County Based Opportunites for Children

BQOM 2578 | Data Mining

Group 8: Anthony Pulleo, Hannah Shernisky, Theresa Wohlever

Sunday, November 9, 2025

Table of contents

Executive Summary

1. With the intention of predicting the Child Opportunity Index (COI), a resource quality measure for healthy development of children our full dataset is aggregated from the following sources:
    - diversitydatakids.org
    - Census data
    - Urban influence codes, and
    - 2025 Country Health data

2. Using our project data we read in the full dataset, clean, wrangle, then prune. The pruned dataset is refined specifically for Random Forest (RF) and Artificial Neural Network (ANN). For the purpose of this homework, we omitted data exploration topics, covered in the final project. The COI score is continuous, therefore we will use regression models through the course of the homework.
3. For random forest, we identified an *optimal mtry at 16*. Running this model, we see that the average number of unhealthy days influences the dependent variable most across the trees. Further, an *OSR of 93, represents a 27% increase in model "accuracy"* over a regression tree.
4. For the neural network, we generated both a zero and one hidden layer model. OSR calculations for both models returned model performance in the *low 80%*. When compared against the Random forest, this is 10-11% less, therefore RF performed better than ANN.

Group 8: Anthony Pulleo, Hannah Shernisky, Theresa Wohlever

Data Preparation

The Child Opportunity Index (COI) measures and maps the quality of resources and conditions like these that matter for children's healthy development in the neighborhoods where they live.

Importing Data, Cleaning, & Wrangling

The data from diversitydatakids.org contains a series of indices, and does not provide the raw data that was used in the index calculation. The indices are normalized across different areas, like education or housing. We will pull from other datasets to see if factors calculated / assessed by those datasets influence the COI.

These datasets include the Urban Influence Codes from USDA, Census Data as part of the American Community Survey, and the 2025 County Health Rankings Data. Select variables will be appended via a left_join by County FIPS codes.

Modeling

Split data into training and testing

We will deviate slightly from the dataset we will use in our final project for the purpose of this homework. When doing the ANN, we identified a few variables that made the ANN return

NaNs, and therefore inhibited our ability to successfully run these models. We will create df_newDataset2 in the code below with variables that allow the ANN to work.

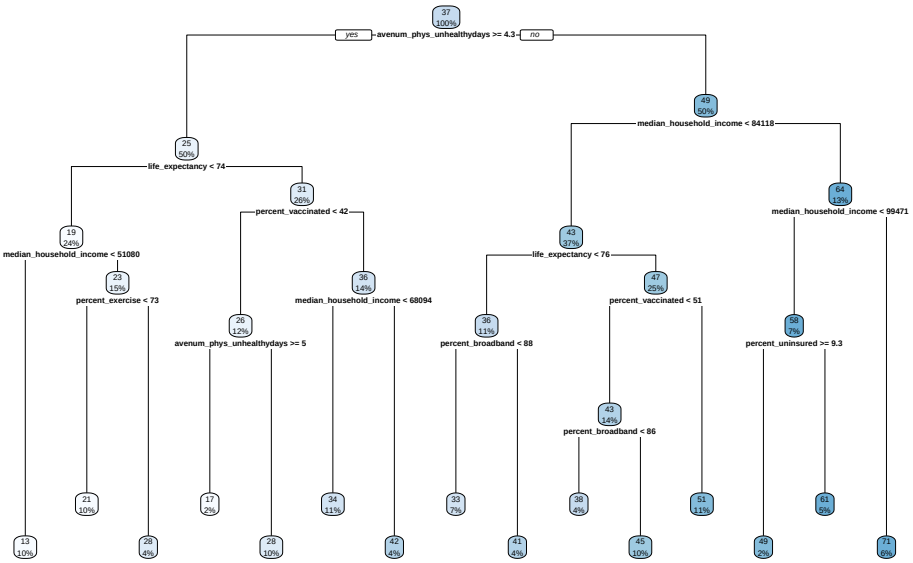
```
df_newDataset2 <- df_newDataset %>% select(county_fips,r_COI_nat,life_expectancy,percent_vaccinated,pop,median_household_income,avenum

set.seed(123, sample.kind = "Rejection")
spl = sample(nrow(df_newDataset2),0.8*nrow(df_newDataset2))
train.RF = df_newDataset2[spl,]
test.RF = df_newDataset2[-spl,]
```

