AI-Powered Socioeconomic Prediction of Lifespan

Rustom Ichhaporia [rustomi2@illinois.edu][*]

2020-12-12

Abstract

How long will you live? This age-old question has extensive implications in the billions of risk estimations made by individuals planning for the future every day. Although never certain, a stronger approximation of an indidual's lifespan can enable more reliable future planning and a greater sense of stability than none at all. We reviewed publicly available datasets containing socioeconomic information about U.S. citizens to create a naïve model that predicts the likelihood of a person's death at different ages given characteristics such as location, income, place of birth, and more. The results are explained and visualized in this report. While more work must be done to achieve a more accurate predictor, this work provides a baseline for lifespan prediction in coordination with other financial models to aid financial planning.

1 Background

1.1 Desired Features

Lifespan prediction is important to a broad variety of scientific and industrial fields. Each field in which it is relevant requires a different scope, accuracy, and set of input features. For example, the medical field might look at a patient's weight and blood pressure to determine their risk of passing away from heart disease. Millions of specific, dynamic variables affect an individual's lifespan, ranging from minute physical details to sociological environments to occupational conditions. Naturally, it is currently impossible to accurately measure all of these variables for an individual, let alone every individual. Thus, we must let the domain of our application dictate which variables we use, as well as the size and diversity of the dataset we use to predict lifespan. If we were to make a perfect predictor of mortality, some of the high-level features we might consider include:

- General biographical information (e.g. age, sex)
- Location
- Occupational and residential environment conditions
- Income
- Medical information

In particular, the most likely available features would fall into the groups of socioeconomic and medical. Unfortunately, in our dataset search, finding a dataset that combined socioeconomic status, medical information, and lifespan information was difficult to come by. There are datasets linking two of those three features, but a comprehensive, large scale study documenting all three with useful sample sizes was not found. As a result, some compromises had to be made in favoring the socioeconomic data over medical data, as that is more relevant and available in actuarial settings.

1.2 Dataset Selection

1.2.1 CDC Dataset

The first dataset that we attempted to use was the Mortality Multiple Cause-of-Death dataset created by the U.S. Center for Disease Control (CDC)[1]. While this dataset contained several of the features that we wanted to include in our analysis, it was still missing a lot of the socioeconomic factors that we were looking for and the medical information it contained was difficult to parse. After a few weeks of attempting to work with this data, we decided to search for a new dataset that better matched the needs of the research.

1.2.2 NLMS Dataset

The best dataset that we found within our timeframe was from the National Longitudinal Mortality Study

[*]This research project was completed during my time as a research intern at the Illinois Risk Lab (https://irisklabuiuc.wixsite.com/) during the Fall of 2020. My research was a part of the AI-Powered Lifecycle Financial Planning project, which is still under development. I appreciate the help of Dr. Runhuan Feng, Dr. Frank Quan, Dr. Yong Xie, Dr. Linfeng Zhang, and my fellow interns throughout the process. Thank you!

[1]https://www.cdc.gov/nchs/nvss/mortality_public_use_data.htm

(NLMS) created by the United States Census Bureau[2]. The dataset is not available for direct online download, so we requested the data from the Bureau through a short application that was approved. The NLMS dataset contained over 40 features including many of the socioeconomic features that we were looking for. Unfortunately, the medical data provided by the dataset was mostly limited to medical causes of death, which are of dubious explanatory integrity when attempting to predict lifespan. A sample of the features used in the analysis is detailed below:

- Age
- Number of household members
- Inflation-adjusted income
- Income relative to poverty line
- State of residence and urban/rural classification
- Race
- Sex
- Occupation categories

Each entry had a corresponding weight. The purpose of the weight was to attempt to account for oversampling or undersampling of specific demographic populations in the United States, as the NLMS study was a conglomeration of smaller field surveys. The dataset was available in several forms, including a separate dataset solely for the purpose of tracking deaths of people who smoked.

2 Preprocessing

After selecting the dataset, we began the steps of preprocessing the data for model creation. A number of steps that were taken have been omitted from this summary, some of which were kept and some of which were discarded through the research process.

