Milestone 2
Jasper Evans, Krehl Kasayan, Paul Kiefer, Pablo Suarez
INST737
4/8/2024

Introduction: Before we could begin building and assessing our predictive models, we had to transform raw American Community Survey data into scaled metrics that can be compared between census tracts. For context, the American Community Survey is an annual survey of a sample of US residents at various levels of geography. Because it takes time to collect those samples, survey results are provided in five-year spans.

Those variables are:

1. avg_bed: The average number of beds in a housing unit at the tract level as of the 2016-2020 ACS.
2. tract_homevalue_2020: The mean self-reported home value at the tract level as of the 2016-2020 ACS.
3. tract_medage_2020: The median age of residents at the tract level as of the 2016-2020 ACS.
4. tract_medincome_2020: The median income of households at the tract level as of the 2016-2020 ACS.
5. tract_medincome_2010: The median income of households at the tract level as of the 2006-2010 ACS.
6. foreclosure_pc_2010: The number of foreclosures per 1,000 residents at the tract level during the height of the Great Recession from 2008-2010.
7. pct_built_2020_later: The percentage of housing units in a tract built in or after 2020.
8. pct_built_2010_2019: The percentage of housing units in a tract built between 2010-2019.
9. pct_built_2000_2009: The percentage of housing units in a tract built between 2000-2009.
10. pct_built_1990_1999: The percentage of housing units in a tract built between 1990-1999.
11. pct_built_1980_1989: The percentage of housing units in a tractbuilt between 1980-1989.
12. pct_built_1970_1979: The percentage of housing units in a tract built between 1970-1979.
13. pct_built_pre_1960: The percentage of housing units in a tract built prior to 1960.
14. pct_0_bd: The percentage of housing units with no separate bedroom (i.e. studios) in a tract.
15. pct_1_bd: The percentage of housing units with one bedroom in a tract.
16. pct_2_bd: The percentage of housing units with two bedrooms in a tract.
17. pct_3_bd: The percentage of housing units with 3 bedrooms in a tract.
18. pct_4_more_bd: The percentage of housing units with 4 or more bedrooms in a tract.
19. poverty_2010: The percentage of a tract’s population with incomes below the federal poverty line as of the 2006-2010 ACS.

20. poverty_2020: The percentage of a tract's population with incomes below the federal poverty line as of the 2016-2020 ACS.
21. nhwhite_2010: The percentage of a tract's population that self-identified as non-Hispanic white in the 2006-2010 ACS.
22. nhwhite_2020: The percentage of a tract's population that self-identified as non-Hispanic white in the 2016-2020 ACS.
23. mortgaged_2010: The percentage of all housing units in a tract with a mortgage or similar loan in the 2006-2010 ACS.
24. mortgaged_2015: The percentage of all housing units in a tract with a mortgage or similar loan in the 2011-2015 ACS.
25. mortgaged_2020: The percentage of all housing units in a tract with a mortgage or similar loan in the 2016-2020 ACS.
26. ownoccupied_2010: The percentage of all housing units in a tract that were owner-occupied in the 2006-2010 ACS.
27. ownoccupied_2015: The percentage of all housing units in a tract that were owner-occupied n in the 2011-2015 ACS.
28. ownoccupied_2020: The percentage of all housing units in a tract that were owner-occupied in the 2016-2020 ACS.
29. mortgage_change_2010_2015: The change in the percentage of all housing units in a tract with a mortgage or similar loan between the 2006-2010 ACS and 2011-2015 ACS.
30. mortgage_change_2015_2020: The change in the percentage of all housing units in a tract with a mortgage or similar loan between the 2011-2015 ACS and 2016-2020 ACS.
31. mortgage_change_2010_2020: The change in the percentage of all housing units in a tract with a mortgage or similar loan between the 2006-2010 ACS and 2011-2015 ACS.
32. ownoccupied_change_2010_2015: The change in the percentage of all housing units in a tract that were owner-occupied between the 2006-2010 ACS and 2011-2015 ACS.
33. ownoccupied_change_2015_2020: The change in the percentage of all housing units in a tract that were owner-occupied between the 2011-2015 ACS and 2016-2020 ACS.
34. ownoccupied_change_2010_2020: The change in the percentage of all housing units in a tract that were owner-occupied between the 2006-2010 ACS and 2016-2020 ACS.
35. poverty_change_2010_2020: The change in the percentage of all residents in a tract living below the federal poverty line between the 2006-2010 ACS and 2016-2020 ACS.
36. nhwhite_change_2010_2020: The change in the percentage tract residents who self-identified as non-Hispanic white between the 2006-2010 ACS and 2016-2020 ACS.
37. medincome_change_2010_2015: The change in the reported tract-level median household income between the 2006-2010 ACS and 2011-2015 ACS.
38. medincome_change_2015_2020: The change in the reported tract-level median household income between the 2011-2015 ACS and 2016-2020 ACS.
39. medincome_change_2010_2020: The change in the reported tract-level median household income between the 2006-2010 ACS and 2016-2020 ACS.
40. pop_change_pct: The change in the reported tract-level population between the 2006-2010 ACS and 2016-2020 ACS.

These data points are all compared to our response variable (or a version of it): foreclosure_pc_2020.

That variable represents the number of reported foreclosures in a census tract between 2011-2023 divided by the estimated population of the tract in the 2016-2020 ACS. That may seem like an imperfect point of comparison, and in some ways, it is: ideally, we would work with foreclosures per capita in the most recent year for which we have both foreclosure and population data available.

Unfortunately, the Prince George's County dataset that records foreclosures by address contains very few records dated after 2020 – which may mean some foreclosures have been omitted – so we would have hardly any samples if we were to limit ourselves to recent years.

Instead, we consider all foreclosures between the end of the Great Recession – defined in our case as 2011, though that is up for debate – and the end of our dataset in 2011. This gives us a larger sample of tracts to assess, and we include variables measuring characteristics of each tract at various points in that decade to ensure we are considering changes in a tract's risks as part of our analysis.

We may have too many variables for a sample size of only roughly 170 tracts, and in the next step of this process, we may cut back the number of variables while attempting to increase our sample size.

None of the models attempted in this iteration of the project were ideal, and there are certainly other variables that could be more predictive than what we have available. Federal Home Mortgage Disclosure Act (HMDA) data, for instance, could give us tract-level statistics on the percentage of mortgage loans that were denied over a given time period, the percentage of approved mortgage loans that were categorized as refinancing over a given period of time, and the percentage of mortgage loans given for non-owner-occupied properties over a given time period. Unfortunately, the census tract ID numbers included in the HMDA database are incomplete, and we have not yet found a way to correct them.

Another possible tweak would involve shifting to census block-level analysis, which would increase the size of our sample but decrease the number of statistics available on the residents and housing units in a given block. Block-level statistics are only collected during decennial censuses, and decennial censuses are less comprehensive than the American Community Surveys.

Question 1: Linear Regressions

Divide your dataset into training and testing sets as we have seen in class and report:

Linear Regression Parameters:

Because our dataset included some columns that do not contain independent variables, we began this process by defining the independent variables of interest for our analysis using the following code:

```
variables_of_interest <- c("avg_bed", "tract_homevalue_2020",
"tract_medage_2020", "tract_medincome_2020", "tract_medincome_2010",
"foreclosure_pc_2010", "pct_built_2020_later", "pct_built_2010_2019",
"pct_built_2000_2009", "pct_built_1990_1999", "pct_built_1980_1989",
"pct_built_1970_1979",  "pct_built_pre_1960", "pct_0_bd", "pct_1_bd",
"pct_2_bd", "pct_3_bd", "pct_4_more_bd", "poverty_2010", "poverty_2020",
"nhwhite_2010", "nhwhite_2020", "mortgaged_2010", "mortgaged_2015",
"mortgaged_2020", "ownoccupied_2010", "ownoccupied_2015",
"ownoccupied_2020", "mortgage_change_2010_2015",
"mortgage_change_2015_2020","mortgage_change_2010_2020","ownoccupied_change
_2010_2015", "ownoccupied_change_2015_2020","ownoccupied_change_2010_2020",
"poverty_change_2010_2020","nhwhite_change_2010_2020","medincome_change_201
0_2015","medincome_change_2015_2020", "medincome_change_2010_2020",
"pop_change_pct")
```

We then divide our dataset into a training set – a random sample of 70% of our dataset – and a test set made up of a random sample of 30% of our dataset.

For each independent variable in your model, compute a linear regression with respect to the dependent feature and report:

We then use the following code to loop through each of the 40 independent variables and calculate the **intercept, correlation coefficient and mean squared error of each independent variable relative to the response variable, as well as whether the independent variable is predictive.**

```
for (variable in variables_of_interest) {
  # Perform linear regression
  model <- lm(foreclosure_pc_2020 ~ ., data = pg_foreclosures_train[,
c(variable, "foreclosure_pc_2020")])

  # Extract required information
  intercept <- coef(model)[1]
  coefficient <- coef(model)[2]

  # Check if the coefficient is statistically significant
```

```
  p_value <-
summary(model)$coefficients[which(rownames(summary(model)$coefficients) ==
variable), "Pr(>|t|)"]
  is_predictive <- ifelse(p_value < 0.05, "Yes", "No")

  # Compute residuals
  residuals <- resid(model)

  # Compute correlation between predicted and real values
  correlation <- cor(predict(model),
pg_foreclosures_train$foreclosure_pc_2020 )

  # Compute mean square error
  mse <- mean((predict(model) - pg_foreclosures_train$foreclosure_pc_2020
)^2)

  # Print results
  cat("Variable:", variable, "\n")
  cat("Intercept:", intercept, "\n")
  cat("Coefficient:", coefficient, "\n")
  cat("Is it a predictive feature?:", is_predictive, "\n")
  cat("Correlation:", correlation, "\n")
  cat("Mean Square Error:", mse, "\n")
  cat("\n")
}
```

