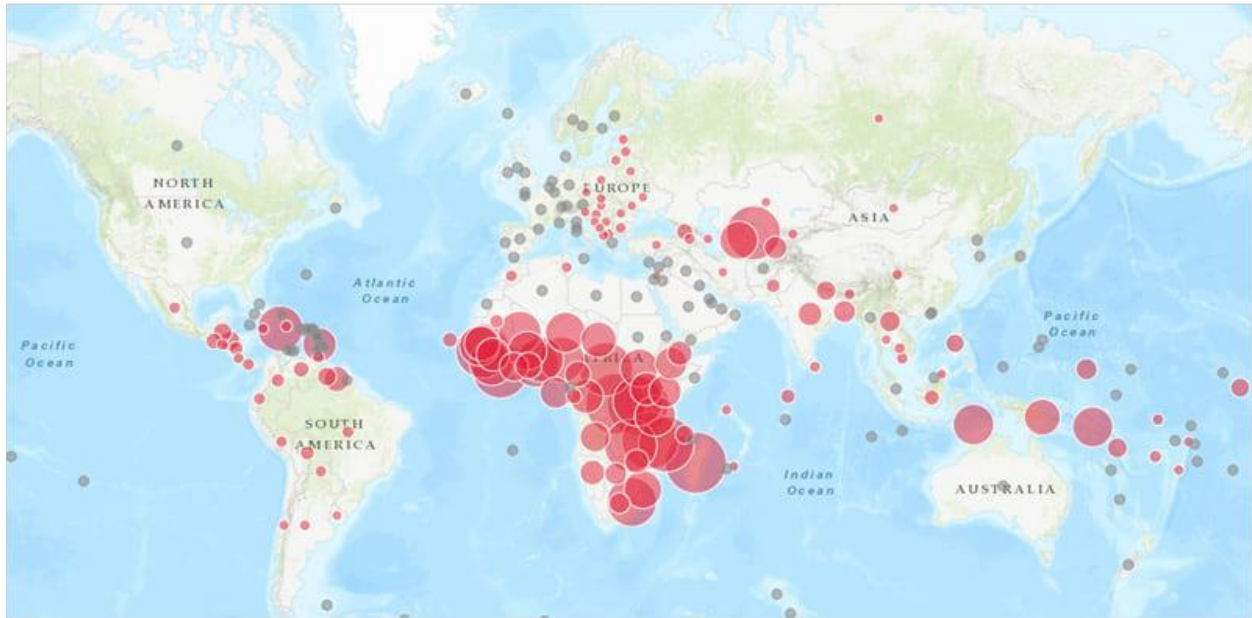


Global Poverty Analysis Report

By Rajib Kumar De



As poverty is a distinctly complex and multivariate issue, determining the true underlying causes contributing to it is paramount to addressing it effectively.

Executive Summary:

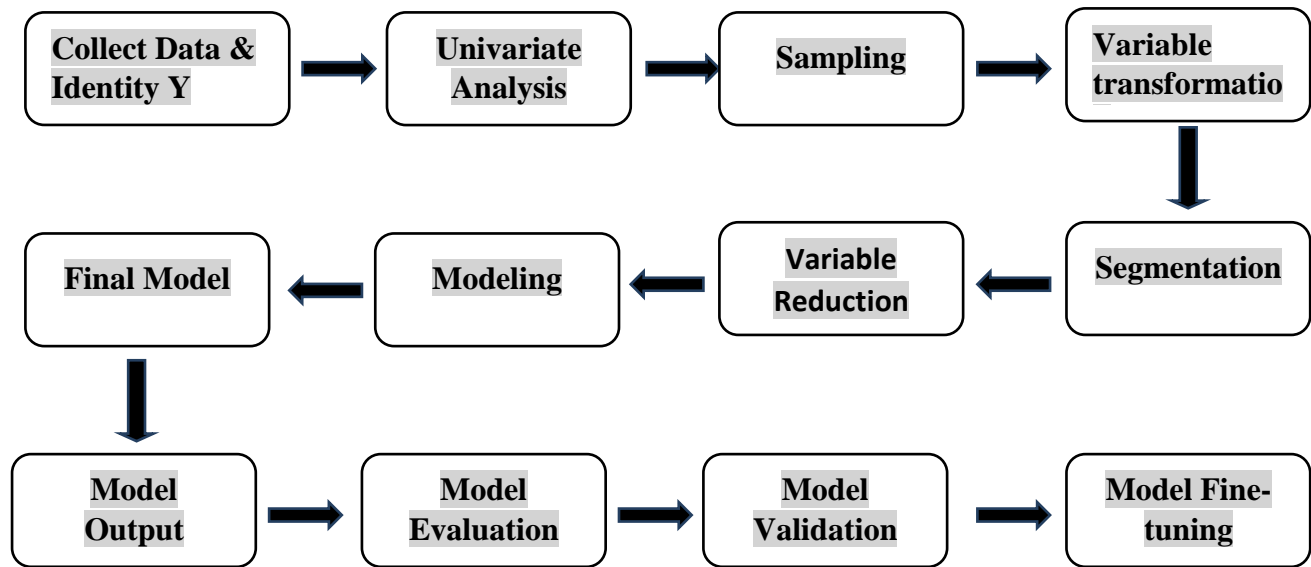
This document presents an analysis of data concerning socioeconomic indicators of individuals and the probability that those individuals live below the poverty line. The analysis is based on data for 12,600 individuals. The probability of being in poverty was calculated by the Poverty Probability Index, which estimates an individual's poverty status using 10 questions about a household's characteristics and asset ownership. The remaining data comes from the Financial Inclusion Insights household surveys conducted by Inter Media.