Roughly half of the variables in the NLMS dataset had very high missing rates. Although tactical imputation can be used for values with medium levels of missing data, many of these features were missing more than half of their entries. Simply removing all entries with missing values would shrink the dataset tenfold and very likely skew the data, so I instead opted to drop features with high missing rates. The cutoff point identified was that features with `>20%` of their data missing were removed. The remaining missing values were much easier to handle and discard.

The features related to smoking were removed because they were almost entirely absent from this dataset and were meant for use in a separate, compatible dataset. A more robust iteration of this research may desire to make use of these smoking datasets, but they were not included in this analysis.

The categorical features in the dataset were converted to binary dummy variables in which a number of columns equal to the number of unique categories in feature is created with a `1` in the column for every entry that contained the corresponding category for that feature and a `0` in all other entries.

The response variable, mortality within the 10-year follow-up periods of the study in the dataset, was present in two separate features, `indalg` and `inddea`. These represent two different ways that the NLMS dataset identified deaths. At the recommendation of the reference manual attached to the data, we combined the two features with an inner merge into one named `indmort` and only counted deaths that were present in both categories.

The appendix of this report contains the code for training the model and saving the results in a file. It does not include the code for statistical plots.

3 Modeling

The preprocessed data was then used to fit multiple models with the object predicting mortality likelihood at different ages of individuals on a macroscopic scale. The two models used were `LogisticRegression` from the `scikit-learn` package[3] and `LGBMClassifier` from Microsoft's LightGBM package[4].

3.1 Computation

3.2 Models

3.3 Metrics

3.4 Hyperparameter Optimization

4 Results

Last week, I registered for HAL access so that I could run the hyperparameter optimization script remotely because my computer overheated and could not run it for the appropriate number of trials. I was able to upload my files and run the first half of the script, but unfortunately when running the `fmin` optimization function, the program crashes after the first of 150 loops with the error:

[2]https://www.census.gov/topics/research/nlms.html

[3]https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Logis

[4]https://github.com/microsoft/LightGBM

4.1 Limitations

Several significant drawbacks of our approach to estimating lifespan using this dataset and model are recognized.

One significant drawback of the approach taken to estimating lifespan in this dataset was the

5 Appendix

script.py

```python
'''Imports'''

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LogisticRegressionCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import plot_precision_recall_curve
from sklearn.metrics import plot_roc_curve

from imblearn.over_sampling import SMOTE

sns.set()

'''Preprocessing'''

df_raw = pd.read_csv('data/11.csv')
print('Data successfully loaded.')

df_raw = df_raw.drop(columns=['smok100', 'agesmk', 'smokstat', 'smokhome', 'curruse', 'everuse'])

df_raw['indmort'] = df_raw['inddea'][(df_raw['inddea'] == 1) & (df_raw['indalg'] == 1)]
df_raw['indmort'] = df_raw['indmort'].fillna(0)

used_numerical = ['age', 'hhnum']
used_ordinal = ['povpct', 'adjinc']
used_categorical = ['stater', 'pob', 'sex', 'race', 'urban', 'smsast']
used_special = ['wt', 'indmort']

used_features = used_numerical + used_ordinal + used_categorical + used_special

df_raw = df_raw[used_features]

df_raw[used_categorical] = df_raw[used_categorical].astype('category')

df_raw = df_raw.dropna(axis=0)
```

```python

df = pd.get_dummies(df_raw)

X = df.drop(columns=['indmort'])
y = df['indmort']

'''Sampling'''

X_train, X_test, y_train, y_test = train_test_split(X, y)

print('Proportion of data from minority class before SMOTE:', y_train.sum() / y_train.shape[0])
X_train, y_train = SMOTE().fit_resample(X_train, y_train)
print('Proportion of data from minority class after SMOTE:', y_train.sum() / y_train.shape[0])

'''Modeling'''

model = LogisticRegressionCV(scoring='roc_auc', random_state=0, n_jobs=-1, verbose=1).fit(X_train.drop(columns=['wt']), y_train, sample_weight=X_train['wt'])

print(classification_report(model.predict(X_test.drop(columns=['wt'])), y_test))

pred_probs = model.predict_proba(X_test.drop(columns=['wt']))[:, 1]

print(classification_report(np.round(pred_probs + 0.25), y_test, sample_weight=X_test['wt']))
```