort to survey and several economic factors such as income, cost of living etc. should be considered to determine the poverty. Despite all these considerations, sometimes we only get a rough estimates of these factors. This problem is addressed by using machine learning algorithms which help in predicting poverty in a neighbourhood.

1.2 Goal and Technical Objective

The goal of this project is to identify the poverty level in the block groups which are part of Kansas state. Block Groups (BGs) are statistical divisions of census tracts, are generally defined to contain between 600 and 3,000 people, and are used to present data and control block numbering.[1] Machine learning techniques are applied on the American Community Survey data to predict poverty of block groups present in Kansas. Supervised learning algorithms such as Random Forests, Support Vector Machines and Logistic Regression are used for binary classification of poverty. American Community Survey data is used as a baseline to predict poverty. Since, the community survey is based on estimations, poverty prediction can be improved by using the neighbourhood images to identify any other signs of poverty using deep learning. Deep learning is used to identify objects such as broken window, grafitti etc. in the image which are some other signs of poverty. The overall goal is to use both the American community Survey data and the neighbourhood visuals to predict poverty. Using visual data, however, is out of the scope of this project.

1.3 Overview

Poverty is predicted based on the American Community Survey conducted on about 35000 block groups in the United States. The Department of Housing and Urban Development data is used as the ground truth which determines the actual poverty. Dr.Katherine Nesse, Assistant Professor in the department of Landscape Architecture and Regional Community Planning helped in identifying the features that are most important from a set of about 862 features and also identify the ground truth against which the performance of machine learning techniques are verified. Supervised machine learning technique build a model based on the training data provided and classifies the test data. These test data classifications are verified against ground truth to measure the performance of the classification models. Scikit-learn is used to build classification models. It is a machine learning package in Python helps

in data-mining and data analysis. Random Forests, Support Vector Machines and Logistic Regression are run on the American Community Survey dataset and the results of these algorithms are compared. It is observed that there is a scope of improvement based on confusion matrix, ROC curves of these algorithms.

Chapter 2

Background and Related Work

This chapter discusses the previous that has been done on poverty and the necessary background information such as the machine learning algorithms used for our proposed work.

2.1 Literature Survey

Previous work has focused on the identification of poverty at finer levels (households) and coarser levels (cities and districts) than a block group, and on the duration of poverty. In this work, the focus is on the identification of block groups with high percentages of households in poverty in a block group in order to support remediation goals.

The American Community Survey (ACS) provide information at the block group level, representing aggregate rates of poverty across individual households and blocks. The focus was on analytics using socioeconomic features, but a longer-term goal is to incorporate demographic information where available, including median age, education, and other information concerning households.

2.2 Established Methods and Evaluation Approaches

A Machine learning technique called supervised learning is used in this work. In the predictive or supervised learning approach, the goal is to learn a mapping from inputs x to outputs y, given a labeled set of input-output pairs D = $(x_i, y_i)_{i=1}^{N}$. Here D is called the training set, and N is the number of training examples.[2]

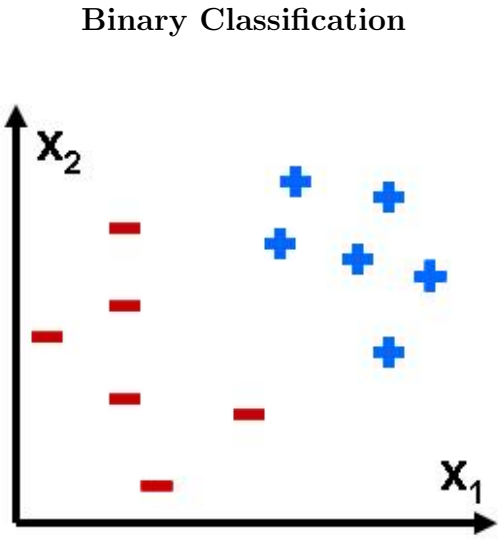

Figure 2.1: *Illustration of the binary classification technique. Adapted from(Lyle Ungar,2016)*

In this work, binary classification technique is used. In a binary classification, the number of output classes is two i.e C=2.

2.2.1 Classification Algorithms

There are many classification algorithms which can be used for binary classifcation like Decision Trees, Naives Bayes etc. But, there are certain draw backs like over-fitting using these algorithms. In this project, Logistic Regression, Linear SVM, and Random Forests

algorithms are implemented and their results are compared. Random forests algorithm works by constructing an ensemble of decision trees.

2.2.1.1 Random Forests

Random Forest is an ensemble learning methodology which follows a divide and conquer strategy where a group of weak learning models come together to form a strong model. Random Forest constructs multiple decision trees (Decision trees are predictive models that use a set of binary rules to calculate a target value)[3] by random sub-sampling of features and data set and takes the mode of the class of each individual tree predictions. The weak model is formed by the individual decision trees while the strong model formed is the group of this decision trees called Random Forests.[4]

Randomization is employed in two places in a random forest algorithm:

1. Random feature subset to select only a few features from the list of features present within the dataset and train on only those selected features.

