final

Name: Brian Deng

```
library(bis557)
library(cluster) # k-means clustering
library(leaps)   # best subset selection
library(splines) # splines and GAM
```

Name: Brian Deng (BIS557 FINAL - Extended Abstract)

Topic: Boston 2020 Property Assessment

Introduction to Dataset

In this *final* project's abstract, the aim is to investigate **Boston's property values and property taxes** on **residential** real estate properties for the year 2020. Boston is the primary city of New England, where many prestigious universities and economic sectors (e.g. health care, life sciences, education, tech, research, etc.) attract many people worldwide, which has a huge effect on the real estate market and its asset values. Therefore, I **investigate** the analytics of Boston's high demand for residential real estate below.

This CSV data was retrieved from **Analyze Boston**, which is the main source for Boston's open-source data. The websites for Boston's Property Assessment are below:

> https://data.boston.gov/dataset/property-assessment.
>
> https://data.boston.gov/dataset/property-assessment/resource/8de4e3a0-c1d2-47cb-8202-98b9cbe3bd04.

The dataset is loaded in this {bis557} package, and is named `data(final_data_bos)`.

Although this dataset also contains commercial real estate, I will focus exclusively on **residential** real estate (excluding apartments and condominiums). Several columns that are interesting to analyze are shown below:

- `ZIPCODE`: zipcode of the property (useful for geographic analysis)
- `AV_TOTAL`: the total property value for 2020 (building + land)
- `GROSS_TAX`: the property tax amount for 2020
- `LAND_SF`: the land (lot) area (in square feet)
- `YR_BUILT`: the year the property was built
- `LIVING_AREA`: the size of the inside living area (in square feet)

Of course, the analysis will not be limited to these predictors.

After data cleaning, the analysis will use a combination of computational ML methods with *specialized* user-input penalty weights. Analyzing this dataset to make ML predictions requires some **domain knowledge** to succeed in designing the loss and optimization function.

Higher-level **diagnostic analytics** will be used to infer "*why*" and see "*how*" this data provides certain relationships. This includes **nonlinear modeling**, with *validation sets* and *best-subset selection*, to see how

several predictors can lead to the target variables of `AV_TOTAL` and `GROSS_TAX`. Note that `GROSS_TAX` is the result of multiplying the values of `AV_TOTAL` by a scalar, the property tax rate.

The nonlinear models can provide a good intuition to predict property values, per sq ft, if one wants to sell Boston residential properties. In the data cleaning process, I included a new variable `AV_TOTAL_SQFT`, the total value per sq ft, which is `AV_TOTAL` divided by `LIVING_AREA`. Of course, trying to assess real estate properties is difficult, given the relative inefficiency and illiquidity compared to stocks/equities, so even actual valuations from the original dataset might not represent the true value of the property (since prices are only determined during a real estate transaction).

Besides, these models can be **assessed** on the basis of using validation and test sets. A good idea to assess models is to **find** a property currently on sale on Zillow and see how accurate and reliable (of course, managing the bias-variance tradeoff) the models are.

Since property characteristics, and hence the property values, **depend heavily on geographic factors** (e.g. specific neighborhoods, distance from Downtown Boston), **multivariate** statistical methods, such as clustering, will be used to see the extent of the zipcodes' similarities and differences. This is where **domain knowledge** comes in (e.g. real estate, Boston's demographics and geography, labor market and prevalent sectors, etc.). Clustering zipcodes with similar real estate and economic characteristics can allow models to be slightly more accurate, which can provide **novelty** to models.

Data Cleaning

Before doing advanced data analysis, data cleaning must be performed to make sure that only **residential** properties are analyzed. The function used is named `bis557::final_clean()`:

```
df <- final_clean()
```

The function's script provides the *details* of data cleaning (specifically rows and columns to delete and values to modify). The script includes creating the new variable `AV_TOTAL_SQFT`. Here, the data table `df` is the cleaned version of the original dataset.

Domain Knowledge: Clustering Zipcodes

One of the main hypotheses is that many properties are **similar** in related zipcodes. In other words, some zipcodes are more similar than others due to **geographic reasons**. The domain knowledge of human geography, demography, and local economic trends will help in analyzing properties in a specific neighborhood. Specifically, distance from downtown, distance from areas with excellent job markets in lucrative sectors, distance from the beaches, and distance from excellent school districts will largely determine the property value. This domain knowledge must be incorporated to ML models to update the loss functions, and hopefully increase accuracy.

