Criminal Justice - Recidivism

DeBoris Leonard

11/9/2020

INTRODUCTION:

```
Recidivism is the $10,000 word for the likely hood of a convicted criminal to repeat offend. In recent y
```

```
In this project we are going to look at data on recidivism from Broward County and ProPublica to see if
```

OBSERVATIONS:

Total Count of Defendents:

```
## [1] 6011
```

Age Range of Defendants:

```
##         25 - 45 Greater than 45    Less than 25
##            3384            1172            1455
```

```
Based on the data most defendants in fall between 25-45 years of age which aligns with the average age o
```

Total Defendants by Race:

```
## African-American            Asian        Caucasian         Hispanic
##             3342               21             1933              403
##  Native American            Other
##                8              304
```

```
print("Black: %.2f%%" %           (3342 / 6011 * 100))
```

```
## Black: 55.60%
```

```
print("White: %.2f%%" %           (1933 / 6011 * 100))
```

```
## White: 32.16%
```

```
print("Hispanic: %.2f%%" %        (403 / 6011 * 100))
```

```
## Hispanic: 6.70%
```

```
print("Other: %.2f%%" %            (304 / 6011 * 100))
```

```
## Other: 5.06%
```

```
print("Asian: %.2f%%" %            (21  / 6011 * 100))
```

```
## Asian: 0.35%
```

```
print("Native American: %.2f%%"%    (8   / 6011 * 100))
```

```
## Native American: 0.13%
```

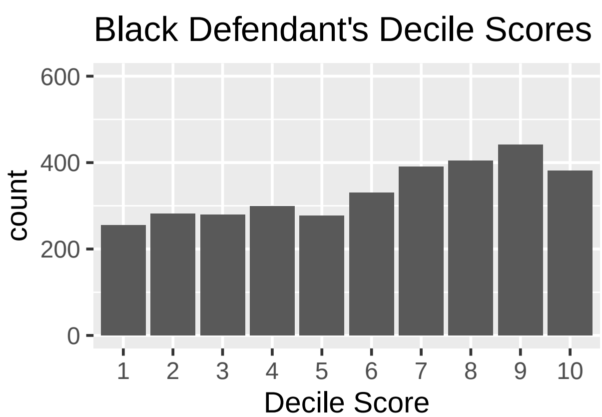
```
The percentage of defendants broken down by race shows that Black or African American individuals compr
```

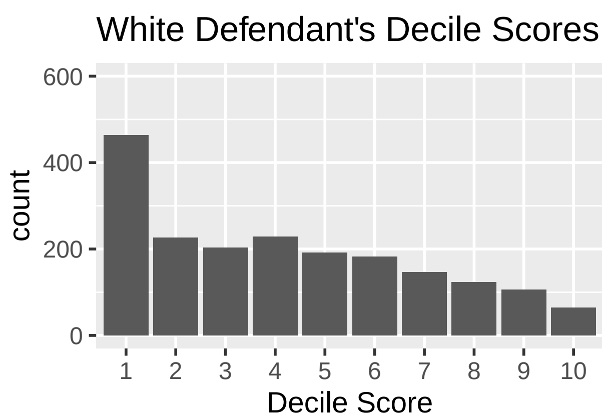
Total Poluation by Gender:

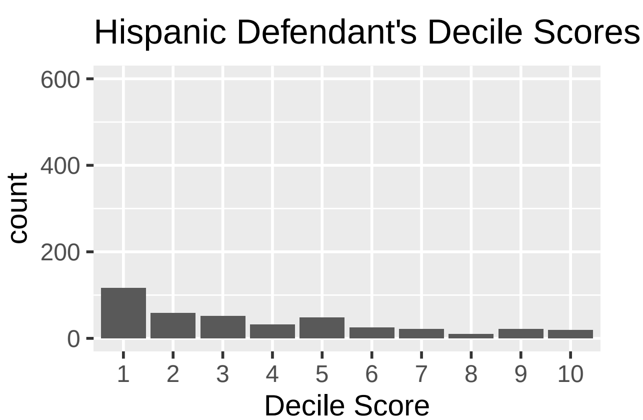
```
## Female   Male
##    936   5075
```