Regression Tree

```
regTree = rpart(r_COI_nat ~ ., data = train.RF, method = "anova",
                minbucket = 300, cp = 0.005)

rpart.plot(regTree, digits=-2)
```



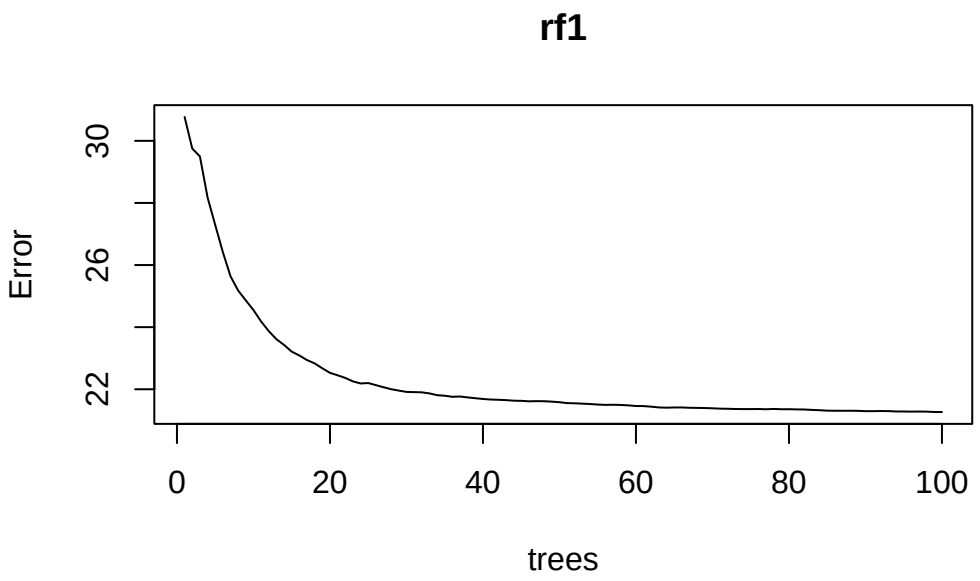
Random Forest

We will start by creating a random forest with similar parameters called for in class, and then we will identify the best mtry.

```
rf1 = randomForest(r_COI_nat~.,data=train.RF,ntree=100,nodesize=50,mtry=7)
```

We see in the plot that after about 20 trees, the model starts to stabilize error around 22, therefore there is not a significant benefit of a model greater than 40, but probably not greater than 20.

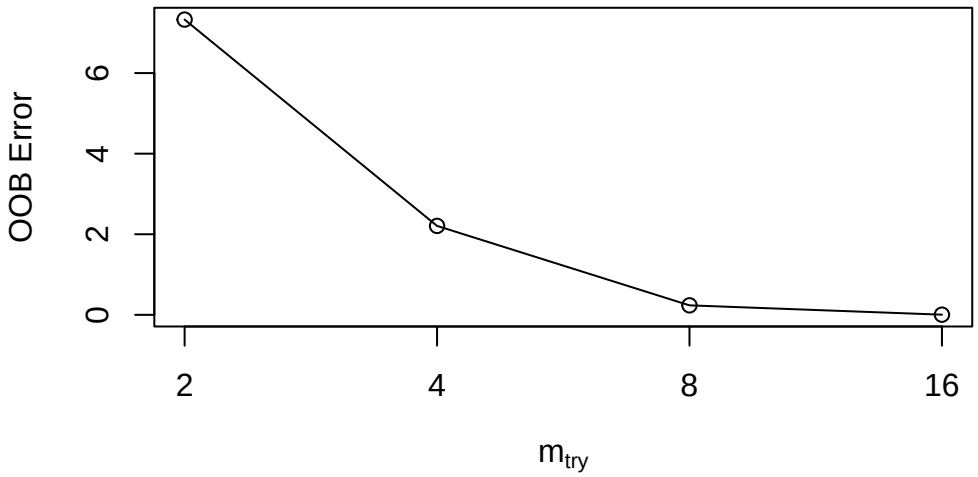
```
plot(rf1)
```




```
x = train.RF[,-4]
y = train.RF$r_COI_nat

tuneRF(x, y, mtryStart = 4, stepFactor = 2, ntreeTry=50, nodesize=50, improve=0.01)
```

```
mtry = 4  OOB error = 2.20686
Searching left ...
mtry = 2    OOB error = 7.327495
-2.320327 0.01
Searching right ...
mtry = 8    OOB error = 0.2359907
0.8930649 0.01
mtry = 16   OOB error = 0.003919777
0.9833901 0.01
```