After exploring the data by calculating summary and descriptive statistics, and by creating visualizations of the data, several potential relationships between socioeconomic indicators and poverty probability were identified. After exploring the data, a regression model was created to predict an individual's poverty probability.

Methodology:

Cleansing, transformation, and analysis of data was performed using Python 3 as needed. Development of machine-learning model was performed in Python 3 (Jupyter Notebook).

Process Flow:



I generally followed above roadmap for Machine Learning problems (Regression and Classification). For our problem, I followed the steps which required. Follow the bellow analysis:

Import general useful packages

```
import pandas as pd
```

Import all machine learning algorithms

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
import xgboost as xgb
```

After importing packages, loaded train and test data and checked shape of the both data set and also checked the missing. YES there were some variables who have missing values and then removed those variables who have > 70% missing for both dataset.

```
#keeping columns where missing values are less than 70% (training data)
indexes=[]
thresh=len(training_data) * 0.70 #threshold for minimum of null/missing values
for i in training_data:
    if (training_data[i].isna().sum() > thresh):
        indexes.append(i)
```

```
#keeping columns where missing values are less than 70% (testing data)
indexes=[]
thresh=len(testing_data) * 0.70 #threshold for minimum of null/missing values
for i in testing_data:
    if (testing_data[i].isna().sum() > thresh):
        indexes.append(i)
```

Then again checked missing if there were any missing in the dataset, Yes TWO variables have missing observation: ['education_level', 'share_hh_income_provided']

Replace missing by MEDIAN as those are categorical observation.

```
#handling missing values where there are less than 1%
# fill missing values for the select columns under study with median value
cols = ['education_level', 'share_hh_income_provided']
training_data.fillna(training_data[cols].median(), inplace=True)
testing_data.fillna(testing_data[cols].median(), inplace=True)
```

Converted all **Boolean** features to **Integer** and separated **Independent** and **dependent** feature for both cases.

```
# Get X and y for training data
y_train = training_data['poverty_probability']
X_train = training_data.drop(columns = ['poverty_probability'])

# Get X and y for testing data
y_test = testing_data['poverty_probability']
X_test = testing_data.drop(columns = ['poverty_probability'])
```

ML Algorithms:

Here I have applied **THREE** ML algorithms:

- **Random Forest**
- **Gradient Boosting**
- **XGBoosting**

Using Random Forest:

```
# Applying various Classification algorithms without doing variable reductions

# Random Forest
clf = RandomForestRegressor().fit(X_train, y_train)
prediction = clf.predict(X_test)
predictions_df = pd.DataFrame(prediction)
predictions_df.rename(columns={0:'poverty_probability'}, inplace=True)

result1 = pd.concat([testing_data['row_id'], predictions_df], axis=1)
print(result1)
result1.to_csv('C:\\Users\\datacore\\OneDrive\\Desktop\\Capstone Project\\result1.csv', index=False)
```

Finally, the selected model was modified to be trained on the *entire* training set and then output predicted values of ‘poverty_probability’ for the test dataset. The model performed suitably in deployment as well, producing a coefficient of determination of **0.3267**.

Using XGBoosting:

```
xg_reg = xgb.XGBRegressor()
xg_reg.fit(X_train,y_train)

preds = xg_reg.predict(X_test)
#print("R-squared for Train: %.2f" %xg_reg.score(X_train, y_train) )
predictions_df = pd.DataFrame(preds)
predictions_df.rename(columns={0:'poverty_probability'}, inplace=True)

result1 = pd.concat([testing_data['row_id'], predictions_df], axis=1)
print(result1)
result1.to_csv('C:\\Users\\datacore\\OneDrive\\Desktop\\Capstone Project\\result31.csv', index=False)
```

Finally, the selected model was modified to be trained on the *entire* training set and then output predicted values of ‘poverty_probability’ for the test dataset. The model performed suitably in deployment as well, producing a coefficient of determination of **0.3617**.

Using Gradient Boosting:

```
#Let's go instantiate, fit and predict.
gbrt=GradientBoostingRegressor(n_estimators=100)
gbrt.fit(X_train, y_train)
y_pred=gbrt.predict(X_test)
print("R-squared for Train: %.2f" %gbrt.score(X_train, y_train) )
predictions_df = pd.DataFrame(y_pred)
predictions_df.rename(columns={0:'poverty_probability'}, inplace=True)

result1 = pd.concat([testing_data['row_id'], predictions_df], axis=1)
print(result1)
result1.to_csv('C:\\Users\\datacore\\OneDrive\\Desktop\\Capstone Project\\result21.csv', index=False)
```

The selected model was modified to be trained on the *entire* training set and then output predicted values of ‘poverty_probability’ for the test dataset. The model performed suitably in deployment as well, producing a coefficient of determination of **0.3632**.

Tune Hyperparameter:

Boosting is a sequential technique which works on the principle of ensemble. It combines a set of weak learners and delivers improved prediction accuracy. At any instant t, the model outcomes are weighed based on the outcomes of previous instant t-1. The outcomes predicted correctly are given a lower weight and the ones miss-classified are weighted higher. This technique is followed for a classification problem

```
#parameter hypertuning
GBR=GradientBoostingRegressor()
search_grid={'n_estimators':[30,50,100,200,300],'learning_rate':[0.01, 0.025, 0.05, 0.075, 0.1, 0.15, 0.2],
            'max_depth':[1,2,4,7,10],'subsample':[.5,.75,1],'random_state':[1,3,5,10]}
search=GridSearchCV(estimator=GBR,param_grid=search_grid,scoring='neg_mean_squared_error',n_jobs=3)
















search.fit(X_train, y_train)
print(search.best_params_)
#https://shankarmsy.github.io/stories/gbrt-sklearn.html

#Let's go instantiate, fit and predict.
gbrt1=GradientBoostingRegressor(n_estimators=100, learning_rate = 0.1, max_depth = 4, random_state=1, subsample=1)
gbrt1.fit(X_train, y_train)
y_pred=gbrt1.predict(X_test)
print("R-squared for Train: %.2f" %gbrt1.score(X_train, y_train))
predictions_df = pd.DataFrame(y_pred)
predictions_df.rename(columns={0:'poverty_probability'}, inplace=True)

result1 = pd.concat([testing_data['row_id'], predictions_df], axis=1)
print(result1)
result1.to_csv('C:\\Users\\datacore\\OneDrive\\Desktop\\Capstone Project\\result4.csv', index=False)
```

Finally, the selected model was modified to be trained on the *entire* training set and then output predicted values of ‘poverty_probability’ for the test dataset. The model performed suitably in deployment as well, producing a coefficient of determination of **0.4247**.

Till now I am at No. 7

	User or team		Best public score ⓘ	Timestamp ⓘ	Trend (last 10)	# Entries
	janvanegmond	1	0.43	2019-07-06 20:01:32		3
	arrabi	2	0.43	2019-07-22 23:37:49		8
	VamsiNellutla	3	0.43	2019-07-02 22:54:29		5
	Harold.Villacorte	4	0.43	2019-07-13 05:40:43		17
	Ranisz	5	0.43	2019-07-24 08:20:06		9
	John0159	6	0.43	2019-07-21 07:39:40		14
	rajibstats	7	0.42	2019-07-24 09:36:07		2
	regret	8	0.42	2019-07-23 16:09:49		3
	anish_talukdar	9	0.42	2019-07-22 12:37:46		19

Conclusion

The analysis of the InterMedia survey data and PPI poverty probabilities yielded many interesting findings, and in the end the development of a predictive model determined that just over 40% of the variance in 'poverty_probability' can be determined by characteristic variations within the feature set. While a further investigation into the underlying accuracy of the PPI values themselves may also be worthwhile, the BDT regression model developed in this analysis process is now ready for web-service deployment and could potentially be of use to researchers and activists combatting poverty in these regions.