Total Population by Gender and Race:

```
##         race
## sex       African-American Asian Caucasian Hispanic Native American Other
##   Female               416     0       431       57               2    30
##   Male                2926    21      1502      346               6   274
```







```
The Decile Score is the measurement of the likelihood of recidivism. The higher the Decile Score the mo
```

```
But what if there is bias built in the algorithm? Could COMPAS' built in bias create a system that rate
```

Decile Score by Race:

```
##             race
## decile_score African-American Asian Caucasian Hispanic Native American Other
##           1              255     8       464      116               0    92
##           2              282     1       226       59               2    52
##           3              280     3       203       51               1    34
##           4              299     0       228       32               0    33
##           5              277     1       191       48               0    22
##           6              330     2       182       25               0    27
##           7              391     2       146       21               1    17
##           8              405     4       123       10               0    11
##           9              442     0       106       22               2     4
##           10             381     0        64       19               2    12
```

```
##
## Call:
## glm(formula = score_factor ~ gender_factor + age_factor + race_factor +
##     priors_count + crime_factor + two_year_recid, family = "binomial",
##     data = compas_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0611  -0.7597   0.2124   0.7538   2.6312
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 -1.57471    0.08915 -17.663  < 2e-16 ***
## gender_factorFemale          0.18448    0.08885   2.076   0.0379 *
## age_factorGreater than 45   -1.48482    0.10210 -14.543  < 2e-16 ***
## age_factorLess than 25       1.29450    0.07917  16.352  < 2e-16 ***
## race_factorAfrican-American  0.43348    0.07233   5.993 2.06e-09 ***
## race_factorAsian             0.76737    0.49806   1.541   0.1234
## race_factorHispanic         -0.31006    0.13930  -2.226   0.0260 *
## race_factorNative American   0.82142    0.88861   0.924   0.3553
## race_factorOther            -0.75951    0.15646  -4.854 1.21e-06 ***
## priors_count                 0.26984    0.01090  24.754  < 2e-16 ***
## crime_factorM               -0.33000    0.07025  -4.697 2.63e-06 ***
## two_year_recid               0.95981    0.07028  13.657  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8279.1  on 6010  degrees of freedom
## Residual deviance: 5768.3  on 5999  degrees of freedom
## AIC: 5792.3
##
## Number of Fisher Scoring iterations: 5
```

```
## [1] 1.427987
```

Based on the Broward County data Black or African American individuals are almost 43% more likely to rec

```r
exp(0.118109) / (1 - control + (control * exp(0.118109)))
```

```
## [1] 1.098859
```