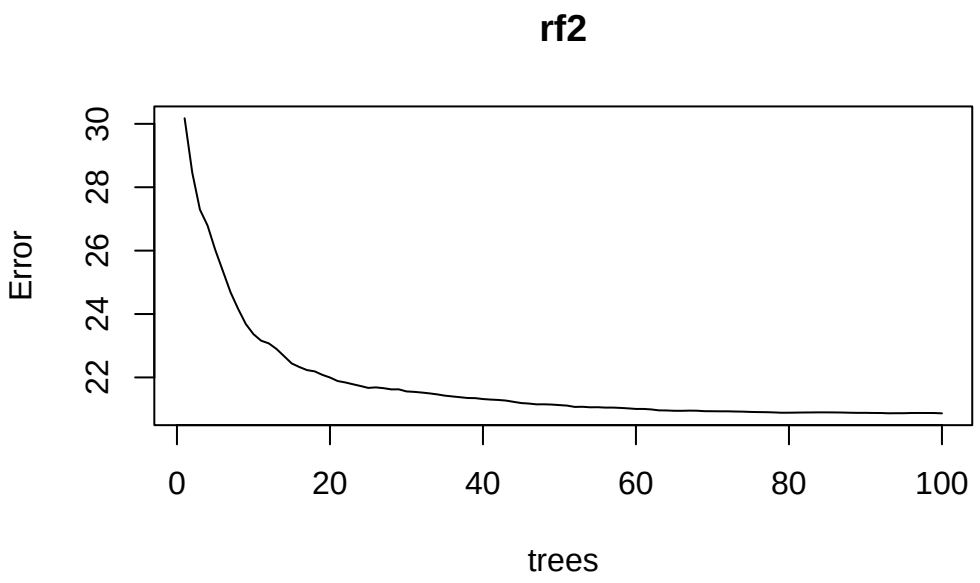



```
   mtry    OOBError
2     2 7.327495033
4     4 2.206859632
8     8 0.235990673
16   16 0.003919777
```

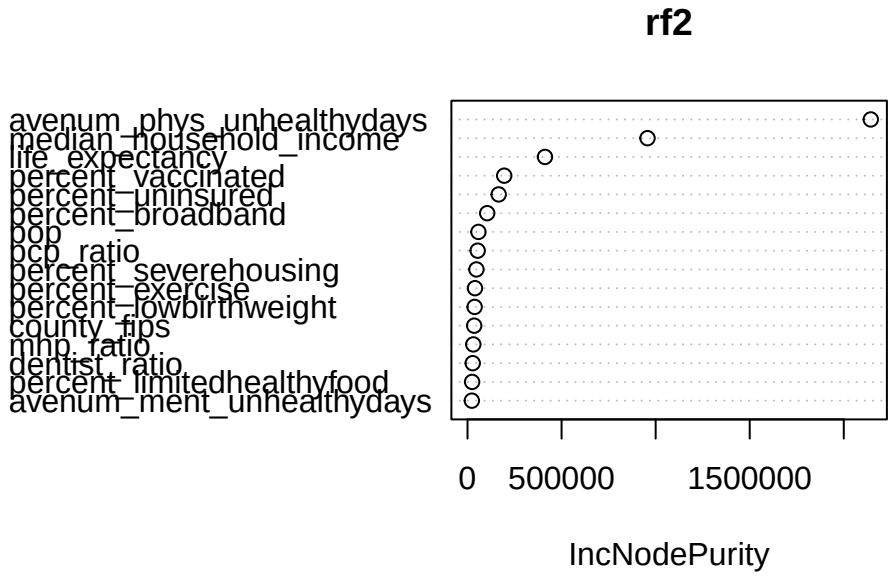
The lowest Out-of-Bag error from the dataset and model indicates that an optimal mtry is 16, because this represents the lowest error in the model. Despite the step of 2, we are fairly confident there are not further dips in the OOB Error curve, given the gentle descent of the curve.

