AUTHOR
Jamie Hackney

PUBLISHED
March 1, 2024

Abstract:

This analysis aims to reproduce and discuss some of the results obtained by by Obermeyer et al. (2019) in their paper discussing the racial bias present in an algorithm used to refer patients for care. Through the reproduction of figures, it is shown that on average, Black patients have a lower total medical cost for a given unmber of chronic illnesses than White patients. Since the algorithm uses cost as a proxy for sickness to assign each patient a risk score, this means that Black patients are generally considered less sick than White patients for a given risk score. Contextually, this means that when making referrals to care programs based on the risk score, Black patients actually have to be more sick than White patients to get referred.

Part A: Get the Data

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score

url = "https://gitlab.com/labsysmed/dissecting-bias/-/raw/master/data/data_new.csv?inlin
df = pd.read_csv(url)
df.head()
```

	risk_score_t	program_enrolled_t	cost_t	cost_avoidable_t	bps_mean_t	ghba1c_mean_t	hct_mean_t	cre_mean_t	l
0	1.987430	0	1200.0	0.0	NaN	5.4	NaN	1.110000	1
1	7.677934	0	2600.0	0.0	119.0	5.5	40.4	0.860000	9
2	0.407678	0	500.0	0.0	NaN	NaN	NaN	NaN	N
3	0.798369	0	1300.0	0.0	117.0	NaN	NaN	NaN	N
4	17.513165	0	1100.0	0.0	116.0	NaN	34.1	1.303333	5

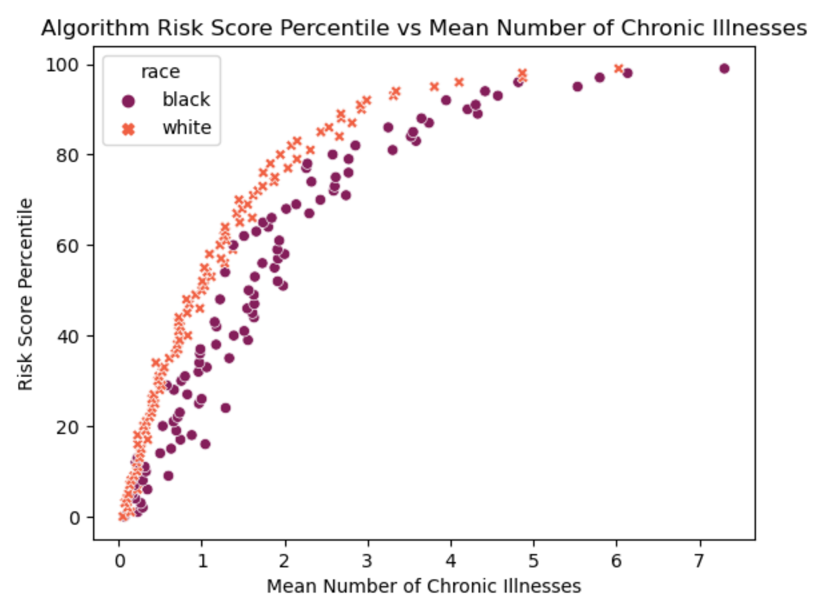
5 rows × 160 columns

Part B: Reproducing Fig. 1

```
df['risk_percentile'] = pd.qcut(df['risk_score_t'], q=100, labels=False, duplicates='dro
mean_illness_percentiles = df.groupby(['risk_percentile', 'race'])['gagne_sum_t'].mean()

percentile_plot = sns.scatterplot(data=mean_illness_percentiles, x='gagne_sum_t', y='ris
percentile_plot.set_title('Algorithm Risk Score Percentile vs Mean Number of Chronic Ill
percentile_plot.set_xlabel('Mean Number of Chronic Illnesses')
percentile_plot.set_ylabel('Risk Score Percentile')
```