Women are almost 10% more likely to receive a higher score than their male counterparts.

```python
from sys import stdout
from csv import DictReader, DictWriter

class PeekyReader:
    def __init__(self, reader):
        self.peeked = None
        self.reader = reader

    def peek(self):
        if self.peeked is None:
            self.peeked = next(self.reader)
        return self.peeked

    def __iter__(self):
        return self

    def __next__(self):
        if self.peeked is not None:
            ret = self.peeked
            self.peeked = None
            return ret
        try:
            return next(self.reader)
        except StopIteration:
            self.peeked = None
            raise StopIteration


class Person:
    def __init__(self, reader):
        self.__rows = []
        self.__idx = reader.peek()['id']
        try:
            while reader.peek()['id'] == self.__idx:
                self.__rows.append(next(reader))
        except StopIteration:
            pass

    @property
    def lifetime(self):
        memo = 0
        for it in self.__rows:
```

```
            memo += int(it['end']) - int(it['start'])
        return memo

    @property
    def recidivist(self):
        return (self.__rows[0]['is_recid'] == "1" and
                self.lifetime <= 730)

    @property
    def violent_recidivist(self):
        return (self.__rows[0]['is_violent_recid'] == "1" and
                self.lifetime <= 730)

    @property
    def low(self):
        return self.__rows[0]['score_text'] == "Low"

    @property
    def high(self):
        return not self.low

    @property
    def low_med(self):
        return self.low or self.score == "Medium"

    @property
    def true_high(self):
        return self.score == "High"

    @property
    def vlow(self):
        return self.__rows[0]['v_score_text'] == "Low"

    @property
    def vhigh(self):
        return not self.vlow

    @property
    def vlow_med(self):
        return self.vlow or self.vscore == "Medium"

    @property
    def vtrue_high(self):
        return self.vscore == "High"

    @property
    def score(self):
        return self.__rows[0]['score_text']

    @property
    def vscore(self):
        return self.__rows[0]['v_score_text']
```

```
    @property
    def race(self):
        return self.__rows[0]['race']

    @property
    def valid(self):
        return (self.__rows[0]['is_recid'] != "-1" and
                (self.recidivist and self.lifetime <= 730) or
                self.lifetime > 730)

    @property
    def compas_felony(self):
        return 'F' in self.__rows[0]['c_charge_degree']

    @property
    def score_valid(self):
        return self.score in ["Low", "Medium", "High"]

    @property
    def vscore_valid(self):
        return self.vscore in ["Low", "Medium", "High"]

    @property
    def rows(self):
        return self.__rows


def count(fn, data):
    return len(list(filter(fn, list(data))))


def t(tn, fp, fn, tp):
    surv = tn + fp
    recid = tp + fn
    print("            \tLow\tHigh")
    print("Survived    \t%i\t%i\t%.2f" % (tn, fp, surv / (surv + recid)))
    print("Recidivated\t%i\t%i\t%.2f" % (fn, tp, recid / (surv + recid)))
    print("Total: %.2f" % (surv + recid))
    print("False positive rate: %.2f" % (fp / surv * 100))
    print("False negative rate: %.2f" % (fn / recid * 100))
    spec = tn / (tn + fp)
    sens = tp / (tp + fn)
    ppv = tp / (tp + fp)
    npv = tn / (tn + fn)
    prev = recid / (surv + recid)
    print("Specificity: %.2f" % spec)
    print("Sensitivity: %.2f" % sens)
    print("Prevalence: %.2f" % prev)
    print("PPV: %.2f" % ppv)
    print("NPV: %.2f" % npv)
    print("LR+: %.2f" % (sens / (1 - spec)))
    print("LR-: %.2f" % ((1-sens) / spec))
```

```python
def table(recid, surv, prefix=''):
    tn = count(lambda i: getattr(i, prefix + 'low'), surv)
    fp = count(lambda i: getattr(i, prefix + 'high'), surv)
    fn = count(lambda i: getattr(i, prefix + 'low'), recid)
    tp = count(lambda i: getattr(i, prefix + 'high'), recid)
    t(tn, fp, fn, tp)


def hightable(recid, surv, prefix=''):
    tn = count(lambda i: getattr(i, prefix + 'low_med'), surv)
    fp = count(lambda i: getattr(i, prefix + 'true_high'), surv)
    fn = count(lambda i: getattr(i, prefix + 'low_med'), recid)
    tp = count(lambda i: getattr(i, prefix + 'true_high'), recid)
    t(tn, fp, fn, tp)


def vtable(recid, surv):
    table(recid, surv, prefix='v')


def vhightable(recid, surv):
    hightable(recid, surv, prefix='v')


def is_race(race):
    return lambda x: x.race == race


def write_two_year_file(f, pop, test, headers):
    headers = list(headers)