We will then generate a "final" Random Forest, with the optimal mtry:

```
rf2 = randomForest(r_COI_nat~.,data=train.RF,ntree=100,nodesize=50,mtry=16)
plot(rf2)
```




```
varImpPlot(rf2)
```



Average number of unhealthy days has the highest impact on inceasing node purity, followed by median household income, and life expectancy

```
#Predictions for the Random Forest
meanCOI = mean(train.RF$r_COI_nat)
pred_rf = predict(rf2, newdata=test.RF)
SSE_rf = sum((test.RF$r_COI_nat - pred_rf)^2)
SST_rf = sum((test.RF$r_COI_nat - meanCOI)^2)
```

```
OSR_rf = 1 - SSE_rf/SST_rf
OSR_rf
```

```
[1] 0.9343833
```

Comparison to Regression Tree

```
# Predictions from Regression Tree
pred = predict(regTree, newdata=test.RF)
SSE = sum((test.RF$r_COI_nat - pred)^2)
SST = sum((test.RF$r_COI_nat - meanCOI)^2)
OSR2 = 1 - SSE/SST
OSR2
```

```
[1] 0.7350934
```

```
#% Improvement with random forest
OSR_rf/OSR2  -  1
```

```
[1] 0.2711083
```

27% improvement in model performance by using random forest

Artificial Neural Network

```
train.ANN = df_newDataset2[spl,]
test.ANN = df_newDataset2[-spl,]
```