```
Text(0, 0.5, 'Risk Score Percentile')
```



This figure shows that there is a disparity between Black patients and White patients risk scores as as function of their number of chronic illnesses. Specifically, Black patients on average have a lower risk score than white patients for the same number of chronic illnesses. In this context, the risk score translates to referrals to a care management program. Therefore, on average, Black patients have to be sicker than White patients to be referred into this program.

Part C: Reproducing Fig. 3

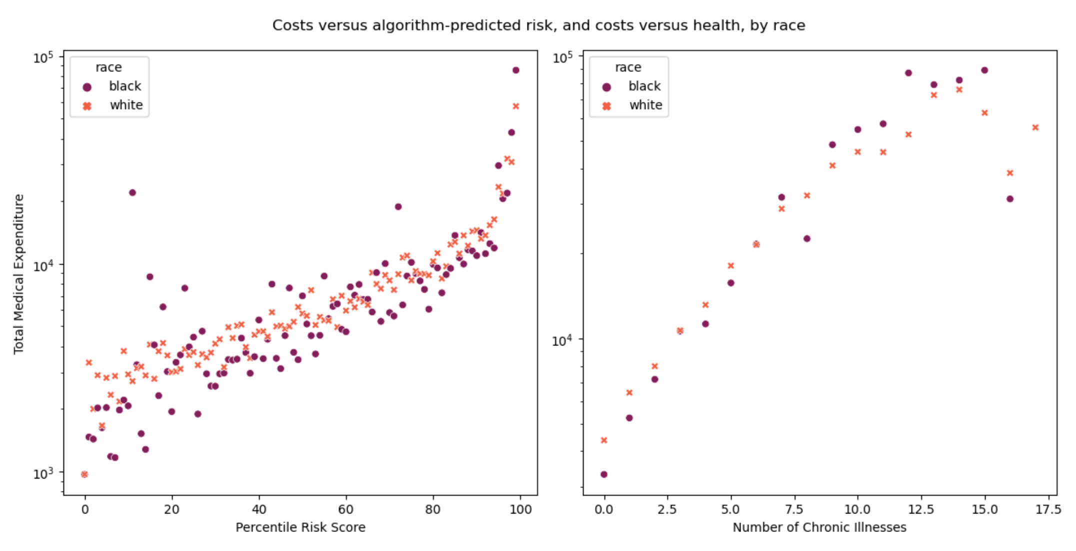
```
df['risk_percentile'] = pd.qcut(df['risk_score_t'], q=100, labels=False, duplicates='dro
cost_risk_percentiles = df.groupby(['risk_percentile', 'race'])['cost_t'].mean().reset_i
cost_illness_percentiles = df.groupby(['gagne_sum_t', 'race'])['cost_t'].mean().reset_in

fig, axs = plt.subplots(1, 2, figsize=(12, 6))

sns.scatterplot(data=cost_risk_percentiles, x='risk_percentile', y='cost_t', hue='race',
axs[0].set_ylabel('Total Medical Expenditure')
axs[0].set_xlabel('Percentile Risk Score')
axs[0].set_yscale('log')

sns.scatterplot(data=cost_illness_percentiles, x='gagne_sum_t', y='cost_t', hue='race',
axs[1].set_ylabel('')
axs[1].set_xlabel('Number of Chronic Illnesses')
axs[1].set_yscale('log')