2. Random sub sampling i.e selecting only a few records with replacement to train individual decision trees.

Building a Random Forest

Below is the step by step procedure involved in the random forest algorithm.

1. For b = 1 to B:

(a) Draw a bootstrap sample Z^* of size N from the training data.

(b) Grow a random forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n min is reached.

i. Select m variables at random from the p variables.

ii. Pick the best variable/split-point among them.

iii. Split the node into two daughter nodes.

2. Output the ensemble of trees T_b

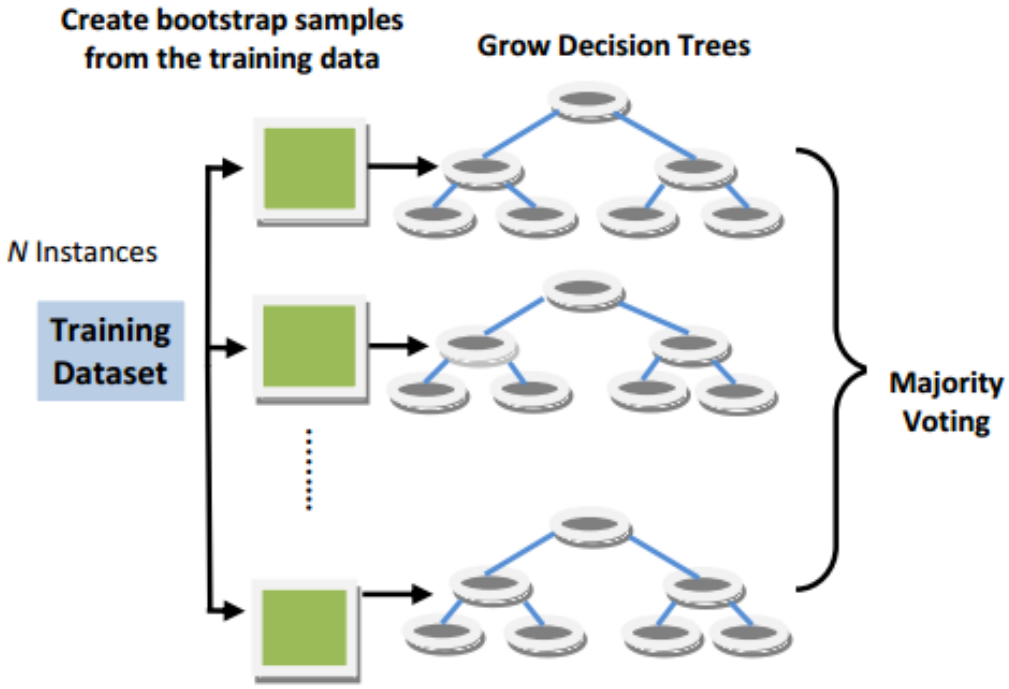

Figure 2.2: *Illustration of the Random Forest classification technique. Adapted from(Adnan and Asifullah Khan,2015)*

To make a prediction at a new point x:

Classification: Let C_b(x) be the class prediction of the random forest tree.

Then C^B_{rf}(x) = majority vote $C_b(x)^B_1$ [3]

Bagged Random Forest

Bagged Random Forest is a variation of Random Forest. The fundamental difference between them is that Bagged Random Forests construct fully grown trees considering all the features while splitting a node where as a Random forests consider only a few features for finding a split node.

2.2.1.2 Support Vector Machines

Support Vector Machines uses the training data which contains two or more categories to build a model which assigns the new incoming test data to any of the previous categories

by mapping non-linear input vectors into high dimensional spaces. SVM's are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.[5].

Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.[5]

Maximum Margin

In order to find a hyperplane that generalizes well, Vladimir proposed the concept of optimal hyperplane. An optimal hyperplane is defined as the linear decision function with maximal margin between the vectors of the two classes, see Figure. 3.2. It was observed that to construct such optimal hyperplanes one only has to take into account a small amount of the training data, the so called support vectors, which determine this margin. It was shown that if the training vectors are separated without errors by an optimal hyperplane the expectation value of the probability of committing an error on a test example is bounded by the ratio between the expectation value of the number of support vectors and the number of training vectors:[5]

$$E[Pr(error)] \leq \frac{E[number of support vectors]}{number of training vectors}$$