Pre-processing data by scaling it to between 0 and 1:

```
maxVals = apply(train.ANN, 2, max)
minVals = apply(train.ANN, 2, min)
```

```
scaled_train = as.data.frame(scale(train.ANN, center = minVals,
                                   scale = maxVals - minVals))
scaled_test = as.data.frame(scale(test.ANN, center = minVals,
                                      scale = maxVals - minVals))
```

```
# No Hidden Layer
neuralzero = neuralnet(r_COI_nat~., data= scaled_train, hidden=0,linear.output=TRUE, threshold=0.01, stepmax=20000, lifesign='full', l
```

```
hidden: 0    thresh: 0.01    rep: 1/1    steps:     500 min thresh: 8.15877246623922
                                            1000 min thresh: 3.09502379801426
                                            1500 min thresh: 1.93919358186797
                                            2000 min thresh: 0.809717638289829
                                            2500 min thresh: 0.308796459801607
                                            3000 min thresh: 0.170084624034912
                                            3500 min thresh: 0.072211049560705
                                            4000 min thresh: 0.034038443190482
                                            4500 min thresh: 0.0150132861984594
                                            4780 error: 53.14168 time: 1.59 secs
```

```
str(neuralzero)
```

```
List of 14
 $ call          : language neuralnet(formula = r_COI_nat ~ ., data = scaled_train, hidden = 0, threshold = 0.01,     stepmax = 20000, lifes|
 $ response           : num [1:14535, 1] 0.311 0.48 0.575 0.379 0.382 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:14535] "1" "2" "3" "4" ...
  .. ..$ : chr "r_COI_nat"
 $ covariate          : num [1:14535, 1:16] 0.22 0.128 0.273 0.657 0.328 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:16] "county_fips" "life_expectancy" "percent_vaccinated" "pop" ...
 $ model.list         :List of 2
  ..$ response : chr "r_COI_nat"
  ..$ variables: chr [1:16] "county_fips" "life_expectancy" "percent_vaccinated" "pop" ...
 $ err.fct            :function (x, y)
  ..- attr(*, "type")= chr "sse"
 $ act.fct            :function (x)
  ..- attr(*, "type")= chr "logistic"
 $ linear.output      : logi TRUE
 $ data               :'data.frame':    14535 obs. of  17 variables:
  ..$ county_fips               : num [1:14535] 0.22 0.128 0.273 0.657 0.328 ...
  ..$ r_COI_nat                 : num [1:14535] 0.311 0.48 0.575 0.379 0.382 ...
  ..$ life_expectancy           : num [1:14535] 0.45 0.526 0.592 0.433 0.541 ...
  ..$ percent_vaccinated        : num [1:14535] 0.652 0.742 0.682 0.652 0.848 ...
  ..$ pop                       : num [1:14535] 0.00766 0.05777 0.04262 0.00537 0.00108 ...
  ..$ median_household_income   : num [1:14535] 0.247 0.39 0.376 0.208 0.296 ...
```

```
  ..$ avenum_phys_unhealthydays : num [1:14535] 0.414 0.258 0.264 0.381 0.2 ...
  ..$ avenum_ment_unhealthydays : num [1:14535] 0.564 0.492 0.402 0.414 0.316 ...
  ..$ percent_lowbirthweight    : num [1:14535] 0.557 0.449 0.217 0.413 0.102 ...
  ..$ percent_exercise          : num [1:14535] 0.735 0.991 0.916 0.677 0.603 ...
  ..$ pcp_ratio                 : num [1:14535] 0.0694 0.0235 0.0378 0.0978 0.1108 ...
  ..$ dentist_ratio             : num [1:14535] 0.0564 0.0346 0.0351 0.0814 0.0708 ...
  ..$ mhp_ratio                 : num [1:14535] 0.02656 0.00443 0.00963 0.01908 0.07335 ...
  ..$ percent_uninsured         : num [1:14535] 0.433 0.194 0.152 0.319 0.129 ...
  ..$ percent_severehousing     : num [1:14535] 0.194 0.265 0.15 0.168 0.076 ...
  ..$ percent_broadband         : num [1:14535] 0.736 0.884 0.931 0.778 0.722 ...
  ..$ percent_limitedhealthyfood: num [1:14535] 0.2856 0.0984 0.1442 0.0504 0.0731 ...
 $ exclude            : NULL
 $ net.result         :List of 1
  ..$ : num [1:14535, 1] 0.323 0.565 0.6 0.318 0.559 ...
 $ weights            :List of 1
  ..$ :List of 1
  .. ..$ : num [1:17, 1] -0.00498 0.0204 0.38483 0.20779 -0.05509 ...
 $ generalized.weights:List of 1
  ..$ : num [1:14535, 1:16] 0.0933 0.083 0.085 0.094 0.0827 ...
 $ startweights       :List of 1
  ..$ :List of 1
  .. ..$ : num [1:17, 1] 0.4137 -0.3462 1.1343 -0.0765 -1.2503 ...
 $ result.matrix      : num [1:20, 1] 5.31e+01 9.25e-03 4.78e+03 -4.98e-03 2.04e-02 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:20] "error" "reached.threshold" "steps" "Intercept.to.r_COI_nat" ...
  .. ..$ : NULL
 - attr(*, "class")= chr "nn"
```

```r
summary(neuralzero)
```

```
                   Length Class      Mode
call                    9 -none-     call
response            14535 -none-     numeric
covariate          232560 -none-     numeric
model.list              2 -none-     list
err.fct                 1 -none-     function
act.fct                 1 -none-     function
linear.output           1 -none-     logical
data                   17 data.frame list
exclude                 0 -none-     NULL
net.result              1 -none-     list
weights                 1 -none-     list
```

```
generalized.weights      1 -none-     list
startweights             1 -none-     list
result.matrix           20 -none-     numeric
```

```
plot(neuralzero)
```

```
# One Hidden Layer
neuralone = neuralnet(r_COI_nat~., data= scaled_train, hidden = 1, linear.output=TRUE, threshold=0.01, stepmax=15000, lifesign='full',
```

```
hidden: 1    thresh: 0.01    rep: 1/1    steps:     500 min thresh: 1.93841166922843
                                            1000 min thresh: 0.907146916951592
                                            1500 min thresh: 0.657570465021893
                                            2000 min thresh: 0.318897605961635
                                            2500 min thresh: 0.255958847706694
                                            3000 min thresh: 0.114903028270988
                                            3500 min thresh: 0.0618872364332845
                                            4000 min thresh: 0.0294876340825522
                                            4500 min thresh: 0.016177811530856
                                            4888 error: 50.08657 time: 2.91 secs
```

```
summary(neuralone)
```

```
                   Length Class      Mode
call                    9 -none-     call
response            14535 -none-     numeric
covariate          232560 -none-     numeric
model.list              2 -none-     list
err.fct                 1 -none-     function
act.fct                 1 -none-     function
linear.output           1 -none-     logical
data                   17 data.frame list
exclude                 0 -none-     NULL
net.result              1 -none-     list
weights                 1 -none-     list
generalized.weights     1 -none-     list
startweights            1 -none-     list
result.matrix          22 -none-     numeric
```

```
plot(neuralone)
```