fig.suptitle('Costs versus algorithm-predicted risk, and costs versus health, by race')
plt.tight_layout()
plt.show()
```



This first figure shows that Black patients seem to average a lower total medical expenditure per percentile risk score than White patients. I also noticed that the few outliers in the first graph are all Black patients. In the second figure, I again saw a steady pattern disparity between the cost incurred by Black patients and White patients. Specifically, Black patients tended to incur fewer costs as White patients for the same number of chronic illnesses. This pattern breaks down at high numbers of chronic illnesses, presumably because of the small amount of data avilable at that point increasing variability.

Part D: Modeling Cost Disparity

To model the cost disparity, I first focused on patients in the data with 5 or fewer chronic illnesses. This represents 96% of the data, which justifies focusing on these patients. Then, I calculated the log cost of each patient to use as the target variable. Log cost works better in this context because the cost varies by orders of magnitude. To avoid the undefined $log(0)$, I subsetted the data to be only patients with costs greater than 0. I then one-hot encoded the race column to be 0 for 'White' and 1 for 'Black'. Finally, I separated the data into predictor and target sets, X and y respectively. Based on the graph in the previous part, the relationship between the number of chronic conditions and the cost might be nonlinear. For that reason, I fit a linear regression model with polynomial features in the number of active chronic conditions. To find the best degree, I looped through different degrees and evaluated each with cross-validation to find the best preforming model. To investigate the cost incurred by a Black patient in comparison to an equally sick white patient, I wanted to calulate e^{w_b}, as that would provide an estimate of the percentage of that cost. To find w_b, I extracted the model's coefficients with `LR.coef_` and saw that the coefficient corresponding to the race column was 1.16×10^{-4}. Finally, using this coefficient, I calculated $e^{w_b} = 1.00$. Therefore, the estimate of the percentage of cost incurred by a Black patient in comparison to an equally sick white patient was about 1%.

```
(df['gagne_sum_t'] <= 5 ).sum() / len(df) * 100 # percent of patients with 5 or fewer ch
```

```
95.53952115447689
```

```
model_df = df[df['cost_t'] > 0]
model_df['log_cost'] = np.log(model_df['cost_t'])
model_df['dummy_race'] = model_df['race'].apply(lambda x: 1 if x == 'black' else 0)

X = model_df[['gagne_sum_t', 'dummy_race']]
y = model_df[['log_cost']]
```

```
/var/folders/rt/d3s5hnhn7sb9m39n0ky1cyrc0000gn/T/ipykernel_12380/1798967815.py:2: Setting
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
  model_df['log_cost'] = np.log(model_df['cost_t'])
/var/folders/rt/d3s5hnhn7sb9m39n0ky1cyrc0000gn/T/ipykernel_12380/1798967815.py:3: Setting
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
  model_df['dummy_race'] = model_df['race'].apply(lambda x: 1 if x == 'black' else 0)
```

```
def add_polynomial_features(X, degree):
  X_ = X.copy()
  for j in range(1, degree):
    X_[f"poly_{j}"] = X_["gagne_sum_t"]**j
  return X_

scores = []
degrees = range(1, 20, 1)
for degree in degrees:
  X_ = add_polynomial_features(X, degree)
  LR = LinearRegression()
  LR.fit(X_, y)
  cv_score = cross_val_score(LR, X_, y, cv=5)
  scores.append((degree, cv_score.mean()))

print(max(scores, key=lambda x: x[1]))
```

```
(10, 0.14830062441981076)
```

```
X_poly = add_polynomial_features(X, 10)
LR = LinearRegression()
LR.fit(X_, y)

print(df)
print(LR.coef_) # race coefficient is the 9th one
```

```
       risk_score_t  program_enrolled_t  cost_t  cost_avoidable_t  bps_mean_t  \
0          1.987430                   0  1200.0               0.0         NaN
1          7.677934                   0  2600.0               0.0       119.0
2          0.407678                   0   500.0               0.0         NaN
3          0.798369                   0  1300.0               0.0       117.0
4         17.513165                   0  1100.0               0.0       116.0
...             ...                 ...     ...               ...         ...
48779      0.611517                   0   800.0               0.0         NaN
48780      2.615933                   0  2200.0               0.0       112.0
48781      1.358926                   0   800.0               0.0       105.0
48782     10.990318                   0  1300.0               0.0       132.0
48783      1.681671                   0  4400.0               0.0       115.0

       ghba1c_mean_t  hct_mean_t  cre_mean_t  ldl_mean_t   race  ...  \
0                5.4         NaN    1.110000       194.0  white  ...
1                5.5        40.4    0.860000        93.0  white  ...
2                NaN         NaN         NaN         NaN  white  ...
3                NaN         NaN         NaN         NaN  white  ...
4                NaN        34.1    1.303333        53.0  white  ...
...              ...         ...         ...         ...    ...  ...
48779            NaN         NaN    1.090000       148.0  white  ...
48780            NaN        41.4    0.810000       172.0  white  ...
48781            NaN         NaN         NaN         NaN  white  ...
48782            NaN         NaN         NaN         NaN  white  ...
48783            5.6        36.6    0.940000         NaN  white  ...