Linear SVM

Given a training dataset of n points of the form

$$\{(\vec{x}_1, y_1), (\vec{x}_2, y_2), ..., (\vec{x}_n, y_n)\}$$

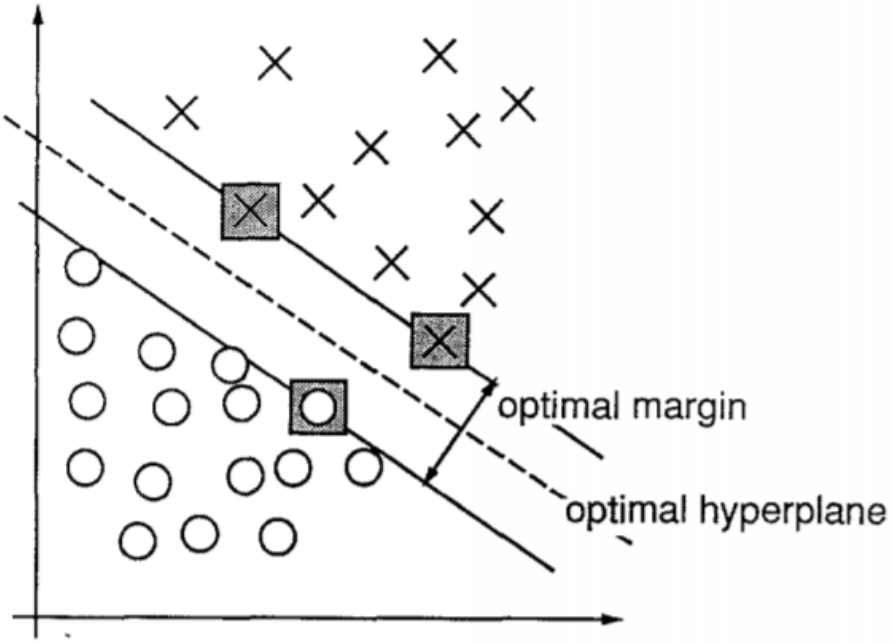

Figure 2.3: *An example of a separable problem in a 2 dimensional space. The support vectors, marked with grey squares, define the margin of largest separation between the two classes. Adapted from(Vladimir Vapnik and Corinna Cortes)*

where the y_i are either 1 or -1, each indicating the class to which the point $\vec{x}_i$ belongs. Each $\vec{x}_i$ is a p-dimensional real vector. We want to find the "maximum-margin hyperplane" that divides the group of points $\vec{x}_i$ for which $y_i = 1$ from the group of points for which $y_i = -1$, which is defined so that the distance between the hyperplane and the nearest point $\vec{x}_i$ from either group is maximized.[6]

Any hyperplane can be written as the set of points $\vec{x}$ satisfying

$$\vec{w} \cdot \vec{x} - b = 0,$$

where $\vec{w}$ is the (not necessarily normalized) normal vector to the hyperplane. The parameter $\frac{b}{\|\vec{w}\|}$ determines the offset of the hyperplane from the origin along the normal vector .[6]

If the training data are linearly separable, we can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible.

The region bounded by these two hyperplanes is called the "margin", and the maximum-margin hyperplane is the hyperplane that lies halfway between them. These hyperplanes can be described by the equations $\vec{w} \cdot \vec{x} - b = 1$ and $\vec{w} \cdot \vec{x} - b = -1$ .The distance between these two hyperplanes is $\frac{2}{\|\vec{w}\|}$, so to maximize the distance between the planes we want to minimize $\|\vec{w}\|$ [6]. To prevent data points from falling into the margin, for each i we can write

$$\vec{w} \cdot \vec{x}_i - b \geq 1, if y_i = 1$$

or

$$\vec{w} \cdot \vec{x}_i - b \leq -1, if y_i = -1.$$

This can be rewritten as: $(\vec{w} \cdot \vec{x}_i - b) \geq 1,$ for all $1 \leq i \leq n$.We can put this together to get the optimization problem:
"Minimize $\|\vec{w}\|$ subject to $y_i(\vec{w} \cdot \vec{x_i} - b) \geq 1, for i = 1, \ldots, n$" [6]

2.2.1.3 Logistic Regression

The goal of a logistic regression analysis is to find the best-fitting and most parsimonious, yet reasonable, model to describe the relationship between an outcome (dependent or response variable) and a set of independent (predictor or explanatory) variables. What distinguishes the logistic regression model from the linear regression model is that the outcome variable in logistic regression is categorical and most usually binary or dichotomous. [7]

Hypothesis Representation

The sigmoidal or logistic function is used to represent a hypothesis to make predictions. The predictions of this hypothesis is between 0 and 1 i.e $0 \leq h_\theta(x) \leq 1$

The hypothesis function $h_\theta(x)$ is defined as

$$h_\theta(x) = g(\theta^T x)$$

where is a column vector of parameters and x is a column vector of input variables/features. The sigmoidal/logistic function, g(z), where z is a real valued number is given by

