Machine Learning Engineer Nano Degree – Capstone Project

Household Poverty Level Prediction using Machine Learning Classification Algorithms

Domain Background

From Wikipedia: "**Poverty** is the scarcity or the lack of a certain (variant) amount of material possessions or money. Poverty is a multifaceted concept, which may include social, economic, and political elements. Absolute poverty, extreme poverty, or *destitution* refers to the complete lack of the means necessary to meet basic personal needs such as food, clothing and shelter".

Poverty has been, and still is, a major concern for many countries around the world, affecting the lives of billions of people. According to www.dosomething.org, "Nearly 1/2 of the world's population — more than 3 billion people — live on less than $2.50 a day. More than 1.3 billion live in extreme poverty — less than $1.25 a day". And even the World Bank (www.worldbank.org) admits that "Despite the progress made in reducing poverty, the number of people living in extreme poverty globally remains unacceptably high. And given global growth forecasts, poverty reduction may not be fast enough to reach the target of ending extreme poverty by 2030".

Thus, it is still crucial for many countries to find & aid those families in need. One way for social welfare to find the population in need is by the use of the **PMT** (**P**roxy **M**eans **T**est) https://olc.worldbank.org/sites/default/files/1.pdf:

The main idea behind PMT is that in many situations we cannot have clear and accurate income reports on these populations. Even the household members themselves cannot report on their income and\or spending. And so, we have to use other, alternative metrics, in order to evaluate their economy status:

1. The type of wall that is surrounding their house (brick or clay)
2. Does the family have livestock ownership (e.g. cattle)
3. Etc…

Naturally, there can be relatively a very large number of parameters that can make up a PMT for making the evaluation of a family's welfare, and doing such an evaluation manually can be a very long and cumbersome work for social welfare organizations.

This is where machine learning can come into play.

Problem Statement

The challenge of this project is to train and find the best classifier for prediction of poverty level of households according to the given PMT input data. Note that because this is a **supervised multi-class classification** problem (see "Datasets and Inputs" below), we can take into account several algorithms for solving it.

In this project I will evaluate several classifiers that fit this problem, testing each one with various set of hyper parameters, and compare the results between them, denoting the best classifier found.

Datasets and Inputs

The dataset to be used is taken from Kaggle recent competition on this subject, contributed by the Inter-American Development Bank. Link to the competition in Kaggle:
https://www.kaggle.com/c/costa-rican-household-poverty-prediction

The specific data set was given for Costa Rica, though the trained models can probably be re-used for other Latin American countries, and probably for other countries in the world.

The data itself consists of a CSV file containing ~ 10,000 labeled records. Each record (line) is composed of 142 (!) features and the target variable which is a whole number in the range 1-4 (1 = Extreme poverty, 2 = Moderate poverty, 3 = vulnerable household, 4 = non vulnerable household).

Solution Statement

The solution will include evaluating several classification algorithms, to name most of them:

Gaussian Naïve Bayes
Decision Trees
AdaBoost
K-Nearest Neighbors
SVC
Random Forrest

In addition, because the data set contains relatively very large number of features, ***extensive feature analysis & preprocessing will be done***, as well as also using PCA to perform complexity (dimensionality) reduction of the data.

Benchmark Model

I will use a ***naïve predictor*** as benchmark model, just to proof that classification model is possible and compare the algorithms in my solution statement to the naïve predictor.

The naïve predictor will simply make a prediction of value ***4 (Non-vulnerable)*** regardless of the feature vector of a given record.

Evaluation Metrics

I will use the ***weighted* F1 score** and **accuracy** as evaluation metric for the classifiers being tested, as the input data is greatly imbalanced (there are very few extreme poverty records compared to the other record labels), so using the regular F1 score here is not good enough. I will also denote the confusion matrix, as well as the running time for each classifier.

Project Design

The workflow of my project will be composed of the following parts:

Input data analysis & pre-processing

- Checking label distribution to verify if data is balanced or imbalanced
- Fixing errors in the data (records without target, records of same household with different poverty target value)
- Handle missing values (for example: in the v18q1 → "number of tablets" column)
- Remove redundant columns (e.g. "male" and "female" columns, "SQBage" and "agesq", all "SQB_xxx" columns)
- Perform one hot encoding \ feature scaling where\if relevant:
    - Log transformation on skewed data
    - Normalization on numeric data
    - On hot encoding on categorical data
- *Visualizations will be supplied as part of the above analysis and pre-processing operations taken*

Creating random train and test data

Perform shuffle and split of the data into train and testing sets.
Also create using PCA reduced data with a limited number of features (also split to train and test data).

Evaluation of the naïve predictor

Perform evaluation of the naïve predictor on the data (both full and reduced) in order to get an initial benchmark result for comparison with the other models.

Evaluation of the selected models

Perform evaluation on each model with both the original and reduced data.
Use hyper parameter tuning using ***Grid Search*** in order to find the best model out of the list of given classifiers.

The following pseudo code can be used (per each classifier):

```
def invoke_classifier(classifier, hyperparameters, X_train, y_train,
X_test, y_test)
      results = {}
      scorer = make_scorer(f1_score, average = 'weighted')
      grid_cv = GridSearchCV(estimator=classifier,
                  param_grid= hyperparameters, scoring=scorer)
      start = time()
      cls = grid_cv.fit(X_train, y_train)
      end = time()

      results['train_time'] = end - start
```

```
preds_test = cls._best_estimator_.predict(X_test)

results['accuracy'] = accuracy_score(y_test, preds_test)
results['f1_score'] = f1_score(y_test, preds_test,
                               average = 'weighted')
```

Summary & discussion of the results

Discuss all the results from the previous steps, main conclusions and further points that could be investigated.