With 30 unique zipcodes, I plan to group these zipcodes into **4 clusters**. Each cluster should represent a group of zipcodes that are *similar* in terms of property values, property size, property age, and other property characteristics. The method used is k-means clustering, using the Euclidean distance metric with Ward's linkage. The clustering plot with $k = 4$ clusters is shown below. The function used is `bis557::final_kclust()`.

```
# Clustering Process
zip_cuts <- final_kclust(df, k = 4)
cut_num <- zip_cuts$x_cuts
# Cluster Plot (axis of first 2 principal components) - {cluster} library
clusplot(zip_cuts$x_zip[,-1], clus = cut_num, lines = 0, shade = FALSE,
         color = TRUE, labels = 2, cex = 0.5, xlab = "PC #1", ylab = "PC #2",
         main = "Boston Zipcodes: K-Means \n Euclidean Distance & Ward's Linkage")
```

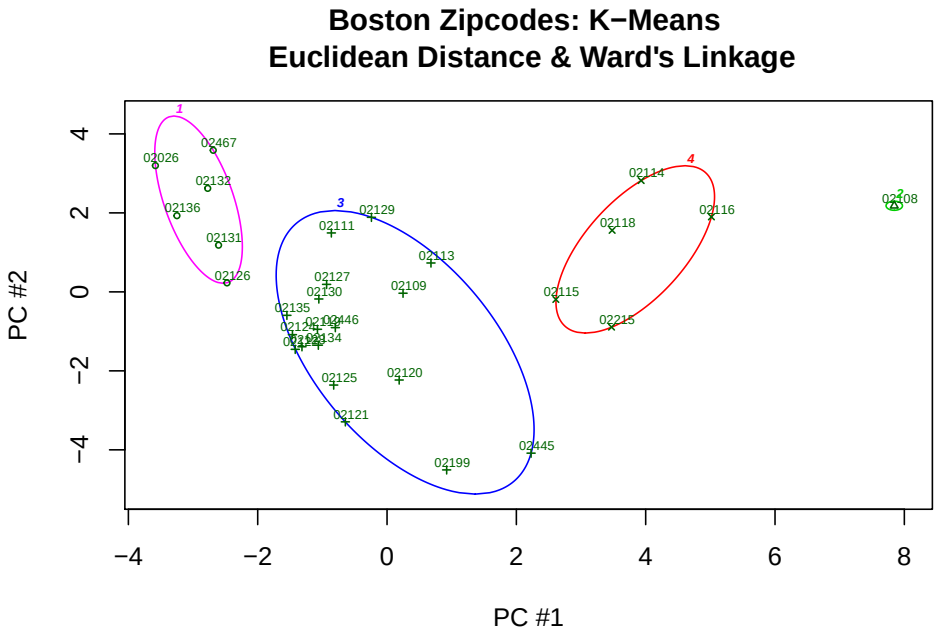

Figure 1: Zipcode Clusters

```
print(cut_num)
#> 02026 02108 02109 02111 02113 02114 02115 02116 02118 02119 02120 02121 02122
#>     1     2     3     3     3     4     4     4     4     3     3     3     3
#> 02124 02125 02126 02127 02128 02129 02130 02131 02132 02134 02135 02136 02199
#>     3     3     1     3     3     3     3     1     1     3     3     1     3
#> 02215 02445 02446 02467
#>     4     3     3     1
```

Here, we see that **cluster 1** (02026, 02126, 02131, 02132, 02136, and 02467) are Boston's southwestern suburban neighborhoods, far from Downtown and close to suburban cities like Dedham, indicating cheaper residential homes.

Here, we see that **cluster 2** (02114, 02115, 02116, 02118, and 02215) are in Downtown Boston (Beacon Hill, Back Bay, etc), with the most expensive real estate and large number of rooms with several floors.

Zipcodes in **cluster 3** lie between Downtown Boston and the most suburban, outer neighborhoods in Boston (some zipcodes can be inner-city Boston neighborhoods with higher crime levels).

The zipcode 02108 has its own cluster, **cluster 4**, which is also one of the zipcodes of Downtown Boston. This zipcode contains the Boston Common, parts of Beacon Hill, and many government-owned buildings right next to the MA State House.

It looks like cluster 2 and cluster 4 are similar, but one of the clusters includes many government buildings in the most centrally located places in Downtown Boston.

These categorizations should be included in models shown below, to show the **geographical component of property valuations**.