$$g(z) = \frac{1}{1 + e^- z}$$

The graphical representation of sigmoidal function is as shown below

Graphical Representation of Logistic Regression

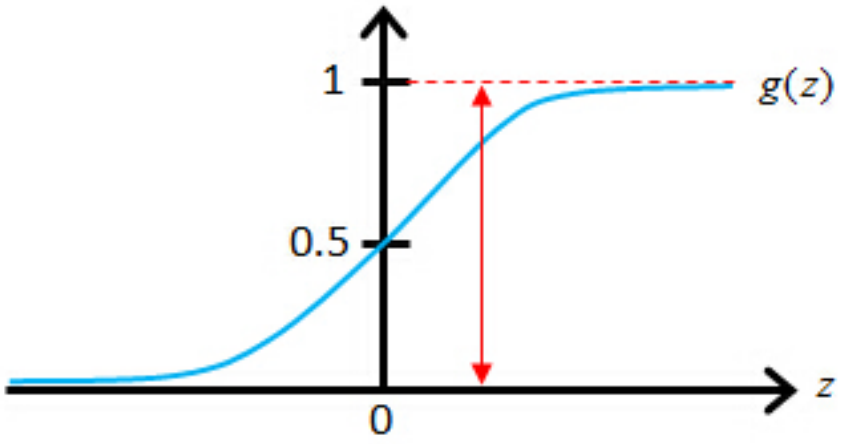

Figure 2.4: *Illustration of graphical representation of Logistic Regression.Adapted from(Andrew ng,2009 and iracer,2016)*

The hypothesis representation in terms of θ is as shown

$$h_\theta(\mathrm{x}) = \frac{1}{1 + e^{-\theta^T x}}$$

Interpretation of Hypothesis Output

The hypothesis output is treated as the estimated probability that y is equal to 1 on an input x, where y is the class label to be predicted i.e $h_\theta(x)$ = estimated probability that y = 1 for given input x.

Say for instance, if there is an input feature vector x and if the hypothesis outputs 0.8 that means the probability of the given input to be of certain category is 80%. So, when the probability is greater than 0.5 it means that the probability of being y=1 is greater. So, in such cases we predict y as 1 else y is 0. The decision boundary separates the two categories and helps in better understanding of what the hypothesis function is computing.

Chapter 3

Implementation

This chapter discusses the data-preprocessing, implementation steps of this project.

3.1 Dataset Preparation

American Community Survey and Department of Housing and urban Development data are in the form of a shape file which are converted into a csv format using ArcGis software.

3.1.1 Data Preprocessing

Data preprocessing and cleaning steps are conducted before the data is fed to the machine learning algorithm. It is an important step in eliminating unnecessary or redundant data to avoid misleading outcomes. Additionally, It brings down the size of the training data and thereby reduces the time required to build a learning model and to make predictions. Below are some of the considerations for data preprocessing.

Feature Selection The American Community Survey data contains about 862 columns/features. All these features are not of importance in determining poverty. Dr.Katherine Nesse, has identified the most important features in determining poverty with her domain expertise. The data overview section 3.1.2 gives details about these features.

Handling Missing Values Data acquired sometimes could be incomplete and may contain missing values. If this is not handled properly this could affect the performance of the model. It is made sure the data does'nt have any missing values. If there are any missing values, then either the data is ignored or is filled with dummy values which could be the mean or mode value of the attribute.

3.1.2 Data Overview

Each block group contains an ID called the GEOID. GEOIDs are numeric codes that uniquely identify all administrative/legal and statistical geographic areas for which the Census Bureau tabulates data. The state id for Kansas is 20. In our dataset

$$GEOID = STATE + COUNTY + TRACT + BLOCKGROUP$$

where each of the STATE, COUNTY, TRACT and BLOCKGROUP have a number assigned to them.[8]

American Community Survey contained the following fields:

GEOID: Geographic Identifier

C17002e1: Ratio of Income to poverty level in the past 12 Months: Total: Population for whom poverty status is determined(Estimated)

C17002e2: Ratio of Income to poverty level in the past 12 Months: Under .50: Population for whom poverty status is determined(Estimated)

C17002e3: Ratio of Income to poverty level in the past 12 Months: Under .50 to .99: Population for whom poverty status is determined(Estimated)

There are two possible class variables for each of these GEOID's: `low_poverty`, `high_poverty`. The ground truth for each of these GEOID's is obtained from Department of Housing and Urban Development(HUD) data.

The GEOID is in `high_poverty` if the $\frac{LOWMOD}{LOWMODUNIV}$ i.e `LOWMOD_PCT` $\geq$ 51 else its in `low_poverty`

Here, **LOWMOD:** Number of people below 80% of area median income (AMI – the median income in that metro area)

LOWMODUNIV: Number of people in the block group living in households

LOWMOD_PCT: Percent of people living below 80% of AMI $\frac{LOWMOD}{LOWMODUNIV}$

3.2 Implementation steps

The below illustration gives an overview of the difference steps performed in poverty prediction. Featuring engineering is done to identify the most relavent features and it is made