    headers.append('two_year_recid')
    with open(f, 'w') as o:
        writer = DictWriter(o, fieldnames=headers)
        writer.writeheader()
        for person in pop:
            row = person.rows[0]
            if getattr(person, test):
                row['two_year_recid'] = 1
            else:
                row['two_year_recid'] = 0

            if person.compas_felony:
                row['c_charge_degree'] = 'F'
            else:
                row['c_charge_degree'] = 'M'
            writer.writerow(row)
            stdout.write('.')


def create_two_year_files():
    people = []
    headers = []
    with open("C:/Users/debor/Documents/GitHub/dsc520/Final Project/cox-violent-parsed.csv") as f:
```

```
        reader = PeekyReader(DictReader(f))
        try:
            while True:
                p = Person(reader)
                if p.valid:
                    people.append(p)
        except StopIteration:
            pass
        headers = reader.reader.fieldnames

    pop = list(filter(lambda i: (i.recidivist and i.lifetime <= 730) or
                      i.lifetime > 730,
                      filter(lambda x: x.score_valid, people)))

    vpop = list(filter(lambda i: (i.violent_recidivist and i.lifetime <= 730) or
                       i.lifetime > 730,
                       filter(lambda x: x.vscore_valid, people)))

    write_two_year_file("./compas-scores-two-years.csv", pop,
                        'recidivist', headers)
    write_two_year_file("./compas-scores-two-years-violent.csv", vpop,
                        'violent_recidivist', headers)


if __name__ == "__main__":
    create_two_year_files()
```

```
## ......................................................................................................
```

```
people = []
with open("C:/Users/debor/Documents/GitHub/dsc520/Final Project/cox-violent-parsed.csv") as f:
    reader = PeekyReader(DictReader(f))
    try:
        while True:
            p = Person(reader)
            if p.valid:
                people.append(p)
    except StopIteration:
        pass

pop = list(filter(lambda i: ((i.recidivist == True and i.lifetime <= 730) or
                              i.lifetime > 730), list(filter(lambda x: x.score_valid, people))))
recid = list(filter(lambda i: i.recidivist == True and i.lifetime <= 730, pop))
rset = set(recid)
surv = [i for i in pop if i not in rset]

print("All Defendants:\n")
```

```
## All Defendants:
```

```r
table(list(recid), list(surv))
```

```
##              Low High
## Survived     1822    803 0.37
## Recidivated  1423    3140    0.63
## Total: 7188.00
## False positive rate: 30.59
## False negative rate: 31.19
## Specificity: 0.69
## Sensitivity: 0.69
## Prevalence: 0.63
## PPV: 0.80
## NPV: 0.56
## LR+: 2.25
## LR-: 0.45
```

```
The test reveals that there is an overall false positive rate of 30.6% for all defendants when comparing
```

```r
print("Black Defendants:\n")
```

```
## Black Defendants:
```

```r
is_afam = is_race("African-American")
table(list(filter(is_afam, recid)), list(filter(is_afam, surv)))
```

```
##              Low High
## Survived     684 502 0.30
## Recidivated  638 2138    0.70
## Total: 3962.00
## False positive rate: 42.33
## False negative rate: 22.98
## Specificity: 0.58
## Sensitivity: 0.77
## Prevalence: 0.70
## PPV: 0.81
## NPV: 0.52
## LR+: 1.82
## LR-: 0.40
```

```r
print("\nWhite Defendants:\n")
```

```
##
## White Defendants:
```

```r
is_white = is_race("Caucasian")
table(list(filter(is_white, recid)), list(filter(is_white, surv)))
```

```
##              Low High
## Survived     762 208 0.42
## Recidivated  572 775 0.58
```

```
## Total: 2317.00
## False positive rate: 21.44
## False negative rate: 42.46
## Specificity: 0.79
## Sensitivity: 0.58
## Prevalence: 0.58
## PPV: 0.79
## NPV: 0.57
## LR+: 2.68
## LR-: 0.54
```

```
print("\nHispanic Defendants:\n")
```

```
##
## Hispanic Defendants:
```

```
is_hisp = is_race("Hispanic")
table(list(filter(is_hisp, recid)), list(filter(is_hisp, surv)))
```

```
##              Low High
## Survived     230 60  0.55
## Recidivated  99  134 0.45
## Total: 523.00
## False positive rate: 20.69
## False negative rate: 42.49
## Specificity: 0.79
## Sensitivity: 0.58
## Prevalence: 0.45
## PPV: 0.69
## NPV: 0.70
## LR+: 2.78
## LR-: 0.54
```

```
After reviewing the data for the top 3 population groups it was determined that Black or African Americ

It would be easy to assume that COMPAS is biased based on these numbers. However, taking look at the nu

I cannot conclude based on this that COMPAS is biased against Black people. These numbers do, however,
```