Calculating OSR2

```
#Neural Zero
neuralzero.test = predict(neuralzero, newdata=scaled_test)
neuralone.test = predict(neuralone,newdata=scaled_test)
summary(neuralzero.test)
```

```
       V1
 Min.   :-0.2944
 1st Qu.: 0.2904
 Median : 0.3962
 Mean   : 0.4010
 3rd Qu.: 0.5103
 Max.   : 1.0712
```

```
m = min(train.ANN$r_COI_nat)
M = max(train.ANN$r_COI_nat)

neuralzeropred.nn = (neuralzero.test * (M - m)) + m
neuralonepred.nn= (neuralone.test * (M - m)) + m

summary(neuralzeropred.nn)
```

```
       V1
 Min.   :-25.44
 1st Qu.: 27.08
 Median : 36.58
 Mean   : 37.01
 3rd Qu.: 46.84
 Max.   : 97.21
```

```
train.mean = mean(train.ANN$r_COI_nat)
SSE.test_zero = sum((neuralzeropred.nn - test.ANN$r_COI_nat)^2)
SSE.test_one = sum((neuralonepred.nn - test.ANN$r_COI_nat)^2)

SST.test = sum((train.mean - test.ANN$r_COI_nat)^2)

OSR2_zero <- 1 - SSE.test_zero/SST.test
OSR2_one <- 1 - SSE.test_one/SST.test

cat("\n \n OSR2 for the Zero Hidden Layer NN is: \n")
```

```
 OSR2 for the Zero Hidden Layer NN is:
```

```
OSR2_zero
```

```
[1] 0.8100496
```

```
cat("\n \n OSR2 for the One Hidden Layer NN is: \n")
```

```
 OSR2 for the One Hidden Layer NN is:
```

```
OSR2_one
```

```
[1] 0.8220976
```

Comparing with the RF performance:

```
best_nn_OCR = max(OSR2_one,OSR2_zero)

OSR_rf - best_nn_OCR
```

```
[1] 0.1122857
```

The Random Forest Performs approximately 11% better than the neural network performed. This may be attributable to the fact that the Neural Network as defined above is not fully optimized (as more nodes and layers may have a positive benefit on the overall performance of the model).

References

https://www.diversitydatakids.org/research-library/child-opportunity-index-30-2023-county-data

https://www.diversitydatakids.org/research-library/research-brief/what-child-opportunity

https://www.ers.usda.gov/data-products/urban-influence-codes