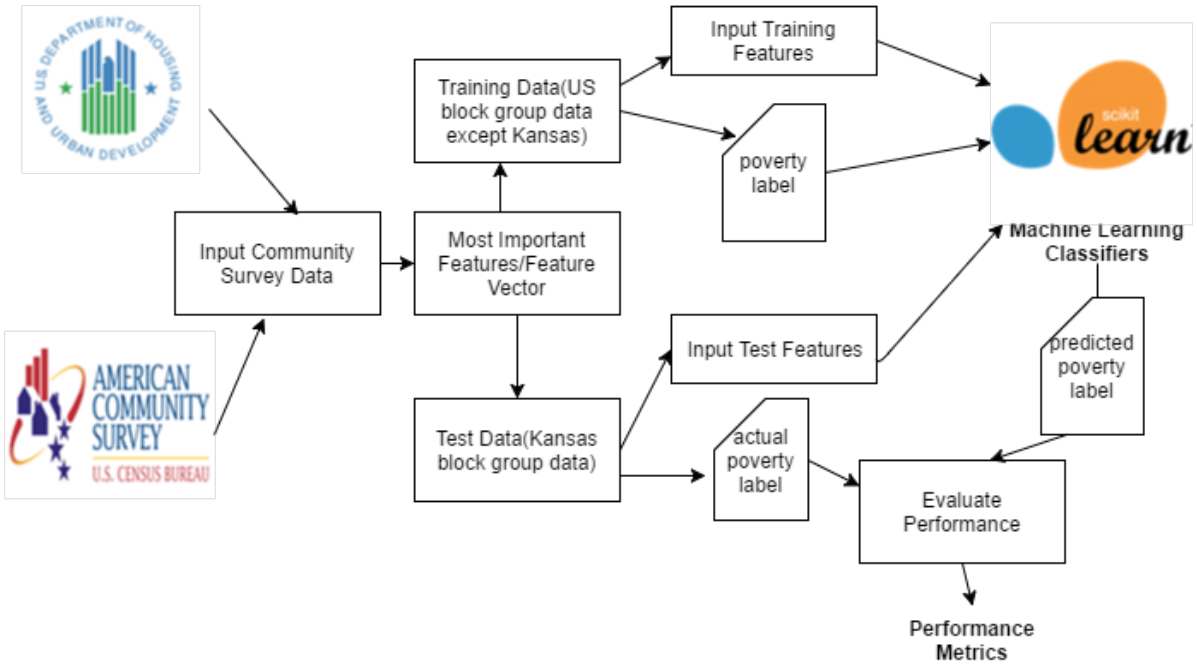

Figure 3.1: *Illustration of implementation of poverty prediction system*

sure that there is no noise and missing values in the data. The ACS data is joined with HUD data to get the feature vector and poverty label i.e ground truth together forming the poverty dataset which is split into training and test set. The training data is used to

build the classifier which makes predictions when the test data is passed. The predictions are then compared with the actual label to determine the performance of the classifiers.

Chapter 4

Experiments

This chapter explains the dataset used and the experiments conducted on the dataset using the various approaches mentioned in sections 2.2.1.1, 2.2.1.2, 2.2.1.3

4.1 Training and Test Datasets

In this project the goal is to identify the poverty of block groups in Kansas State. So, the training data contains all the block groups in the United States except block groups in Kansas. While, the test data contains all the block groups in Kansas.

The training data is used to build the model and the test data is used to determine the performance of the classification model.

- Total dataset **size** = 34933
- Training data **points** = 34251
- Test data **points** = 682

4.2 Experimental Design

This section explains the experiments that were conducted on the dataset using several machine learning classification algorithms mentioned in the section 2.2.1.1, 2.2.1.2, 2.2.1.3 All the algorithms were implemented using scikit-learn : A machine learning package in Python. The performance of the classification models is determined by the F1-score which is the harmonic mean of precision and recall. precision and recall are the measures which depend on true positive,true negative ,false positive and false negative values.

4.2.1 Support Vector Machines

Implemented Support Vector Machines using the sklearn SVM classes i.e LinearSVC, SVC. My experiments show that the algorithm works best for LinearSVC when compared to Non-linear methods. Verified the performance of several kernel functions like poly, sigmoid, rbf etc.

4.2.2 Logistic Regression

Implemented Logistic Regression in sklearn using the sklearn.linear_model.LogisticRegression class[9]. The fit method is used to fit the training data and to construct a model. The predict method is used to make actual predictions.

4.2.3 Random Forests

Implemented random forest algorithm sklearn's sklearn.ensemble.RandomForestClassifier.[10] Evaluated the performance of scikit-learn by tuning the number of trees in ensemble, maximum features used to construct a decision tree in the ensemble. Conducted several experiments for different number of trees that are used in the random forest ensemble. Our experiments show that the for the tree size equal to 100 achieves higher AUC.
Also, implemented Bagged Random Forest which also performs equally well this is due to the

fact that our input dataset is subsampled into several trees. In a bagged random features, we use all the features and build trees by sub-sampling input data.

Chapter 5

Results

This chapter explains the results of several experiments performed in the section 2.2.1.1, 2.2.1.2, 2.2.1.3

5.1 Performance Metrics

Below metrics are considered to evaluate the performance of the classification model

5.1.1 Confusion Matrix

A Confusion matrix for a model is a matrix which has the true positive, true negative, false positive and false negative values for the given test data.