That produces the following results:

```
Variable: avg_bed
Intercept: 19.06023
Coefficient: 16.9744
Is it a predictive feature?: Yes
Correlation: 0.3746637
Mean Square Error: 1109.453

Variable: tract_homevalue_2020
Intercept: 81.22493
Coefficient: -0.00003569445
Is it a predictive feature?: No
Correlation: 0.07536366
Mean Square Error: 1283.291
```

```
Variable: tract_medage_2020
Intercept: -39.79579
Coefficient: 2.790767
Is it a predictive feature?: Yes
Correlation: 0.4556463
Mean Square Error: 1022.671

Variable: tract_medincome_2020
Intercept: 41.09195
Coefficient: 0.0003171854
Is it a predictive feature?: Yes
Correlation: 0.2672517
Mean Square Error: 1198.441

Variable: tract_medincome_2010
Intercept: 37.44741
Coefficient: 0.0004233989
Is it a predictive feature?: Yes
Correlation: 0.3053277
Mean Square Error: 1170.304

Variable: foreclosure_pc_2010
Intercept: 20.68089
Coefficient: 3.992378
Is it a predictive feature?: Yes
Correlation: 0.6491367
Mean Square Error: 746.7816

Variable: pct_built_2020_later
Intercept: 68.74373
Coefficient: 360.2854
Is it a predictive feature?: No
Correlation: 0.1201536
Mean Square Error: 1271.989

Variable: pct_built_2010_2019
Intercept: 69.72607
Coefficient: 9.976878
Is it a predictive feature?: No
Correlation: 0.02755463
Mean Square Error: 1289.642

Variable: pct_built_2000_2009
```

```
Intercept: 62.20988
Coefficient: 83.76972
Is it a predictive feature?: Yes
Correlation: 0.2605284
Mean Square Error: 1203.021

Variable: pct_built_1990_1999
Intercept: 58.84653
Coefficient: 73.53769
Is it a predictive feature?: Yes
Correlation: 0.2630944
Mean Square Error: 1201.287

Variable: pct_built_1980_1989
Intercept: 63.33133
Coefficient: 48.99304
Is it a predictive feature?: No
Correlation: 0.146484
Mean Square Error: 1262.928

Variable: pct_built_1970_1979
Intercept: 70.38708
Coefficient: -0.9380611
Is it a predictive feature?: No
Correlation: 0.002405406
Mean Square Error: 1290.614

Variable: pct_built_pre_1960
Intercept: 89.31805
Coefficient: -46.31767
Is it a predictive feature?: Yes
Correlation: 0.326355
Mean Square Error: 1153.161

Variable: pct_0_bd
Intercept: 80.02659
Coefficient: -482.1701
Is it a predictive feature?: Yes
Correlation: 0.3962199
Mean Square Error: 1088.007

Variable: pct_1_bd
Intercept: 85.44572
```

```
Coefficient: -133.0005
Is it a predictive feature?: Yes
Correlation: 0.4482354
Mean Square Error: 1031.317

Variable: pct_2_bd
Intercept: 87.1228
Coefficient: -85.43262
Is it a predictive feature?: Yes
Correlation: 0.3832088
Mean Square Error: 1101.095

Variable: pct_3_bd
Intercept: 33.78992
Coefficient: 116.2271
Is it a predictive feature?: Yes
Correlation: 0.422721
Mean Square Error: 1059.997

Variable: pct_4_more_bd
Intercept: 53.82052
Coefficient: 46.38325
Is it a predictive feature?: Yes
Correlation: 0.305918
Mean Square Error: 1169.838

Variable: poverty_2010
Intercept: 87.40257
Coefficient: -254.5372
Is it a predictive feature?: Yes
Correlation: 0.3931962
Mean Square Error: 1091.087

Variable: poverty_2020
Intercept: 90.8764
Coefficient: -265.7121
Is it a predictive feature?: Yes
Correlation: 0.3723224
Mean Square Error: 1111.711

Variable: nhwhite_2010
Intercept: 78.41266
Coefficient: -54.77476
```

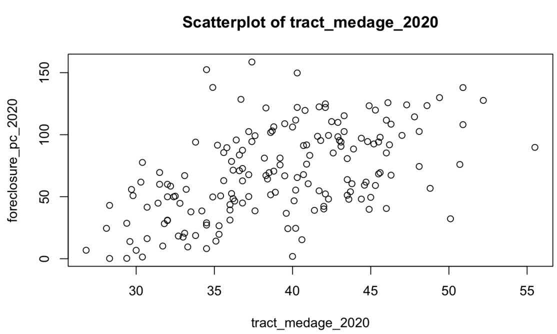
```
Is it a predictive feature?: Yes
Correlation: 0.2512609
Mean Square Error: 1209.142

Variable: nhwhite_2020
Intercept: 77.66176
Coefficient: -64.05339
Is it a predictive feature?: Yes
Correlation: 0.2492505
Mean Square Error: 1210.441

Variable: mortgaged_2010
Intercept: 20.78945
Coefficient: 93.89575
Is it a predictive feature?: Yes
Correlation: 0.607003
Mean Square Error: 815.0888

Variable: mortgaged_2015
Intercept: 24.32348
Coefficient: 94.48867
Is it a predictive feature?: Yes
Correlation: 0.5803045
Mean Square Error: 856.0007

Variable: mortgaged_2020
Intercept: 24.26386
Coefficient: 93.65878
Is it a predictive feature?: Yes
Correlation: 0.5779621
Mean Square Error: 859.5023

Variable: ownoccupied_2010
Intercept: 22.55678
Coefficient: 77.30333
Is it a predictive feature?: Yes
Correlation: 0.5639822
Mean Square Error: 880.106

Variable: ownoccupied_2015
Intercept: 24.97609
Coefficient: 77.30393
Is it a predictive feature?: Yes
```

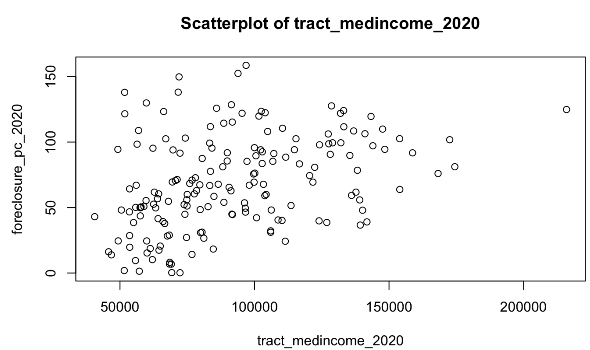
```
Correlation: 0.5406337
Mean Square Error: 913.3927

Variable: ownoccupied_2020
Intercept: 24.14714
Coefficient: 76.14728
Is it a predictive feature?: Yes
Correlation: 0.5469823
Mean Square Error: 904.4811

Variable: mortgage_change_2010_2015
Intercept: 66.7023
Coefficient: -87.32843
Is it a predictive feature?: Yes
Correlation: 0.1773345
Mean Square Error: 1250.035

Variable: mortgage_change_2015_2020
Intercept: 70.2538
Coefficient: 0.6904628
Is it a predictive feature?: No
Correlation: 0.001342858
Mean Square Error: 1290.619

Variable: mortgage_change_2010_2020
Intercept: 68.15521
Coefficient: 58.77525
Is it a predictive feature?: No
Correlation: 0.1449546
Mean Square Error: 1263.503

Variable: ownoccupied_change_2010_2015
Intercept: 67.35991
Coefficient: -92.56352
Is it a predictive feature?: No
Correlation: 0.175742
Mean Square Error: 1250.761

Variable: ownoccupied_change_2015_2020
Intercept: 69.34088
Coefficient: 46.3183
Is it a predictive feature?: No
Correlation: 0.08282612
```

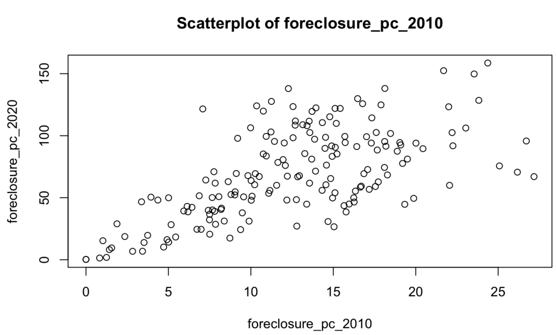
```
Mean Square Error: 1281.768

Variable: ownoccupied_change_2010_2020
Intercept: 69.90148
Coefficient: -30.88702
Is it a predictive feature?: No
Correlation: 0.0757052
Mean Square Error: 1283.225

Variable: poverty_change_2010_2020
Intercept: 69.74779
Coefficient: 49.74486
Is it a predictive feature?: No
Correlation: 0.0652858
Mean Square Error: 1285.121

Variable: nhwhite_change_2010_2020
Intercept: 72.14002
Coefficient: 56.55597
Is it a predictive feature?: No
Correlation: 0.1016417
Mean Square Error: 1277.288

Variable: medincome_change_2010_2015
Intercept: 70.38322
Coefficient: -2.950833
Is it a predictive feature?: No
Correlation: 0.01251915
Mean Square Error: 1290.42

Variable: medincome_change_2015_2020
Intercept: 74.16712
Coefficient: -24.11258
Is it a predictive feature?: No
Correlation: 0.1140783
Mean Square Error: 1273.826

Variable: medincome_change_2010_2020
Intercept: 73.96636
Coefficient: -18.26635
Is it a predictive feature?: No
Correlation: 0.1064072
Mean Square Error: 1276.009
```

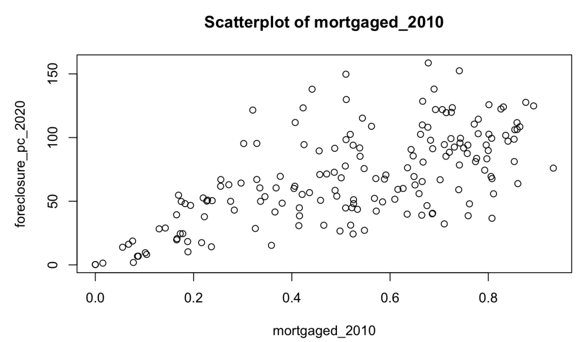
```
Variable: pop_change_pct
Intercept: 71.01631
Coefficient: -9.99906
Is it a predictive feature?: No
Correlation: 0.06121883
Mean Square Error: 1285.785
```

We can also plot the relationships between our individual independent variables and our response variables using scatter plots. For example:









Which are the most predictive features according to the training data?

Evidently, the most predictive features are:

1. mortgaged_2010/mortgaged_2015/mortgaged_2020. These are perhaps a little too obvious -- zip codes with more mortgaged homes are intuitively more likely to have a higher number of foreclosures per capita.

2. foreclosure_pc_2010. Once again, this is intuitive -- tracts that saw more foreclosures per capita during the height of the great recession likely have risk factors (included in our list of variables or not) that remained after 2010.

3. ownoccupied_2010/ownoccupied_2015/ownoccupied_2020. The reasons for this relationship may be similar to the reasons for the relationship between the percentage of units with a mortgage and the per capita rate of foreclosure.

4. tract_medage_2020. According to the linear model, tracts with a higher median age are more likely to see higher rates of foreclosure per capita.

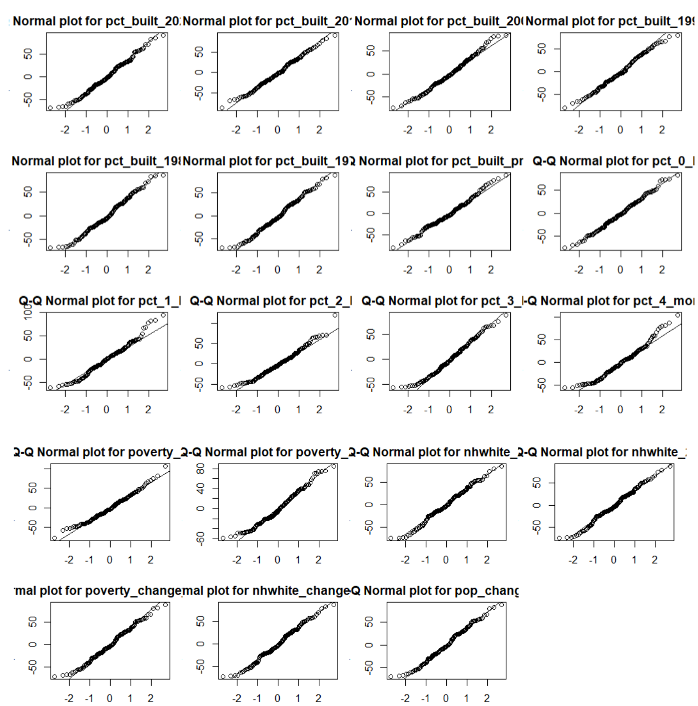
5. pct_0_bd/pct_1_bd/pct_2_bd/etc. The nature of the statistical relationship between the percentage of units in a tract with a given number of beds and the rate of foreclosures per capita is both difficult to explain and difficult to describe.