After categorizing zipcodes to 4 clusters, the mean property value for each cluster is shown below.

```
#> [1] "Average Property Value for Each Cluster: "
#> ZIP_GROUP1 ZIP_GROUP2 ZIP_GROUP3 ZIP_GROUP4
#>   509317.0  5015543.7   714501.9  2966457.9
```

```
#> [1] "Average Property Value per Square Feet for Each Cluster: "
#> ZIP_GROUP1 ZIP_GROUP2 ZIP_GROUP3 ZIP_GROUP4
#>   283.6370  1213.4245   287.9633   884.8090
```

Obviously, neighborhoods in Cluster 2 and 4 (downtown area) are the most expensive, while the outer suburban regions of Boston (Cluster 1) are much cheaper. This applies to both the total price and total price per square footage. In the following section, several methods will be used to more carefully model and predict Boston's property values.

Domain Knowledge: Minimum Distance to Downtown

For each zipcode, including the *straight distance* "as the crow flies" to either Downtown Boston, Back Bay, or Cambridge (choose the minimum distance) will influence the property values, since those three places contain the Boston Metro Area's most expensive real estate and the most lucrative economic sectors. The data will be obtained in Google Maps, using its ruler to measure distance (and will be recorded in 2-mile intervals), with the dataset named as `data(bos_zip_dist)`.

Polynomial Modelling

First, polynomial models should be fit to this data. The exponents (cubic, quartic, etc) would be a deciding factor after considering the loss function with my user inputs. The target variable will be `AV_TOTAL_SQFT`, since adjusting for size and living area gives the best indicator of the property's value. Each model will become more and more complex (with higher exponents). The modelling will be for the training data (70% of observations).

(**Note**: When modeling on `ZIP_GROUP`, the treatment contrast sets the intercept for Cluster 1.)

```
# Set the indices for the training set for all 6 polynomial models
set.seed(2020)
n_train <- round(0.7 * nrow(df))
idx <- matrix(nrow = 6, ncol = n_train)
for (k in 1:6) {
  idx[k,] <- sample(nrow(df), size = n_train)
}

# Adjust exponents for each linear model
fit1 <- lm(AV_TOTAL_SQFT ~ poly(LAND_SF,1) + YR_BUILT + poly(GROSS_AREA,1)
           + poly(LIVING_AREA,1) + poly(NUM_FLOORS,1) + poly(R_BDRMS,1)
           + poly(R_FULL_BTH,1) + poly(R_KITCH,1) + R_FPLACE + YR_LAST_UPGRA
           + ZIP_GROUP + poly(DIST_FROM_DT,2), data = df[idx[1,],])

fit2 <- lm(AV_TOTAL_SQFT ~ poly(LAND_SF,2) + YR_BUILT + poly(GROSS_AREA,2)
           + poly(LIVING_AREA,2) + poly(NUM_FLOORS,1) + poly(R_BDRMS,1)
           + poly(R_FULL_BTH,1) + poly(R_KITCH,1) + R_FPLACE + YR_LAST_UPGRA
           + ZIP_GROUP + poly(DIST_FROM_DT,2), data = df[idx[2,],])

fit3 <- lm(AV_TOTAL_SQFT ~ poly(LAND_SF,2) + YR_BUILT + poly(GROSS_AREA,2)
           + poly(LIVING_AREA,2) + poly(NUM_FLOORS,2) + poly(R_BDRMS,2)
           + poly(R_FULL_BTH,2) + poly(R_KITCH,2) + R_FPLACE + YR_LAST_UPGRA
           + ZIP_GROUP + poly(DIST_FROM_DT,2), data = df[idx[3,],])

fit4 <- lm(AV_TOTAL_SQFT ~ poly(LAND_SF,3) + YR_BUILT + poly(GROSS_AREA,3)
           + poly(LIVING_AREA,3) + poly(NUM_FLOORS,2) + poly(R_BDRMS,2)
           + poly(R_FULL_BTH,2) + poly(R_KITCH,2) + R_FPLACE + YR_LAST_UPGRA
```

```
           + ZIP_GROUP + poly(DIST_FROM_DT,2), data = df[idx[4,],])

fit5 <- lm(AV_TOTAL_SQFT ~ poly(LAND_SF,4) + YR_BUILT + poly(GROSS_AREA,4)
           + poly(LIVING_AREA,4) + poly(NUM_FLOORS,3) + poly(R_BDRMS,2)
           + poly(R_FULL_BTH,2) + poly(R_KITCH,2) + R_FPLACE + YR_LAST_UPGRA
           + ZIP_GROUP + poly(DIST_FROM_DT,2), data = df[idx[5,],])

fit6 <- lm(AV_TOTAL_SQFT ~ poly(LAND_SF,4) + poly(YR_BUILT,3) + poly(GROSS_AREA,4)
           + poly(LIVING_AREA,4) + poly(NUM_FLOORS,3) + poly(R_BDRMS,2)
           + poly(R_FULL_BTH,2) + poly(R_KITCH,2) + poly(R_FPLACE,2)
           + poly(YR_LAST_UPGRA,3) + ZIP_GROUP + poly(DIST_FROM_DT,2),
           data = df[idx[6,],])
```