5.1.2 Precision and Recall

Precision is the fraction of retrieved instances that are relevant. Precision is the ratio of True Positives to the sum of True Positive and False Positive.[11];[12]

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

Recall is the fraction of relevant instances that are retrieved. Recall, also called true positive rate or sensitivity is the ratio of True Positive to the sum of True Positive and False Negative.[11];[12]

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

5.1.3 F1-score

F1 score is a measure of the test's accuracy. It is calculated using the precision and recall of the test data. F1-score is also defined as the harmonic mean of precision and recall[12]

$$F1 - score = \frac{2 * PrecisionRecall}{Precision + Recall}$$

5.1.4 ROC Curve

A receiver operating curve is a graphical plot between the true positive rate i.e recall also called sensitivity and false positive rate also know as fall-out or false alarm. Thus, the ROC curve is thus the sensitivity as a function to fall-out. The quality of ROC is often summarized as a single number using the area under the curve or AUC. Higher AUC scores are better and the perfect score is its maximum value 1.[13];[12];[14]

5.2 Experimental Results

This section discusses the performance metrics obtained for each of the classification models discussed in section 2.2.1.1, 2.2.1.2, 2.2.1.3

5.2.1 Support Vector Machines

Several experiments were conducted using the SVM's using linear, sigmoid, poly, rbf kernel. It is observed that the best performance is obtained for linear kernel.

The tables 5.1,5.2 show the performance results of Linear Support Vector Machine

Performance Measure	Value
Accuracy	0.75
Precision	0.71
Recall	0.71
F1 score	0.71

Table 5.1: *Performance metrics for Linear SVM*

	low_poverty	*high_poverty*
low_poverty	387	83
high_poverty	84	128

Table 5.2: *Confusion Matrix for Linear SVM*

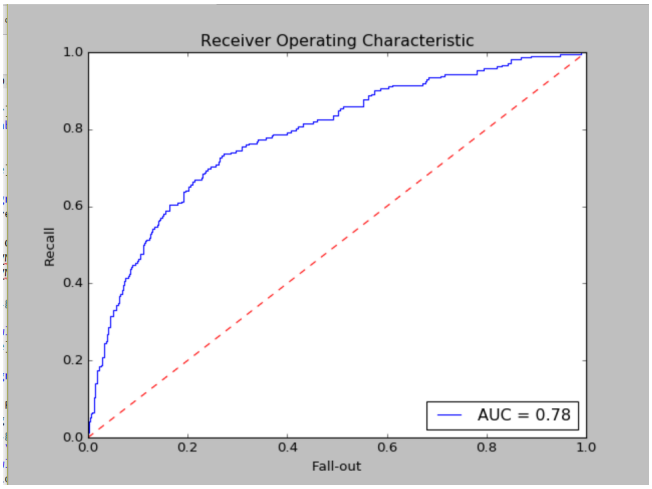

Figure 5.1: *Receiver Operating Curve for Linear SVM*

5.2.2 Logistic Regression

The tables 5.3,5.4 show the performance results of logistic regression

Performance Measure	Value
Accuracy	0.78
Precision	0.79
Recall	0.68
F1 score	0.69

Table 5.3: *Performance metrics for Logistic Regression*

	low_poverty	*high_poverty*
low_poverty	449	21
high_poverty	126	86

Table 5.4: *Confusion Matrix for Logistic Regression*

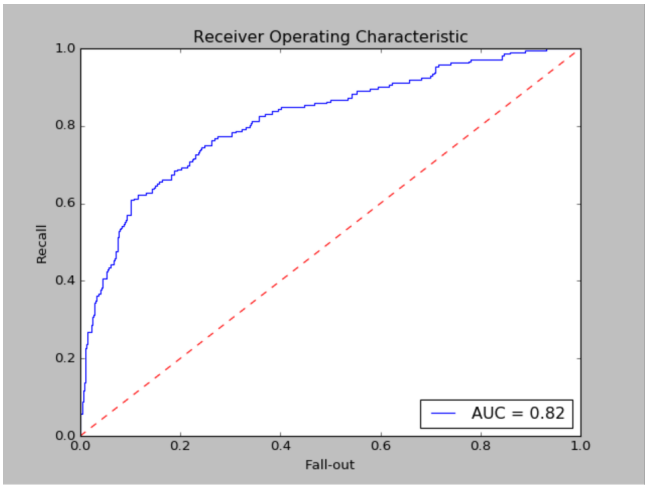

Figure 5.2: *Receiver Operating Curve for Logistic Regression*

5.2.3 Random Forests

Several experiments were conducted on random forests by tuning parameters like number of trees, split criteria and also on the number of features. A variation of random forest called bagged random forest is also implemented.

It is observed that random forest worked well when the number of trees were more than 100 and for bagged random forest.