6. poverty_2010. The tract-level poverty rate in 2020 is more predictive than the poverty rate in 2010 and the median income at the tract level in 2010 or 2020. This may be misleading -- a tract that saw a high rate of foreclosures between 2010-2020 may be poorer as a result of those foreclosures.

What are the residuals? Is a linear regression applicable to your problem?

Broadly speaking, the residuals for almost all variables follow a normal distribution, meaning a core assumption of linear regression (that residuals are normally distributed) are met.

That indicates that linear regression is generally applicable to our problem, though some variables – including pct_1_bd – may merit further investigation.



Use the trained model to predict using your testing data. Show results together with confidence and prediction bands. Report prediction accuracy using (1) the correlation between the predicted and real values and (2) the mean square error between the two.

Because of the number of independent variables within our dataset, we divided the variables of interest into subsets and wrote following code to loop through each variable to create a linear regression model, predict using the testing data, calculate the correlation and mean squared error, and plot the results:

```
for (variable in variables_of_interest_subset_1) {
  # Perform linear regression
  model <- lm(foreclosure_pc_2020 ~ ., data = pg_foreclosures_train[,
c(variable, "foreclosure_pc_2020")])

  # Use the trained model to predict on testing data and obtain confidence
intervals
  predictions <- predict(model, newdata = pg_foreclosures_test, interval =
"confidence", level = 0.95)

  # Calculate correlation between predicted and actual values
  correlation_test <- cor(predictions[, "fit"],
pg_foreclosures_test$foreclosure_pc_2020)

  # Calculate mean square error
  mse_test <- mean((predictions[, "fit"] -
pg_foreclosures_test$foreclosure_pc_2020)^2)

  # Plot predicted vs. actual values
  plot(pg_foreclosures_test$foreclosure_pc_2020, predictions[, "fit"],
       main = paste("Predicted vs. Actual for", variable),
       xlab = "Actual Values", ylab = "Predicted Values", ylim =
range(c(predictions[, "fit"], pg_foreclosures_test$foreclosure_pc_2020)))
  abline(0, 1, col = "red") # Add a diagonal line for reference

  # Print correlation and mean square error
  cat("Variable:", variable, "\n")
  cat("Correlation on Testing Data:", correlation_test, "\n")
  cat("Mean Square Error on Testing Data:", mse_test, "\n")
  cat("\n")
}
```

We repeat the same code for each subset of independent variables and get the following results:

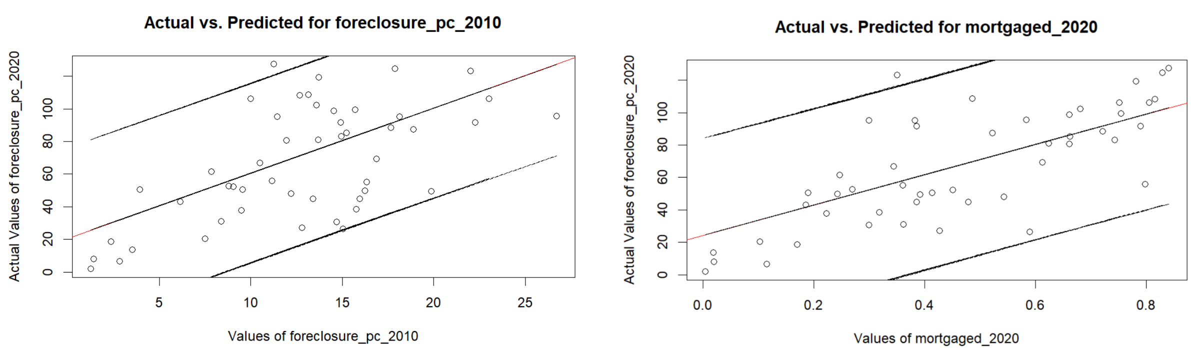
```
Variable: avg_bed
Correlation on Testing Data: 0.5117439
Mean Square Error on Testing Data: 850.9713

Variable: tract_homevalue_2020
Correlation on Testing Data: -0.3193279
Mean Square Error on Testing Data: 1170.482

Variable: tract_medage_2020
Correlation on Testing Data: 0.6158819
Mean Square Error on Testing Data: 717.7623

Variable: tract_medincome_2020
Correlation on Testing Data: 0.3378183
Mean Square Error on Testing Data: 999.4049

Variable: tract_medincome_2010
Correlation on Testing Data: 0.3635291
Mean Square Error on Testing Data: 979.4372

Variable: medincome_change_2010_2015
Correlation on Testing Data: 0.08832201
Mean Square Error on Testing Data: 1129.182

Variable: medincome_change_2015_2020
Correlation on Testing Data: 0.1959899
Mean Square Error on Testing Data: 1091.314

Variable: medincome_change_2010_2020
Correlation on Testing Data: 0.1258555
Mean Square Error on Testing Data: 1120.687

Variable: foreclosure_pc_2010
Correlation on Testing Data: 0.3411653
Mean Square Error on Testing Data: 1104.988

Variable: pct_built_2020_later
Correlation on Testing Data: 0.05592942
Mean Square Error on Testing Data: 1128.079

Variable: pct_built_2010_2019
Correlation on Testing Data: -0.1423304
Mean Square Error on Testing Data: 1191.664

Variable: pct_built_2000_2009
Correlation on Testing Data: 0.1736346
Mean Square Error on Testing Data: 1140.387
```

```
Variable: pct_built_1990_1999
Correlation on Testing Data: 0.2315283
Mean Square Error on Testing Data: 1073.282

Variable: pct_built_1980_1989
Correlation on Testing Data: 0.1130736
Mean Square Error on Testing Data: 1117.865

Variable: pct_built_1970_1979
Correlation on Testing Data: 0.05390233
Mean Square Error on Testing Data: 1132.112

Variable: pct_built_pre_1960
Correlation on Testing Data: 0.1382384
Mean Square Error on Testing Data: 1211.898

Variable: pct_0_bd
Correlation on Testing Data: 0.5573989
Mean Square Error on Testing Data: 778.2076

Variable: pct_1_bd
Correlation on Testing Data: 0.476484
Mean Square Error on Testing Data: 876.044

Variable: pct_2_bd
Correlation on Testing Data: 0.4531857
Mean Square Error on Testing Data: 901.0094

Variable: pct_3_bd
Correlation on Testing Data: 0.3998965
Mean Square Error on Testing Data: 949.6106

Variable: pct_4_more_bd
Correlation on Testing Data: 0.4270028
Mean Square Error on Testing Data: 930.5358

Variable: poverty_2010
Correlation on Testing Data: 0.5019256
Mean Square Error on Testing Data: 843.8109

Variable: poverty_2020
Correlation on Testing Data: 0.05570013
Mean Square Error on Testing Data: 1214

Variable: nhwhite_2010
Correlation on Testing Data: 0.3283699
```

```
Mean Square Error on Testing Data: 1059.075

Variable: nhwhite_2020
Correlation on Testing Data: 0.4733185
Mean Square Error on Testing Data: 1049.925

Variable: poverty_change_2010_2020
Correlation on Testing Data: 0.5307815
Mean Square Error on Testing Data: 1096.449

Variable: nhwhite_change_2010_2020
Correlation on Testing Data: 0.05467011
Mean Square Error on Testing Data: 1128.522

Variable: pop_change_pct
Correlation on Testing Data: 0.1093619
Mean Square Error on Testing Data: 1120.843

Variable: mortgaged_2010
Correlation on Testing Data: 0.6191706
Mean Square Error on Testing Data: 699.3978

Variable: mortgaged_2015
Correlation on Testing Data: 0.5824819
Mean Square Error on Testing Data: 768.5658

Variable: mortgaged_2020
Correlation on Testing Data: 0.5966077
Mean Square Error on Testing Data: 735.8966

Variable: ownoccupied_2010
Correlation on Testing Data: 0.6195836
Mean Square Error on Testing Data: 695.9821

Variable: ownoccupied_2015
Correlation on Testing Data: 0.5838292
Mean Square Error on Testing Data: 744.8564

Variable: ownoccupied_2020
Correlation on Testing Data: 0.5924652
Mean Square Error on Testing Data: 735.3165

Variable: mortgage_change_2010_2015
Correlation on Testing Data: 0.06476888
Mean Square Error on Testing Data: 1146.255

Variable: mortgage_change_2015_2020
```

```
Correlation on Testing Data: -0.04575922
Mean Square Error on Testing Data: 1142.707

Variable: mortgage_change_2010_2020
Correlation on Testing Data: 0.09506252
Mean Square Error on Testing Data: 1123.147

Variable: ownoccupied_change_2010_2015
Correlation on Testing Data: 0.1337618
Mean Square Error on Testing Data: 1110.936

Variable: ownoccupied_change_2015_2020
Correlation on Testing Data: 0.1337533
Mean Square Error on Testing Data: 1110.026

Variable: ownoccupied_change_2010_2020
Correlation on Testing Data: 0.005473917
Mean Square Error on Testing Data: 1134.188
```

It would be too space-intensive to include plots for each of the 40 variables, but here is sample:



We did not successfully plot both confidence and prediction bands; we struggled to apply the matlines() function to the loop we used to train, test and plot a linear regression model for each independent variable.

B. Multivariate regressions

Show whether considering combinations of independent features improves the prediction results. Evaluate different combinations of features as applicable and report those that improve the results shown in question (a).

As a starting point, we use the following code to identify collinear groups of variables using a threshold correlation coefficient of 0.8:

```
# Calculating the correlation matrix
correlation_matrix <- cor(pg_foreclosures_per_tract[variables_of_interest])

# Set a correlation threshold
threshold <- 0.85

# Find highly correlated variable pairs
highly_correlated <- which(correlation_matrix > threshold &
correlation_matrix < 1, arr.ind = TRUE)

# Print highly correlated variable pairs
for (i in 1:nrow(highly_correlated)) {
  var1 <- rownames(correlation_matrix)[highly_correlated[i, 1]]
  var2 <- colnames(correlation_matrix)[highly_correlated[i, 2]]
  corr <- correlation_matrix[highly_correlated[i, 1], highly_correlated[i,
2]]
  cat("Variables", var1, "and", var2, "are highly correlated (correlation
=", corr, ")\n")
}
```

For our first experimental set of independent variables, we select the most predictive independent variable from each group of collinear variables, add any variables that were not collinear with other variables, and create the following subset:

```
test_variables_1 <- c("avg_bed", "tract_homevalue_2020",
"tract_medage_2020", "tract_medincome_2020",   "foreclosure_pc_2010",
"pct_built_2020_later", "pct_built_2010_2019", "pct_built_2000_2009",
"pct_built_1990_1999", "pct_built_1980_1989", "pct_built_1970_1979",
"pct_built_pre_1960", "poverty_2020", "nhwhite_2020", "mortgaged_2010",
"mortgage_change_2010_2015",
"poverty_change_2010_2020","nhwhite_change_2010_2020","medincome_change_201
0_2015", "medincome_change_2015_2020", "medincome_change_2010_2020",
"pop_change_pct")
```