       trig_min-high_tm1  trig_min-normal_tm1  trig_mean-low_tm1  \
0                      0                    0                  0
1                      0                    1                  0
2                      0                    0                  0
3                      0                    0                  0
4                      0                    0                  0
...                  ...                  ...                ...
48779                  0                    0                  0
48780                  0                    1                  0
48781                  0                    1                  0
48782                  0                    0                  0
48783                  0                    0                  0

       trig_mean-high_tm1  trig_mean-normal_tm1  trig_max-low_tm1  \
0                       0                     0                 0
1                       0                     1                 0
2                       0                     0                 0
3                       0                     0                 0
4                       0                     0                 0
...                   ...                   ...               ...
48779                   0                     0                 0
48780                   0                     1                 0
48781                   0                     1                 0
48782                   0                     0                 0
48783                   0                     0                 0

       trig_max-high_tm1  trig_max-normal_tm1  gagne_sum_tm1  gagne_sum_t
0                      0                    0              0            0
1                      0                    1              4            3
2                      0                    0              0            0
3                      0                    0              0            0
4                      0                    0              1            1
...                  ...                  ...            ...          ...
48779                  0                    0              0            0
48780                  0                    1              1            1
48781                  0                    1              1            0
48782                  0                    0              3            3
48783                  0                    0              0            0

[48784 rows x 160 columns]
[[ 1.96813673e-07 -7.86659404e-06  1.62366684e-06  2.02601055e-06
   3.14621145e-06  9.78190404e-06  2.94715908e-05  7.44157570e-05
   1.37104478e-04  1.16828277e-04 -1.10271771e-04  3.29252926e-05
  -5.08045181e-06  4.53468530e-07 -2.36781061e-08  6.73012824e-10
  -8.05479300e-12  8.02281625e-18 -3.41459774e-20  7.50814018e-19]]
```

```
wb = LR.coef_[0][9]
```

```
0.00011682827681117512
```

```
np.exp(wb)
```

```
1.0001371138767738
```

Discussion:

From this analysis, I learned that the presence of bias in an algorithm can be heavily dependent on the target variable, and it is therefore important to choose a target variable that accurately reflects what the model is trying to predict. For example, the target variable of cost was chosen to be a proxy for sickness, with the idea being that if a patient incurred more costs, they were sicker. Therefore, predicting the costs incurred by a patient could predict how sick they would be, allowing the algorithm to refer them to more care earlier. The problem with this, however, is that access to healthcare is not the same for all, so some patients incur fewer costs simply because they cannot access health care, which has nothing to do with how sick they actually are. This was the bias found by by Obermeyer et al. (2019), who showed that "At a given level of health (again measured by number of chronic illnesses), Blacks generate lower costs than Whites—on average, $1801 less per year, holding constant the number of chronic illnesses."

The bias as described by Obermeyer et al. (2019) is generally not best supported by the sufficiency discrimination criteria. They mention this in their paper, saying "conditional on risk score, predictions do not favor Whites or Blacks anywhere in the risk distribution." Instead, the statistical independence definition of discrimination more accurately describes the bias present in this models predictions. For the predictor to be unbiased from a statistical independence perspective, the probabilty of a positive prediction does not depend on group membership. However, with this model we see that for a given number of chronic illnesses, Black patients will generally have a lower risk score, and therefore have a lower probability of being referred for care.