The tables 5.5,5.6 show the performance results of Bagged Random Forests

Performance Measure	Value
Accuracy	0.79
Precision	0.77
Recall	0.72
F1 score	0.73

Table 5.5: *Performance metrics for Bagged Random Forest*

	low_poverty	*high_poverty*
low_poverty	431	39
high_poverty	103	109

Table 5.6: *Confusion Matrix for Bagged Random Forest*

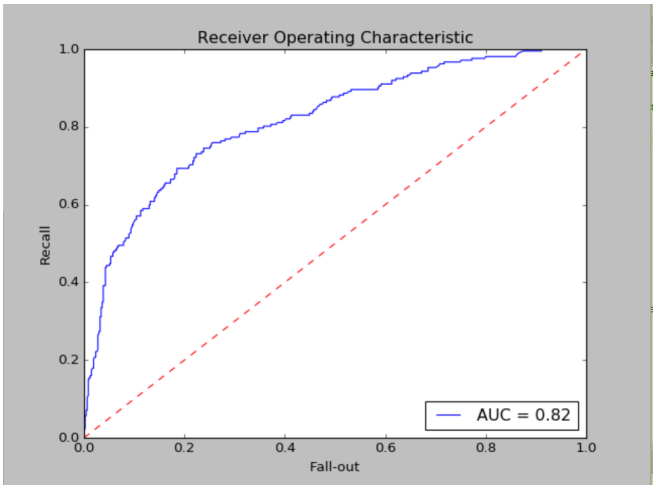

Figure 5.3: *Receiver Operating Curve for Bagged Random Forest*

Number of Trees in Random Forest Vs F1-Score

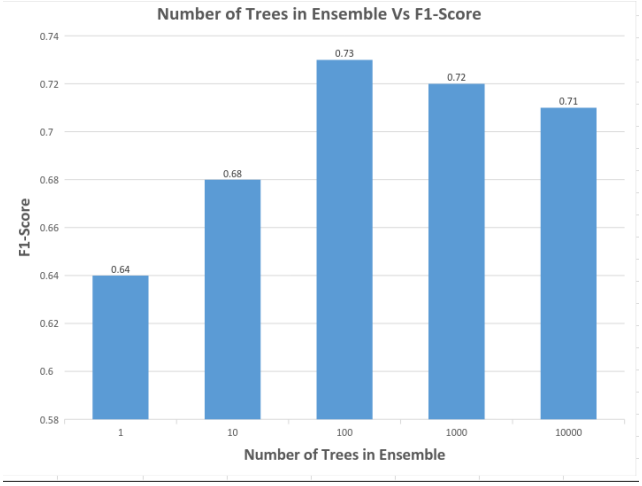

Figure 5.4: *Illustration of Number of Trees in Random Forest Vs F1-Score*

5.2.4 SVM Vs Logistic Regression Vs Random Forests

The bar chart in Figure 5.5 below, shows the F1-scores for different learning algorithms implemented and described in section 2.2.1.1, 2.2.1.2, 2.2.1.3

5.2.5 Support Vector Machines

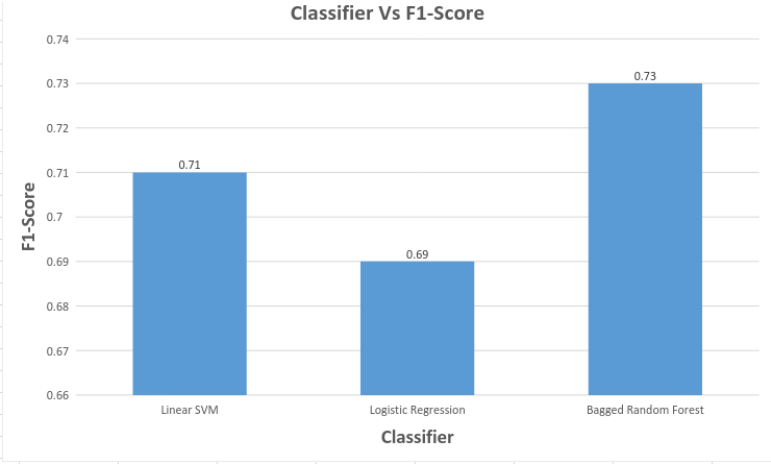

Figure 5.5: *Illustration of Learning Algorithm Vs F1-score*

Chapter 6

Conclusion and Future Work

This chapter draws conclusions out of the results presented in section 5.2.3, 5.2.1, 5.2.2

6.1 Conclusion

Random Forests outperform Logistic Regression and Support Vector Machines. It is observed that random forest classifier predicts well and has better performance metrics. The AUC and F1-score for Bagged Random Forest are better that mean it does a really good job in classifying the instances correctly with high recall. Logistic regression also does good work in classifying the instances but its recall is comparatively lower. It is also observed that random forests are relatively faster when compared to other algorithms. Random Forests are scalable and can be used with huge dataset and dimensions/features, since random forests can easily be run in a distributed environment and there is no data dependency between different trees that are formed as part of the random forest ensemble. The best accuracy is 79% for Bagged Random Forest. There is about 20% misclassification rate in Bagged Random Forest. This shows the need for improvement, which can be done by using the visual features in a neighbourhood.