We then run the following code to divide our dataset into randomized training and test sets, train the linear regression model using the selected variables, use the trained model to predict the response variable, and calculate the **correlation between real and predicted values, the mean squared error between the real and predicted values, the coefficients for each feature, and the most predictive features of the model:**

```
# Convert tract_number to character before splitting the data
```

```
pg_foreclosures_per_tract$tract_number <-
as.character(pg_foreclosures_per_tract$tract_number)

# Split the data into training and testing sets
set.seed(123) # for reproducibility
train_indices <-
createDataPartition(pg_foreclosures_per_tract$foreclosure_pc_2020, p = 0.8,
list = FALSE)
train_data_multivariate_1 <- pg_foreclosures_per_tract[train_indices, ]
test_data_multivariate_1 <- pg_foreclosures_per_tract[-train_indices, ]

# Train the linear regression model on the training data
lm_model <- lm(foreclosure_pc_2020 ~ ., data = train_data_multivariate_1[,
c("foreclosure_pc_2020", test_variables_1)])

# Use the trained model to predict on the testing data
predicted_values <- predict(lm_model, newdata = test_data_multivariate_1[,
test_variables_1])

# Calculate the correlation between the predicted and real values
correlation <- cor(predicted_values,
test_data_multivariate_1$foreclosure_pc_2020)

# Calculate the mean squared error between the predicted and real values
mse <- mean((predicted_values -
test_data_multivariate_1$foreclosure_pc_2020)^2)

# Obtain coefficients for each feature
coefficients <- coef(lm_model)
print(coefficients)

# Identify the most predictive features
# Absolute values of coefficients can be considered for importance
# Higher absolute values indicate more influence on the prediction
absolute_coefficients <- abs(coefficients[-1]) # Exclude intercept
top_predictive_features <-
names(absolute_coefficients)[order(-absolute_coefficients)][1:5] # Select
top 5 features
```

That initial model produces the following output:

```
Coefficients:
```

```
avg_bed: -3.270173433910
tract_homevalue_2020: -0.000071088618
tract_medage_2020: 0.943090883386
tract_medincome_2020: -0.000007804235
foreclosure_pc_2010: 2.600806436706
pct_built_2020_later: 209.729556742509
pct_built_2010_2019: 78.823749017584
pct_built_2000_2009: 19.867570073357
pct_built_1990_1999: 41.230919498650
pct_built_1980_1989: -35.810641773229
pct_built_1970_1979: 31.290999117118
pct_built_pre_1960: NA
poverty_2020: -107.531979132470
nhwhite_2020: -62.911825462744
mortgaged_2010: 62.545986109374
mortgage_change_2010_2015: 40.321693073811
poverty_change_2010_2020: 58.644184450053
nhwhite_change_2010_2020: 40.137487023411
medincome_change_2010_2015: -108.743283778538
medincome_change_2015_2020: -112.855022494620
medincome_change_2010_2020: 102.354402037439
pop_change_pct: -42.460893747868

Most predictive features:
pct_built_2020_later
medincome_change_2015_2020
medincome_change_2010_2015
poverty_2020
medincome_change_2010_2020

Correlation between predicted and real values: 0.7878868
Mean Squared Error: 434.8585
```

That model did, however, also produce this warning, which indicated some degree of collinearity:

```
Warning: prediction from a rank-deficient fit may be misleading
```

After some experimentation, we reached this set of independent variables that both improved the predictiveness of our model and did not produce a warning about rank-deficient fit:

```
test_variables_3 <- c("mortgaged_2010", "tract_homevalue_2020",
"tract_medage_2020", "tract_medincome_2020",   "foreclosure_pc_2010",
"pct_built_2000_2009", "poverty_2010", "nhwhite_2020",
"mortgage_change_2010_2020",
"poverty_change_2010_2020","nhwhite_change_2010_2020","medincome_change_201
0_2015", "medincome_change_2015_2020", "medincome_change_2010_2020",
"pop_change_pct")
```

That combination of variables produced this output:

```
Coefficients:
mortgaged_2010: 22.797472442326
tract_homevalue_2020: 62.996602305439
tract_medage_2020: -0.000074835506
tract_medincome_2020: -0.000004506972
foreclosure_pc_2010: 2.235976962519
pct_built_2000_2009: 47.156334551554
poverty_2010: -126.862854350235
nhwhite_2020: -71.672688012288
mortgage_change_2010_2020: -64.941360273727
poverty_change_2010_2020: -70.380936131425
nhwhite_change_2010_2020: 60.125954488660
medincome_change_2010_2015: -143.305801287334
medincome_change_2015_2020: -140.909399442653
medincome_change_2010_2020: 128.456180060477
pop_change_pct: -27.094446208530

Most predictive features:
medincome_change_2010_2015
medincome_change_2015_2020
medincome_change_2010_2020
poverty_2010
nhwhite_2020

Correlation between predicted and real values: 0.7934256
Mean Squared Error: 437.2731
```

Of the models we tested, that model was the most successful.

C. Regularization

Repeat experiments in (a) and (b) adding regularization. Do you observe any improvements in the prediction results?

We used the following code to add regularization to our calculation of univariate logistic regressions for each variable:

```
# Set the regularization parameter
lambda <- 0.2

for (variable in variables_of_interest) {
    # Prepare the data
    X <- model.matrix(foreclosure_pc_2020 ~ ., data =
pg_foreclosures_train[, c(variables_of_interest, "foreclosure_pc_2020")])[,
-1]
    y <- pg_foreclosures_train$foreclosure_pc_2020

    # Fit the Lasso regression model
    lasso_model <- glmnet(X, y, alpha = 1, lambda = lambda)

    # Extract coefficients
    coef_idx <- which(colnames(X) == variable)
    coefficient <- coef(lasso_model)[coef_idx]

    # Check if the coefficient is non-zero
    is_predictive <- ifelse(abs(coefficient) > 0, "Yes", "No")

    # Compute residuals
    residuals <- y - predict(lasso_model, newx = X)

    # Compute correlation between predicted and real values
    correlation <- cor(predict(lasso_model, newx = X), y)

    # Compute mean square error
    mse <- mean(residuals^2)

    # Print results
    cat("Variable:", variable, "\n")
    cat("Coefficient:", coefficient, "\n")
    cat("Is it a predictive feature?:", is_predictive, "\n")
    cat("Correlation:", correlation, "\n")
    cat("Mean Square Error:", mse, "\n")
    cat("\n")
}
```

Regularization produces consistent correlation coefficients of roughly 0.87 for each variable – an improvement – though the coefficients for each variable remain distinct.

After regularization, the residuals remain normally distributed.

We used the following code to use the regularized model on our test data:

```
# Convert tract_number to character before splitting the data
pg_foreclosures_per_tract$tract_number <-
as.character(pg_foreclosures_per_tract$tract_number)

# Split the data into training and testing sets
set.seed(123) # for reproducibility
train_indices <-
createDataPartition(pg_foreclosures_per_tract$foreclosure_pc_2020, p = 0.8,
list = FALSE)
train_data_multivariate_1 <- pg_foreclosures_per_tract[train_indices, ]
test_data_multivariate_1 <- pg_foreclosures_per_tract[-train_indices, ]

# Define predictor variables
test_variables_3 <- c("ownoccupied_2010", "tract_homevalue_2020",
"tract_medage_2020", "tract_medincome_2020",   "foreclosure_pc_2010",
"pct_built_2000_2009", "poverty_2010", "nhwhite_2020",
"mortgage_change_2010_2020",
"poverty_change_2010_2020","nhwhite_change_2010_2020","medincome_change_201
0_2015", "medincome_change_2015_2020", "medincome_change_2010_2020",
"pop_change_pct")

# Train the Lasso regression model on the training data
lasso_model <- cv.glmnet(as.matrix(train_data_multivariate_1[,
test_variables_3]),
                         train_data_multivariate_1$foreclosure_pc_2020,
                         alpha = 1) # Use alpha = 1 for Lasso regression

# Use the trained model to predict on the testing data
predicted_values <- predict(lasso_model, newx =
as.matrix(test_data_multivariate_1[, test_variables_3]),
                            s = "lambda.min")
```

```r
# Calculate the correlation between the predicted and real values
correlation <- cor(predicted_values,
test_data_multivariate_1$foreclosure_pc_2020)

# Calculate the mean squared error between the predicted and real values
mse <- mean((predicted_values -
test_data_multivariate_1$foreclosure_pc_2020)^2)

# Obtain coefficients for each feature
coefficients <- coef(lasso_model, s = "lambda.min")
print(coefficients)

# Identify the most predictive features
# Non-zero coefficients indicate predictive features in Lasso regression
non_zero_coefficients <- coefficients[-1] # Exclude intercept
top_predictive_features <-
names(non_zero_coefficients[non_zero_coefficients != 0]) # Select features
with non-zero coefficients

# Print the most predictive features, correlation, and mean squared error
cat("Most predictive features:", top_predictive_features, "\n")
cat("Correlation between predicted and real values:", correlation, "\n")
cat("Mean Squared Error:", mse, "\n")
```

That code produces this output:

```
(Intercept)                  5.86204400487
ownoccupied_2010            53.36979029348
tract_homevalue_2020        -0.00005998712
tract_medage_2020            0.92702987857
tract_medincome_2020         .
foreclosure_pc_2010          2.22242203802
pct_built_2000_2009         48.93320838235
poverty_2010               -84.76763233288
nhwhite_2020               -74.74020982294
mortgage_change_2010_2020  -45.66978992889
poverty_change_2010_2020   -36.48885980937
nhwhite_change_2010_2020    54.28663831913
medincome_change_2010_2015   0.79485266353
medincome_change_2015_2020  -0.99014238712
medincome_change_2010_2020   .
pop_change_pct             -22.56302235567
Most predictive features:
Correlation between predicted and real values: 0.7758073
```

```
Mean Squared Error: 456.9231
```

After using the regularized trained model on our test data, we found no "most predictive" features. In practical terms, we can interpret this result as the Lasso algorithm finding that no single feature is dominant in predicting the foreclosure percentages, and the model relies on a combination of features with small contributions from each. This is consistent with the identical correlation coefficients calculated for each of our independent variables using the regularized univariate regression model.

Unfortunately, the regularized model is slightly less accurate than the initial multivariate regression model.

D. Repeat a-c multiple times with different randomly selected training and testing sets and report differences or similarities across runs.

We used the following code to repeat the initial univariate regression model process with multiple unique training and test sets:

```
# Set seed for reproducibility
set.seed(123)

# Define the number of repetitions
num_repetitions <- 5

# Loop for repetitions
for (i in 1:num_repetitions) {
  # Split data into training and testing datasets (e.g., 80% training, 20% testing)
  train_indices <- createDataPartition(pg_foreclosures_per_tract$foreclosure_pc_2020, p = 0.7, list = FALSE)
  pg_foreclosures_train <- pg_foreclosures_per_tract[train_indices, ]
  pg_foreclosures_test <- pg_foreclosures_per_tract[-train_indices, ]

  for (variable in variables_of_interest) {
    # Perform linear regression
    model <- lm(foreclosure_pc_2020 ~ ., data = pg_foreclosures_train[, c(variable, "foreclosure_pc_2020")])

    # Extract required information
    intercept <- coef(model)[1]
    coefficient <- coef(model)[2]
```

```
    # Check if the coefficient is statistically significant
    p_value <-
summary(model)$coefficients[which(rownames(summary(model)$coefficients) ==
variable), "Pr(>|t|)"]
    is_predictive <- ifelse(p_value < 0.05, "Yes", "No")

    # Compute residuals
    residuals <- resid(model)

    # Compute correlation between predicted and real values
    correlation <- cor(predict(model),
pg_foreclosures_train$foreclosure_pc_2020 )

    # Compute mean square error
    mse <- mean((predict(model) - pg_foreclosures_train$foreclosure_pc_2020
)^2)

    # Print results
    cat("Repetition:", i, "\n")
    cat("Variable:", variable, "\n")
    cat("Intercept:", intercept, "\n")
    cat("Coefficient:", coefficient, "\n")
    cat("Is it a predictive feature?:", is_predictive, "\n")
    cat("Correlation:", correlation, "\n")
    cat("Mean Square Error:", mse, "\n")
    cat("\n")
  }
}
```

And the following code to test our most successful multivariate regression model with multiple unique training and test sets:

```
# Set the number of iterations
num_iterations <- 5

# Initialize empty vectors to store results
correlations <- numeric(num_iterations)
mses <- numeric(num_iterations)

for (i in 1:num_iterations) {
  # Convert tract_number to character before splitting the data
```

```
  pg_foreclosures_per_tract$tract_number <-
as.character(pg_foreclosures_per_tract$tract_number)

  # Split the data into training and testing sets
  set.seed(i) # Use different seed for each iteration
  train_indices <-
createDataPartition(pg_foreclosures_per_tract$foreclosure_pc_2020, p = 0.8,
list = FALSE)
  train_data <- pg_foreclosures_per_tract[train_indices, ]
  test_data <- pg_foreclosures_per_tract[-train_indices, ]

  # Train the linear regression model on the training data
  lm_model <- lm(foreclosure_pc_2020 ~ ., data = train_data[,
c("foreclosure_pc_2020", test_variables_3)])

  # Use the trained model to predict on the testing data
  predicted_values <- predict(lm_model, newdata = test_data[,
test_variables_3])

  # Calculate the correlation between the predicted and real values
  correlation <- cor(predicted_values, test_data$foreclosure_pc_2020)

  # Calculate the mean squared error between the predicted and real values
  mse <- mean((predicted_values - test_data$foreclosure_pc_2020)^2)

  # Store correlation and mse
  correlations[i] <- correlation
  mses[i] <- mse
}

# Print the results of each iteration
for (i in 1:num_iterations) {
  cat("Iteration", i, "\n")
  cat("Correlation between predicted and real values:", correlations[i],
"\n")
  cat("Mean Squared Error:", mses[i], "\n\n")
}
```

And the following code to repeat the regularized univariate regression model process for each of our independent variables:

```
# Set the regularization parameter
lambda <- 0.2

# Number of iterations
num_iterations <- 5

for (iteration in 1:num_iterations) {
    # Split data into training and test sets
    set.seed(iteration)  # Set seed for reproducibility
    sample_indices <- sample(1:nrow(pg_foreclosures_per_tract), size = 0.7
* nrow(pg_foreclosures_per_tract), replace = FALSE)
    pg_foreclosures_train <- pg_foreclosures_per_tract[sample_indices, ]
    pg_foreclosures_test <- pg_foreclosures_per_tract[-sample_indices, ]

    for (variable in variables_of_interest) {
        # Prepare the data
        X_train <- model.matrix(foreclosure_pc_2020 ~ ., data =
pg_foreclosures_train[, c(variables_of_interest, "foreclosure_pc_2020")])[,
-1]
        y_train <- pg_foreclosures_train$foreclosure_pc_2020
        X_test <- model.matrix(foreclosure_pc_2020 ~ ., data =
pg_foreclosures_test[, c(variables_of_interest, "foreclosure_pc_2020")])[,
-1]
        y_test <- pg_foreclosures_test$foreclosure_pc_2020

        # Fit the Lasso regression model
        lasso_model <- glmnet(X_train, y_train, alpha = 1, lambda = lambda)

        # Extract coefficients
        coef_idx <- which(colnames(X_train) == variable)
        coefficient <- coef(lasso_model)[coef_idx]

        # Check if the coefficient is non-zero
        is_predictive <- ifelse(abs(coefficient) > 0, "Yes", "No")

        # Compute residuals
        residuals <- y_test - predict(lasso_model, newx = X_test)

        # Compute correlation between predicted and real values
        correlation <- cor(predict(lasso_model, newx = X_test), y_test)

        # Compute mean square error
        mse <- mean(residuals^2)
```

```
        # Print results
        cat("Iteration:", iteration, "\n")
        cat("Variable:", variable, "\n")
        cat("Coefficient:", coefficient, "\n")
        cat("Is it a predictive feature?:", is_predictive, "\n")
        cat("Correlation:", correlation, "\n")
        cat("Mean Square Error:", mse, "\n")
        cat("\n")
    }
}
```

And the following code to test the most regularized multivariate regression model using multiple unique training and test sets:

```
# Set the number of repetitions
num_repetitions <- 5

# Loop through repetitions
for (i in 1:num_repetitions) {
    # Split the data into training and testing sets
    set.seed(i)  # Use different seed for each iteration for
reproducibility
    train_indices <-
createDataPartition(pg_foreclosures_per_tract$foreclosure_pc_2020, p = 0.8,
list = FALSE)
    train_data <- pg_foreclosures_per_tract[train_indices, ]
    test_data <- pg_foreclosures_per_tract[-train_indices, ]

    # Train the Lasso regression model on the training data
    lasso_model <- cv.glmnet(as.matrix(train_data[, test_variables_3]),
                             train_data$foreclosure_pc_2020,
                             alpha = 1)

    # Use the trained model to predict on the testing data
    predicted_values <- predict(lasso_model, newx = as.matrix(test_data[,
test_variables_3]),
                                s = "lambda.min")

    # Calculate the correlation between the predicted and real values
    correlation <- cor(predicted_values, test_data$foreclosure_pc_2020)
```

```
    # Calculate the mean squared error between the predicted and real
values
    mse <- mean((predicted_values - test_data$foreclosure_pc_2020)^2)

    # Obtain coefficients for each feature
    coefficients <- coef(lasso_model, s = "lambda.min")

    # Identify the most predictive features
    non_zero_coefficients <- coefficients[-1]  # Exclude intercept
    top_predictive_features <-
names(non_zero_coefficients[non_zero_coefficients != 0])

    # Print the results for this run
    cat("Run:", i, "\n")
    cat("Correlation between predicted and real values:", correlation,
"\n")
    cat("Mean Squared Error:", mse, "\n")
    cat("Top predictive features:", top_predictive_features, "\n")
    cat("\n")
}
```

For the sake of space, we will only show the output from the final test, which was fairly representative of the other tests.

That code produces this output:

```
Run: 1
Correlation between predicted and real values: 0.8422236
Mean Squared Error: 385.4891
Top predictive features:

Run: 2
Correlation between predicted and real values: 0.8523196
Mean Squared Error: 384.6485
Top predictive features:

Run: 3
Correlation between predicted and real values: 0.8180499
Mean Squared Error: 384.1731
Top predictive features:

Run: 4
```

```
Correlation between predicted and real values: 0.8194143
Mean Squared Error: 340.0986
Top predictive features:

Run: 5
Correlation between predicted and real values: 0.8901317
Mean Squared Error: 293.5396
Top predictive features:
```

In other words, the predictiveness of our regularized multivariate regression model can vary substantially depending on the training and test sets used to build and test it.

The same is true for our unregularized multivariate regression model and both the unregularized and regularized univariate linear regression models, which could indicate that our relatively small sample size makes it difficult to create a model that does not vary significantly in its predictiveness depending on the sample used to train it.

Question 2. Logistic Regression and Naive Bayes:

Preparation: In order to create a logistic regression model and Naive Bayes model for our dataset, we first needed to turn our numeric response variable, foreclosure_pc_2020 – which represents the number of foreclosures in a tract between 2011-2023 divided by the most recent ACS population estimate for said tract – into an ordinal variable. We do this by dividing the range of foreclosure_pc_2020 values into quantiles, each of which becomes a class within the ordinal variable foreclosure_quantile.

A. Logistic Regression:

After that, we built a proportional odds logistic regression formula – which is more appropriate for predicting ordinal variables – to build our model.

For some reason, the exact combination of variables from our most predictive linear regression model did not run successfully when input into our logistic regression model.

For reference, here is that combination of variables:

```
test_variables_3 <- c("mortgaged_2010", "tract_homevalue_2020",
"tract_medage_2020", "tract_medincome_2020",   "foreclosure_pc_2010",
"pct_built_2000_2009", "poverty_2010", "nhwhite_2020",
"mortgage_change_2010_2020",
"poverty_change_2010_2020","nhwhite_change_2010_2020","medincome_change_201
0_2015", "medincome_change_2015_2020", "medincome_change_2010_2020",
"pop_change_pct")
```

Running that combination of variables through the logistic regression model looks like this:

```
train_data$foreclosure_quantile <- factor(train_data$foreclosure_quantile)

# Use the training dataset for model fitting
model_1 <- polr(foreclosure_quantile ~ mortgaged_2010 +
tract_homevalue_2020 + tract_medage_2020 +   tract_medincome_2020 +
foreclosure_pc_2010 + pct_built_2000_2009 + poverty_2010 + nhwhite_2020 +
mortgage_change_2010_2020 + poverty_change_2010_2020 +
nhwhite_change_2010_2020 + medincome_change_2010_2015 +
medincome_change_2015_2020 + medincome_change_2010_2020 + pop_change_pct,
data = train_data, Hess = TRUE)

# Summarize the model
summary(model_1)
```

We get this error:

```
Error in svd(X) : infinite or missing values in 'x'
```

That error generally indicates missing or infinite values, but because there are neither infinite nor missing values in our dataset, it more likely means that there are collinear variables in our model.

In our linear regression section, we set the bar for collinear variables at a correlation coefficient (between independent variables) above 0.8.

It appears that our logistic regression model is more sensitive, so we ran a new test to correlation coefficients above 0.7 using this code:

```
# Select the variables from the regression formula
selected_vars <- c("mortgaged_2010", "tract_homevalue_2020",
"tract_medage_2020",
                   "tract_medincome_2020", "foreclosure_pc_2010",
"pct_built_2000_2009",
                   "poverty_2010", "nhwhite_2020",
"mortgage_change_2010_2020",
                   "poverty_change_2010_2020", "nhwhite_change_2010_2020",
                   "medincome_change_2010_2015",
"medincome_change_2015_2020",
```

```
                    "medincome_change_2010_2020", "pop_change_pct")

# Subset the data with selected variables
selected_data <- train_data[selected_vars]

# Calculate the correlation matrix
correlation_matrix <- cor(selected_data)

# Find pairs of variables with correlation coefficient > 0.7
high_correlation <- which(upper.tri(correlation_matrix, diag = TRUE) &
correlation_matrix > 0.7, arr.ind = TRUE)

# Print the collinear variable pairs
collinear_pairs <- data.frame(variable1 =
rownames(correlation_matrix)[high_correlation[, 1]],
                              variable2 =
colnames(correlation_matrix)[high_correlation[, 2]],
                              correlation =
correlation_matrix[high_correlation])
```

The resulting correlation matrix pointed us to two potential pairs of problem variables:

mortgaged_2010	tract_medincome_2020	0.7636285
tract_homevalue_2020	tract_medincome_2020	0.7285017

Of those, mortgaged_2010 was most statistically significant (as calculated during the linear regression section), so we jettisoned the substantially less-predictive variables tract_medincome_2020 and tract_homevalue_2020 and keep only mortgaged_2010.