6.2 Future Work

Since this data comes from a community survey, all the households might not have been considered for producing the survey data. Augmenting visual features to the community survey data can improve the performance of our model and also improves the accuracy of poverty prediction. The visual features like broken windows, peeling paint, graffiti could be obtained from the google street view. Presence of these signs could be an indication of poverty and can help us determine the block groups in poverty. However, sometimes it could be the case where people were just lazy and couldn't have got their window fixed. In Addition, some places could be abandoned and could contain broken windows and there might not be anyone living in such places. Such scenarios could be considered as false positive and these scenarios should not be handled while determining poverty.

Bibliography

[1] ACS. Geographic terms and concepts - block groups. https://www.census.gov/geo/reference/gtc/gtc_bg.html/, 2010.

[2] Kevin P. Murphy. *Machine Learning : A Probabilistic Perspective*. MIT Press, 2013.

[3] Ned Horning. Random forests : An algorithm for image classification and generation of continuous fields data sets. 2010.

[4] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2014.

[5] Corinna Cortes and Vladmir Vapnik. Support-vector networks. *Kluwer Academic Publishers*, pages 273–282, 1995.

[6] Olexa Riznyk. Support Vector Machine. https://en.wikipedia.org/wiki/Support_vector_machine/, 2016. [Online; accessed 30-October-2016].

[7] N Balakrishnan. *Methods and Applications of Statistics : Methods and Applications of Statistics in Clinical Trials, Volume 2 : Planning, Analysis, and Inferential Methods*. Wiely, 2014.

[8] ACS. Understanding geographic identifiers (geoids). https://www.census.gov/geo/reference/geoidentifiers.html/, 2015.

[9] Trent Hauck. *Scikit-learn Cookbook : over 50 recipes to incorporate scikit-learn into every step of the data science pipeline, from feature extraction to model building and model evaluation*. Packt Publishing, 2014.

[10] Gavin Hackeling. *Mastering machine learning with scikit-learn apply effective learning algorithms to real-world problems using scikit-learn*. Packt Publishing, 2014.

[11] Wikipedia. Precision and recall. https://en.wikipedia.org/wiki/Precision_and_recall/, 2016.

[12] Jose A. Lozano, Guzmn Santaf, and Iaki Inza. Classifier performance evaluation and comparison. *International Conference on Machine Learning and Applications*, 2010.

[13] Wikipedia. Receiver operating curve. https://en.wikipedia.org/wiki/Receiver_operating_characteristic/, 2016.

[14] Tom Fawcet. Roc graphs: Notes and practical considerations for researchers. *International Conference on Machine Learning and Applications*, HP Laboratories, 2004.

Appendix A

Technologies

Scikit-learn is a Python open source module for machine learning built on top of SciPy and distributed under the 3-Clause BSD license.The project was started in 2007 by David Cournapeau as a Google Summer of Code project, and since then many volunteers have contributed. It has simple and efficient tools for data mining and data analysis.
Scikit-learn requires Python,Scipy,Numpy packages.

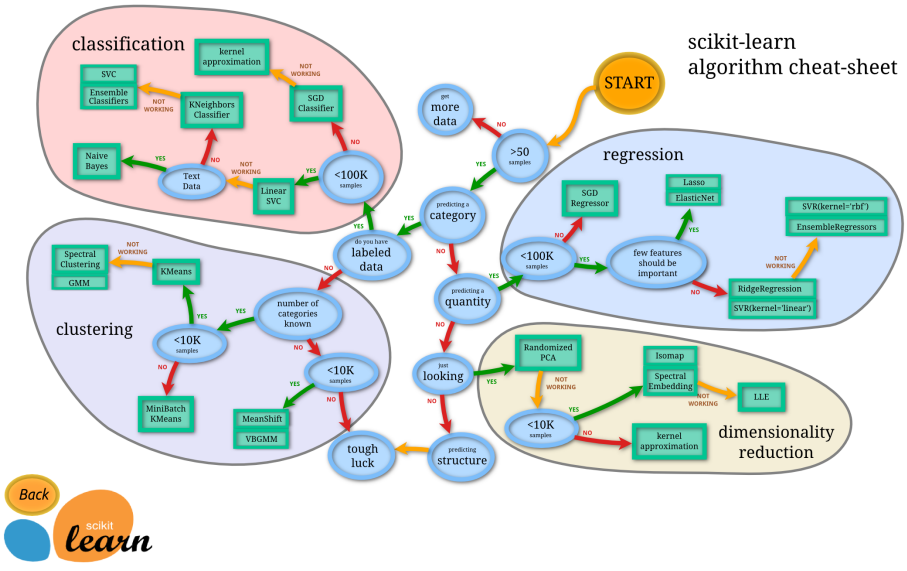

Figure A.1: *List of algorithms in scikit-learn package. Adapted from(Adapted from scikit-learn documentation)*

Scikit-learn has several packages like Classification,Regression,Clustering,Dimensionality Reduction,Feature Extraction etc. Scikit-learn provides support for various machine learning algorithms like Support Vector Machines,Naive Bayes Classifier, K-Nearest Neighbours, Principal Component Analysis etc.