Model 1: That gave us the following model as a starting point:

```
# Use the training dataset for model fitting
model_1 <- polr(foreclosure_quantile ~ mortgaged_2010 + tract_medage_2020 +
foreclosure_pc_2010 + pct_built_2000_2009 + poverty_2010 + nhwhite_2020 +
mortgage_change_2010_2020 + poverty_change_2010_2020 +
nhwhite_change_2010_2020 + medincome_change_2010_2015 +
medincome_change_2015_2020 + medincome_change_2010_2020 + pop_change_pct,
data = train_data, Hess = TRUE)
```

The following are the results of the initial model, which provides both the **coefficients** for each variable and the **intercepts** for the transitions between each quantile:

```
Coefficients:
                              Value Std. Error t value
mortgaged_2010               5.1645    1.40460   3.677
tract_medage_2020            0.1112    0.04456   2.495
foreclosure_pc_2010          0.1988    0.04297   4.627
pct_built_2000_2009          5.3423    2.34005   2.283
poverty_2010                -9.0151    5.84260  -1.543
nhwhite_2020                -8.4189    2.02452  -4.158
mortgage_change_2010_2020   -9.2956    2.78059  -3.343
poverty_change_2010_2020    -5.1738    4.77354  -1.084
nhwhite_change_2010_2020     5.3517    3.08483   1.735
medincome_change_2010_2015 -11.8941    6.80579  -1.748
medincome_change_2015_2020 -10.3685    6.24906  -1.659
medincome_change_2010_2020   9.1935    5.88229   1.563
pop_change_pct              -3.0799    0.94985  -3.243

Intercepts:
    Value    Std. Error t value
1|2   4.3986   1.8510     2.3763
2|3   6.6132   1.9139     3.4554
3|4   8.6115   1.9879     4.3320
4|5  10.4258   2.0457     5.0963

Residual Deviance: 258.4844
AIC: 292.4844
```

Are the coefficients statistically significant?

We used the t-values to determine whether the coefficients were statistically significant. Based on the calculated t-values, the coefficients for foreclosure_pc_2010 (t-value = 4.627), mortgaged_2010 (t-value = 3.677), and nhwhite_2020 (t-value = -4.158) are the most statistically significant.

What are the log-odds and odd ratios of the outcome for a unit increase in each independent variable?

As far as we understand, the coefficients in logistic regression represent the change in the log-odds of the outcome variable for a one-unit change in the predictor variable. Under that interpretation, the log-odds should be equivalent to the coefficients for each variable.

Under that interpretation, the following are the **odds ratios** for a unit increase in each independent variable:

1. mortgaged_2010: 174.947
2. tract_medage_2020: 1.118
3. foreclosure_pc_2010: 1.220
4. pct_built_2000_2009: 209.996
5. poverty_2010: 0.0001216
6. nhwhite_2020: 0.0002207
7. mortgage_change_2010_2020: 0.00009183
8. poverty_change_2010_2020: 0.005663
9. nhwhite_change_2010_2020: 210.964
10. medincome_change_2010_2015: 0.000006830
11. medincome_change_2015_2020: 0.00003141
12. medincome_change_2010_2020: 9832.97
13. pop_change_pct: 0.04596

Which are the most predictive features according to the training data?

We rank the predictiveness of each variable based on both the significance of the variable (indicated by the t-value) and the effect size (indicated by the log-odds). Variables with larger absolute log-odds and t-values are considered **most predictive** in our ranking.

1. foreclosure_pc_2010 (t-value: 4.627, log-odds: 0.1988)
2. nhwhite_2020 (t-value: -4.158, log-odds: -8.4189)
3. mortgaged_2010 (t-value: 3.677, log-odds: 5.1645)
4. mortgage_change_2010_2020 (t-value: -3.343, log-odds: -9.2956)
5. pop_change_pct (t-value: -3.243, log-odds: -3.0799)
6. tract_medage_2020 (t-value: 2.495, log-odds: 0.1112)
7. pct_built_2000_2009 (t-value: 2.283, log-odds: 5.3423)
8. nhwhite_change_2010_2020 (t-value: 1.735, log-odds: 5.3517)
9. medincome_change_2010_2020 (t-value: 1.563, log-odds: 9.1935)
10. medincome_change_2015_2020 (t-value: -1.659, log-odds: -10.3685)
11. medincome_change_2010_2015 (t-value: -1.748, log-odds: -11.8941)
12. poverty_change_2010_2020 (t-value: -1.084, log-odds: -5.1738)
13. poverty_2010 (t-value: -1.543, log-odds: -9.0151)

As expected, foreclosure_pc_2010 is still the most predictive variable; unexpectedly, nhwhite_2020 is the second-most predictive.

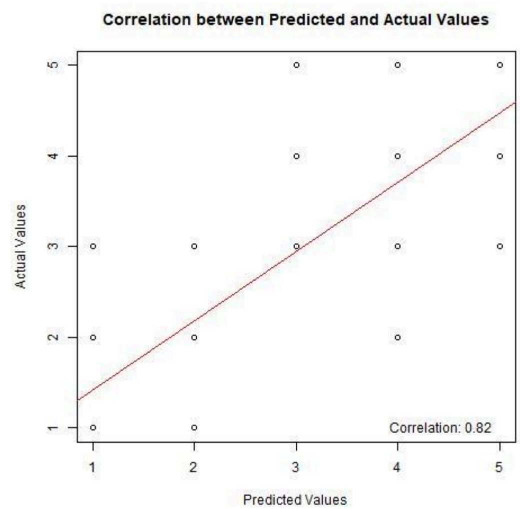
Use the trained model to predict on your testing dataset.

We used the following code to predict foreclosure_quantile values based on the values in a test set:

```
# Predict values using the model and test data
predicted_values <- predict(model_1, newdata = test_data, type = "class")

# Calculate the correlation between predicted and actual values
correlation <- cor(as.numeric(predicted_values),
test_data$foreclosure_quantile)
```

The resulting correlation coefficient between our predicted results and actual values **is 0.82.**

For the sake of thoroughness, we also conducted a one-to-one comparison of the predicted quantiles and actual quantiles to calculate an accuracy rate.



This model accurately predicted quantiles roughly 54% of the time.

That outcome was far from ideal, so **we tested other combinations of variables in search of a more predictive model.**

Model 2: In the training phase, the following model was the second-most predictive:

```
# Use the training dataset for model fitting
model_6 <- polr(foreclosure_quantile ~ foreclosure_pc_2010 +
tract_medage_2020 + poverty_2010 + nhwhite_2020 +
medincome_change_2010_2020 + avg_bed + mortgaged_2010 + pct_1_bd +
pct_built_pre_1960 + pct_built_2000_2009 + mortgage_change_2010_2015 +
pop_change_pct, data = train_data, Hess = TRUE)

# Summarize the model
summary(model_6)
```

That model produced the following output, at least when considering the residual deviance and AIC:

```
Coefficients:
                           Value Std. Error t value
foreclosure_pc_2010       0.1455    0.04298  3.3848
```

```
tract_medage_2020           0.1587    0.04780  3.3207
poverty_2010               -2.1505    4.47238 -0.4808
nhwhite_2020               -8.6907    2.00621 -4.3319
medincome_change_2010_2020  1.1254    0.94781  1.1874
avg_bed                    -1.3916    0.57201 -2.4329
mortgaged_2010              5.7754    2.57321  2.2444
pct_1_bd                   -7.0085    3.31349 -2.1151
pct_built_pre_1960         -0.8271    0.93078 -0.8886
pct_built_2000_2009         3.4087    2.50129  1.3628
mortgage_change_2010_2015   2.7022    3.11060  0.8687
pop_change_pct             -2.8016    0.95431 -2.9357

Intercepts:
    Value   Std. Error t value
1|2  2.1020  2.0460     1.0274
2|3  4.2130  2.0839     2.0217
3|4  5.9874  2.1305     2.8103
4|5  7.6469  2.1608     3.5390

Residual Deviance: 269.425
AIC: 301.425
```

Are the coefficients statistically significant?

Based on the calculated t-values, the coefficients for foreclosure_pc_2010 (t-value = 5.573), tract_medage_2020 (t-value = 3.894), nhwhite_2020 (t-value = -4.3319), and mortgaged_2010 (t-value = 3.677) are the are statistically significant.

What are the odd ratios of the outcome for a unit increase in each independent variable?

The **odds ratios** for a unit increase in each independent variable with this model are:

1. foreclosure_pc_2010: 1.1565860422
2. tract_medage_2020: 1.1720011361
3. poverty_2010: 0.1164288069
4. nhwhite_2020: 0.0001681451
5. medincome_change_2010_2020: 3.0815814210
6. avg_bed: 0.2486755107
7. mortgaged_2010: 322.2597426584
8. pct_1_bd: 0.0009041991
9. pct_built_pre_1960: 0.4373152291
10. pct_built_2000_2009: 30.2260067268

Which are the most predictive features according to the training data?

Using the same metrics for predictiveness as above, the independent variables in this model are **ranked by predictiveness** as follows:

1. foreclosure_pc_2010 (t-value: 5.573, log odds: 0.2029)
2. tract_medage_2020 (t-value: 3.894, log odds: 0.1455)
3. nhwhite_2020 (t-value: -4.3319, log odds: -8.6907)
4. mortgaged_2010 (t-value: 3.677, log odds: 5.1645)
5. avg_bed (t-value: -2.4329, log odds: -0.6755)
6. pct_1_bd (t-value: -2.1151, log odds: -7.0085)
7. medincome_change_2010_2020 (t-value: 1.1874, log odds: 0.6656)
8. pct_built_pre_1960 (t-value: -0.8886, log odds: -1.0883)
9. pct_built_2000_2009 (t-value: 1.3628, log odds: 1.7932)
10. mortgage_change_2010_2015 (t-value: 0.8687, log odds: 2.7022)
11. pop_change_pct (t-value: -2.9357, log odds: -2.8016)
12. poverty_2010 (t-value: -1.926, log odds: -6.7241)

In this case, foreclosure_pc_2010 remains the most predictive variable, and tract_medage_2020 – which was among the most predictive during the linear regression section – is the second-most predictive variable.

Use the trained model to predict on your testing dataset.

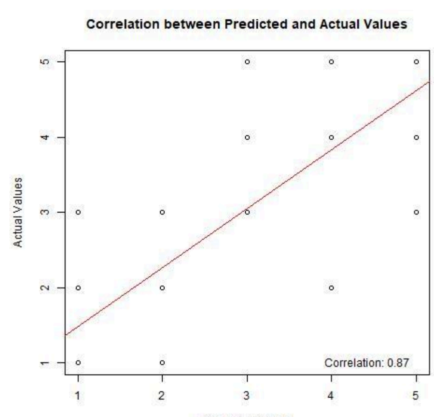
We used the following code to predict foreclosure_quantile values based on the values in a test set:

```
# Predict values using the model and test data
predicted_values <- predict(model_6, newdata = test_data, type = "class")

# Calculate the correlation between predicted and actual values
correlation <- cor(as.numeric(predicted_values),
test_data$foreclosure_quantile)
```

For this model, the resulting correlation coefficient between our predicted results and actual values **is 0.87.**

This model predicts the exact quantile correctly roughly 60.7% of the time.

Why this model is more accurate than our initial model is unclear.



B. Naive Bayes:

Divide your dataset into training and testing set and train the classifier. Report the confusion matrix.

After dividing our dataset into training and testing sets, we used the following code to train the classifier:

```
# Fit the Naive Bayes model
naive_bayes_model <- naiveBayes(foreclosure_quantile ~ nhwhite_2020 +
                                  foreclosure_pc_2010 +
                                  tract_medage_2020 +
                                  poverty_2010 +
                                  medincome_change_2010_2020 +
                                  avg_bed +
                                  mortgaged_2010 +
                                  pct_1_bd +
                                  pct_built_pre_1960 +
                                  pct_built_2000_2009 +
                                  mortgage_change_2010_2015 +
                                  pop_change_pct, data =
pg_foreclosures_per_tract_log_reg)
```

And we used the following code to test the classifier:

```
# Predict using the Naive Bayes model
predictions <- predict(naive_bayes_model, newdata = test_data_2)

# Create the confusion matrix
confusion_matrix <- confusionMatrix(predictions,
factor(test_data_2$foreclosure_quantile, levels = 1:5))
```

That produces the following confusion matrix:

```
Confusion Matrix and Statistics

          Reference
Prediction 1 2 3 4 5
         1 6 2 1 0 0
         2 3 2 0 0 1
         3 0 1 4 1 2
         4 0 0 1 0 3
```

```
        5 0 1 1 2 4

Overall Statistics

               Accuracy : 0.4571
                 95% CI : (0.2883, 0.6335)
    No Information Rate : 0.2857
    P-Value [Acc > NIR] : 0.02302

                  Kappa : 0.3073

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity            0.6667  0.33333   0.5714  0.00000   0.4000
Specificity            0.8846  0.86207   0.8571  0.87500   0.8400
Pos Pred Value         0.6667  0.33333   0.5000  0.00000   0.5000
Neg Pred Value         0.8846  0.86207   0.8889  0.90323   0.7778
Prevalence             0.2571  0.17143   0.2000  0.08571   0.2857
Detection Rate         0.1714  0.05714   0.1143  0.00000   0.1143
Detection Prevalence   0.2571  0.17143   0.2286  0.11429   0.2286
Balanced Accuracy      0.7756  0.59770   0.7143  0.43750   0.6200
```

According to the confusion matrix, the classifier has an overall accuracy of 45.71%. That is not a strong model, but it is significantly better than the No Information Rate (28.57%). However, the classifier's performance varies across quantiles, with higher accuracy for quantile 1, 3, and 5, less accuracy for quantile 2, and no accuracy for quantile 4.

Repeat the process above with the Laplace estimator. Do the results improve?

We retrained the Naive Bayes classifier using a LaPlace estimator with the following code:

```
# Fit the Naive Bayes model with Laplace smoothing
naive_bayes_model_laplace <- naiveBayes(foreclosure_quantile ~ nhwhite_2020
+
                                               foreclosure_pc_2010 +
                                               tract_medage_2020 +
                                               poverty_2010 +
                                               medincome_change_2010_2020 +
                                               avg_bed +
                                               mortgaged_2010 +
                                               pct_1_bd +
```

```
                                                    pct_built_pre_1960 +
                                                    pct_built_2000_2009 +
                                                    mortgage_change_2010_2015 +
                                                    pop_change_pct,
                                               data =
pg_foreclosures_per_tract_log_reg,
                                               laplace = 1)
```

And we used the following code to test the classifier:

```
# Predict using the Naive Bayes model
predictions <- predict(naive_bayes_model_laplace, newdata = test_data_2)

# Create the confusion matrix
confusion_matrix <- confusionMatrix(predictions,
factor(test_data_2$foreclosure_quantile, levels = 1:5))
```

That produces the following confusion matrix:

```
Confusion Matrix and Statistics

          Reference
Prediction 1 2 3 4 5
         1 6 2 1 0 0
         2 3 2 0 0 1
         3 0 1 4 1 2
         4 0 0 1 0 3
         5 0 1 1 2 4

Overall Statistics

               Accuracy : 0.4571
                 95% CI : (0.2883, 0.6335)
    No Information Rate : 0.2857
    P-Value [Acc > NIR] : 0.02302

                  Kappa : 0.3073

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity            0.6667  0.33333   0.5714  0.00000   0.4000
```

```
Specificity            0.8846  0.86207   0.8571  0.87500   0.8400
Pos Pred Value         0.6667  0.33333   0.5000  0.00000   0.5000
Neg Pred Value         0.8846  0.86207   0.8889  0.90323   0.7778
Prevalence             0.2571  0.17143   0.2000  0.08571   0.2857
Detection Rate         0.1714  0.05714   0.1143  0.00000   0.1143
Detection Prevalence   0.2571  0.17143   0.2286  0.11429   0.2286
Balanced Accuracy      0.7756  0.59770   0.7143  0.43750   0.6200
```

Given that all of the categories (quantiles) appear in our training and test data sets, it is unsurprising that the Laplace estimator, which is meant to ensure that categories that do not appear in the training data are still assigned a non-zero probability, does not have an impact on the accuracy of our Naive Bayes model.

Question 3: Decision Trees and Random Forests

To conduct this form of analysis, we opted to divide our column of foreclosure values per capita in 2020 by three foreclosure risk levels—low, medium (moderate), and high. Our range in values prior to dividing the data was as follows: Min = 8.121113, 25th% = 49.55049, 75th% = 94.04721, and Max = 149.7634. In the screenshot below, we organized this range to fit our various risk levels.

```
pg_foreclosures_per_tract <-
pg_foreclosures_per_tract |>
  mutate(ranges=
  if_else(
    foreclosure_pc_2020 >= 0 & foreclosure_pc_2020 < 49.55049,
    "low",
    if_else(
      foreclosure_pc_2020 >= 49.55049 & foreclosure_pc_2020 < 94.04721,
      "medium",
      "high"
      )
    )
  )
```

From there, we saved a new version of our original dataset called “pg_fc_pt” (Prince George’s Foreclosures Per Tract), which removed the “foreclosures per capita in 2020” column to better test the viability of our other variables in the dataset. After completing that step, we set a seed that saved the randomized version of our data. We called this randomized dataset "random_fc2020" and pulled the 173 rows from our cleaned dataset.

A. Testing and Training Datasets

We divided the randomized dataset into testing and training datasets at a 90/10 split of rows. The training dataset contained rows 1 through 156, while the testing dataset contained the remaining rows. These datasets were then entered into proportion tables that enabled us to view their outcome variable distributions based on our previously established risk levels. The training dataset returned the following distribution: high = 30%, low = 30%, and medium = ~40%. The testing dataset, in comparison, offered a slightly different distribution: high = 11%, low = 35%, and medium = ~53%.

B. Decision Tree Training and Confusion Matrix

The screenshots below show the results from our initial training on the decision tree and corresponding confusion matrix. The actual values equal the predicted values. This suggests a 100% accuracy rate, but the low sample size also heavily influences this outcome. This also shows that the distribution after the split is similar to the original.

```
Class specified by attribute `outcome'

Read 156 cases (2 attributes) from undefined.data

Decision tree:

ranges = high: high (47)
ranges = low: low (47)
ranges = medium: medium (62)


Evaluation on training data (156 cases):

          Decision Tree
         ----------------
         Size      Errors

            3    0( 0.0%)   <<


         (a)   (b)   (c)    <-classified as
         ----  ----  ----
          47                (a): class high
                47          (b): class low
                      62    (c): class medium


        Attribute usage:

        100.00% ranges


Time: 0.0 secs
```

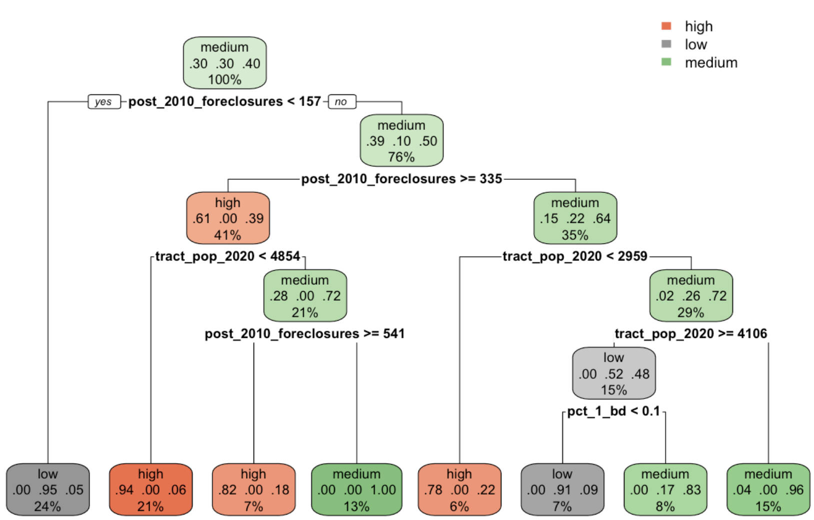
```
   Cell Contents
|-------------------------|
|                       N |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  17


             | predicted
      actual |      high |       low |    medium | Row Total |
-------------|-----------|-----------|-----------|-----------|
        high |         2 |         0 |         0 |         2 |
             |     0.118 |     0.000 |     0.000 |           |
-------------|-----------|-----------|-----------|-----------|
         low |         0 |         6 |         0 |         6 |
             |     0.000 |     0.353 |     0.000 |           |
-------------|-----------|-----------|-----------|-----------|
      medium |         0 |         0 |         9 |         9 |
             |     0.000 |     0.000 |     0.529 |           |
-------------|-----------|-----------|-----------|-----------|
Column Total |         2 |         6 |         9 |        17 |
-------------|-----------|-----------|-----------|-----------|
```

The subsequent screenshot is a plotted example of our decision tree, visualizing how the model determines outcomes using our low, medium, and high structure. We initially expected variables like median income to be our ideal predictors, as opposed to the variables we ultimately observed in the tree. We were surprised to see that omission considering the variable's relevance in other forms of predictive analysis. We didn't necessarily expect post-2010 foreclosures as a predictive variable. However, this makes some sense in retrospect because it is likely that tracts that previously showed lower numbers of foreclosures might be more likely to have a lower foreclosure rate in the future compared to other tracks with a higher number of foreclosures. Tract populations in 2020 were another interesting variable to us. The tree indicates that higher population counts in a given tract led to higher foreclosure potential. This may be explained by the fact that a highly populated area presents more opportunities for foreclosures as opposed to a less populated area. The other unexpected variable was the percentage of 1 bedroom units that were less than 10%. This resulted in a lower probability of foreclosing, which could have been the case because residents of one-bedroom properties might have relatively lower monthly payments and living expenses compared to families or residents of properties with a higher number of bedrooms. However, we would need to examine this further with more data.

Our tree begins by looking at post-2010 foreclosure values per tract in our dataset that are less than 157. If "yes," the tree predicted a low chance of foreclosure, with 24% of the dataset's values. If "no," it checked post-2010 foreclosures greater than or equal to 335. "Yes" values to that variable predicted a high chance of future foreclosures, and it branched off to check for 2020 tract populations of less than 4,854. A "yes" to that variable indicates a high chance of foreclosure, and 21% of our values followed this path. A "no" to 2020 tract populations less than 4,854 led to examining post-2010 foreclosures greater than or equal to 541. "Yes" suggested a high chance of foreclosure, with 7% of our values falling in this range, while "no" predicted a medium chance of foreclosure with 13% of our values.

Going back to post-2010 foreclosures greater than or equal to 335, a "no" examined 2020 tract populations of less than 2,959. A "yes" predicted a high chance of foreclosure and showed 6% of our values in this outcome, while a "no" led to looking at 2020 tract populations greater than or equal to 4,106. "Yes" outcomes led to a low chance of foreclosure (15%) and split off again based on the percentage of 1-bedroom units that were less than 10%. For this split, we saw 7% of our "yes" values with a low chance to foreclose, while "no" values (8%) indicated a medium chance to foreclose. Finally, "no" answers to 2020 tract populations greater than or equal to 4,106 (15%) indicated a medium chance to foreclose.

In summary, we observed a relatively even distribution of outcomes in the decision tree. Roughly 31% fell under low foreclosure probability outcomes, while 36% and 34% fell under medium and high foreclosure probability outcomes, respectively. The most significant low probability indicator came from tracts with post-2010 foreclosure values less than 157 (24%), the most significant medium outcomes came from 2020 tract populations greater than or equal to 4,106 (15%), and the most significant high foreclosure probability outcome came from 2020 tract populations less than 4,854 (21%).



C. Boosting

The results shown in the screenshot below indicate that boosting was attempted with three trials. However, the process was abandoned after the first trial because of the high level of accuracy achieved due to the low number of classifiers. This highlighted a limitation of our data, given that more data is needed in order to fully complete the boosting process and draw valuable conclusions from this step.

```
Call:
C5.0.default(x = training[46], y = training$ranges, trials = 3)


C5.0 [Release 2.07 GPL Edition]        Sat Apr  6 08:36:34 2024
-------------------------------

Class specified by attribute `outcome'

Read 156 cases (2 attributes) from undefined.data

-----  Trial 0:  -----

Decision tree:

ranges = high: high (47)
ranges = low: low (47)
ranges = medium: medium (62)

*** boosting reduced to 1 trial since last classifier is very accurate

*** boosting abandoned (too few classifiers)


Evaluation on training data (156 cases):

            Decision Tree
          ----------------
          Size      Errors

             3    0( 0.0%)   <<


           (a)   (b)   (c)    <-classified as
          ----  ----  ----
            47                (a): class high
                  47          (b): class low
                        62    (c): class medium


        Attribute usage:

        100.00% ranges
```

D. Bagging and Random Forests

Like in previous steps, we set the seed and randomized the dataset. Then, we conducted the bagging analysis, which added bootstrap replications. We ran 50 bootstrap replications as it is a high number that should yield adequate results. After running the code, the misclassification error was 22.5%, which indicates our accuracy was 77.5%. This suggests that random forests with bagging help raise our level of accuracy. However, this analysis would also greatly benefit from more data to run through these models.

```
Bagging classification trees with 50 bootstrap replications

Call: bagging.data.frame(formula = ranges ~ ., data = bagfc2020, nbagg = 50,
    coob = TRUE, control = rpart.control(minsplit = 2, cp = 0,
        min_depth = 2))

Out-of-bag estimate of misclassification error:  0.2254
```

As it pertains to our random forest, we first trained the model with four different numbers of trees. Then, we set a prediction outcome after running the test data through the random forest. Our range of accuracy shown in the screenshot below suggests that the model has moderate predictive performance. However, this finding does not necessarily suggest the format is sufficient for predictive analysis.

In this section, we also calculated importance scores for our dataset variables. Post-2010 foreclosures, mortgaged 2020, owner-occupied properties in 2015, owner-occupied properties in 2020, and the percentage of properties built pre-1960 in each tract had relatively higher importance across all trees in the random forest.

Overall, we would most likely want a different model to help us predict foreclosures per capita.

```
# training
random_forest = randomForest(as.factor(train_data$ranges)~.,data = train_data,ntree = 4,importance = T)

# predictions about the test data
predictions <- predict(random_forest, test_data)

importance_scores <- importance(random_forest)
# we used this function to ascertain the model's accuracy.
accuracy <- sum(predictions == test_data$ranges) / nrow(test_data)
print(paste("Accuracy:", round(accuracy, 2)))
print(importance_scores)

```

[1] "Accuracy: 0.69"
                                   high   low   medium MeanDecreaseAccuracy MeanDecreaseGini
avg_bed   0.00000000 0.000000 0.0000000   0.00000000   0.0000000
tract_number   0.00000000 0.000000 1.9985407   1.99973582   0.9630376
tract_homevalue_2020   -0.06730536 0.000000 -1.1547005   -0.45883147   3.0162659
tract_medage_2020   1.15470054 1.756917 1.1547005   1.56414491   3.3428104
tract_medincome_2020   0.00000000 0.000000 0.0000000   0.00000000   0.0000000
tract_medincome_2010   0.00000000 0.000000 0.0000000   0.00000000   0.6469697
tract_pop_2010   0.00000000 1.154701 -1.1547005   1.15470054   2.7215097
tract_pop_2020   1.15470054 0.000000 0.0000000   1.15470054   1.9495042
great_recession_foreclosures 1.15470054 1.946657 1.1547005   1.67977499   2.9313724
post_2010_foreclosures   5.37471543 1.644573 0.7821117   3.23700172   17.9559473
```
```

Question 4: Comparative Analysis. (Jasper)

[5 pts] Write a summary of all classifiers, their predictive quality and which one would you use to answer your research question(s).

Please add a Contributions section at the end of your report clearly stating the contributions of each student to the report, coding and presentation preparation and recording (Youtube video). For example, in a team with three members VFM, LM and AM, you could write "Question 1a: developed by VFM and LM; Question 1b: developed by VFM; Question 1c: developed by VFM and AM; Question 1d: developed by All; Question 1e: developed by VFM. VFM prepared 70% of the code; LM worked on 20% of the code and AM on 10%. All students contributed equally to the preparation and recording of the presentation".

Team efforts only work if all team members distribute the workload equitably. If a team member's contributions are much lower than the others' I will prorate the final grade accordingly.

Q4: Comparative analysis

Team 2's overarching research question:

Is it possible to forecast the per capita rate of foreclosure for a census tract in Prince George's County, Maryland, using identified independent variables?

Foreclosure is a cross cutting issue that bears negative implications within the financial system writ large, but also affects populations at the community and family level. According to a study conducted by the Urban Institute in Washington DC, the process and experience of foreclosure has generational ripple effects that may stem from legal exposure during the foreclosure process, displacement and housing instability, prolonged financial insecurity, personal/familial stress and mental hardship.

For these reasons, Team 2 has approached it's comparative analysis through two lenses:

1. Which classifier is most equipped to answer our primary research question from the standpoint of statistical accuracy

2. Which classifier bears the ability to supplement decision making capabilities of a decision maker in the public or private sector concerned with mitigating the effects of foreclosure in an area of interest?

Due to the extensive ecological, economical, and demographic impacts of foreclosure, Team 2 believes public and private sector decision makers have vested interest in working to identify areas within their respective locales, states, or countries that are most at risk for foreclosure as to formulate mitigative strategies geared to cater communities classified at high risk. Team 2, toward answering our identified research question, constructed several classifiers that could be used to predict per capita foreclosure rates as well as predict tracts most at risk for foreclosure.

Classifiers and Result Summaries:

Linear Regressions

Team 2's linear regressions model leveraged the following variables determined to be most predictive amongst our available after regularization data:

1. pct_built_2000_2009 with a coefficient of 64.77035.
2. pct_built_2010_2019 with a coefficient of 55.26635.
3. avg_bed with a coefficient of 51.21755.
4. pct_built_1990_1999 with a coefficient of 29.19928.
5. pct_built_1980_1989 with a coefficient of 12.14104.
6. ownoccupied_2015 with a coefficient of 78.21088.
7. pct_1_bd with a coefficient of -103.1094.
8. pct_built_2020_later with a coefficient of 1.904033.
9. mortgage_change_2015_2020 with a coefficient of 32.56724.
10. ownoccupied_change_2010_2015 with a coefficient of -47.27385.

Multivariate Linear regressions

The most predictive features of Team 2's Multivariate Linear regressions were medincome_change_2010_2015, medincome_change_2015_2020, poverty_2010, medincome_change_2010_2020, and nhwhite_2020. This model had a correlation value of 79% between predicted and real values, and had a mean squared error of 436.93l.

Logistic Regression and Naïve Bayes

Logistic Regression:

Team 2's logistic regression model leveraged 13 variables that were determined to be the most predictive amongst our existing variable set. These variables are noted in detail within the Logistic Regression section above in this Milestone 2 report.

Of these 13 variables, nhwhite_2020, the percentage of residents in a tract claiming non-Hispanic White ancestry, as well as foreclosure_pc_2010 were most predictive. After executing this model against our test dataset, we found that the model accurately predicted per capita quantile foreclosure rates at the tract level 60% of the time, measured against quantiles of our dependent variable foreclosure_pc_2020.

Naïve Bayes:

Team 2 trained its Naïve Bayes model using the same set of selected variables used to train its logistic regression model described in this report. Team 2's Naïve Bayes classifier was accurate 45% of the time according to the confusion matrix produced by the model, though it was significantly more accurate than the no-information prediction.

Decision Trees

Team 2, in constructing its decision tree model, established low, medium and high ranges for prioritized variables, which were post-2010 foreclosures, 2020 tract population and percentage of 1 bedroom units. This breakdown essentially allowed us to classify census tracts within Prince Georges County MD, based on foreclosure risk level (low, medium, or high) based on our selected attributes within our decision tree.

Team 2’s decision tree model ultimately reported 31% of Prince George's County census tracts fell under low foreclosure risk, 36% fell under medium risk, and 34% fell under high risk, based on our identified variables of interest. This decision tree model reported its results with 100% accuracy; however, the small size of the tract-level data sample used by Team 2 likely influenced this accuracy level. Team 2 would expect potentially richer insights from block-level data, which would allow decision makers to administer mitigative efforts to target areas with higher accuracy.

Random Forest:

Team 2’s Random Forest model was 77.5% accurate, with a misclassification error of 22.5% after executing bagging models with 50 bootstrap replications. This model was trained using four separate sets of trees for training.

Outcome

Team 2’s most accurate classifiers were its decision tree and its logistic regression classifiers. Based on these results we would likely want to use a logistic regression model for the specific purpose of predicting per capita foreclosures based on our most predictive variables. Such a classifier has the potential to answer our research question of whether we can predict per capita foreclosures at the census tract level in Prince George's County. Adding additional, granular data from the block level of our Prince George's county census tracts could potentially improve the accuracy and utility of our overall model. Team 2 aims to explore this in future milestones of our project.

To inform a decision maker, we would likely opt to use a combination of methods. Team 2 believes the decision tree classifier it has constructed could be used effectively to classify census tracts based on their foreclosure risk. In conjunction, the logistic regression model could be used to forecast foreclosure rates. Such a combination of data streams would effectively enable a decision maker to identify areas of risk/concern, with a forecast to what unmitigated outcomes could look like. Based on Team 2’s research question and working data, this combination of products would create a machine learning pipeline that would be the most useful to a decision maker.

Potential Drawbacks

Decision trees have a high degree of interpretability as they visually represent decision making and are relatively intuitive to understand and visualize. For decision makers and those responsible for briefing and or influencing decision makers, these attributes are extremely important. Generally, when it comes to the accuracy of decision trees, the models are prone to overfitting, which typically occurs when there are too few data points. A decision tree with a model with higher utility may need to incorporate more data points to overcome this. Logistic regressions additionally require sufficient sample sizes for reliable results to occur. If this condition is not met, logistic regression models additionally are likely to overfit predictions.

Contributions:

Question 1: Krehl Kasayan and Paul Kiefer

Question 2: Paul Kiefer

Question 3: Pablo Suarez

Question 4: Jasper Evans

Report: Equal input by all four members.

Video Presentation: Video editing by Pablo Suarez