The predictions for the test set (30% of observations) will be made for each of the 6 polynomial models.

Assessment and Loss Function

For **novelty** purposes, the loss function should be *designed* so that properties in Cluster 2 and 4 (downtown area) would be *penalized heavier* than those in other clusters. This is because many properties in Downtown are extremely expensive and thus skews the data and the model. Therefore, the loss function must be *weighted*, to give a heavier weight to Clusters 2 and 4 and a lighter weight to Cluster 1.

The weighted loss function for the test set is designed as follows, where $\mathbf{y}$ is the actual vector and $\hat{\mathbf{y}}$ is the predicted vector:

$$L = \frac{1}{n_{\text{test}}}(\mathbf{y} - \hat{\mathbf{y}})^T W(\mathbf{y} - \hat{\mathbf{y}}),$$

where $W = \text{diag}(w_1, ..., w_n)$ is a diagonal matrix such that w_i is the weight assigned to observation i depending on the cluster group. In other words, $w_i = a_{k(i)}$, where $k = k(i)$ is the cluster group number of observation i and the constant $a_k = a_{k(i)}$ is the constant weight assigned to cluster k. From my criteria, $a_2 > a_1$ since Cluster 2 is Downtown while Cluster 1 is the outskirt suburbs of Boston. The loss function is named `bis557::final_loss1()`.

This experiment will include two sets of weights. The first set of weights will be a "*strict*" version, while the second set of weights will be a "*moderated*" version. Each set of weights show that different polynomial models have the lowest loss.

Now, we find the polynomial model with the smallest loss, using the weights $a = (0.5, 4, 0.9, 3)$ for each zipcode cluster (more severe penalties for expensive zipcodes and "lenient" weights for cheaper zipcodes).

```
#> [1] "Loss for all 6 polynomial models - strict weight (0.5, 4, 0.9, 3): "
#> [1] 9304.067 8143.688 8369.425 8665.424 8875.932 8219.665
```

Using the strict weights above, it looks like **model 2** has the smallest loss. Note that due to applying "*strict*" weights, a simpler polynomial model is chosen.

Next, we find the polynomial model with the smallest loss, using the weights $a = (0.7, 1.5, 1, 1.7)$ for each zipcode cluster (less severe penalties for expensive zipcodes and slightly less lenient penalties for cheaper zipcodes).

```
#> [1] "Loss for all 6 polynomial models - moderated weight (0.7, 1.5, 1, 1.7): "
#> [1] 7968.123 7331.498 7270.892 7428.419 7542.015 7085.068
```

Using the moderated weights above, it looks like **model 6** has the smallest loss.

The coefficients of polynomial model 2 (best model using strict weights) are shown below.

```
#>
#> Call:
#> lm(formula = AV_TOTAL_SQFT ~ poly(LAND_SF, 2) + YR_BUILT + poly(GROSS_AREA,
#>     2) + poly(LIVING_AREA, 2) + poly(NUM_FLOORS, 1) + poly(R_BDRMS,
#>     1) + poly(R_FULL_BTH, 1) + poly(R_KITCH, 1) + R_FPLACE +
#>     YR_LAST_UPGRA + ZIP_GROUP + poly(DIST_FROM_DT, 2), data = df[idx[2,
#>     ], ])
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -1141.76   -51.42    -3.76    46.23  2317.20
#>
#> Coefficients:
#>                           Estimate Std. Error t value Pr(>|t|)
#> (Intercept)            -2.550e+00  3.053e+01  -0.084 0.933417
#> poly(LAND_SF, 2)1       3.471e+03  1.085e+02  31.978  < 2e-16 ***
#> poly(LAND_SF, 2)2      -1.275e+03  9.218e+01 -13.835  < 2e-16 ***
#> YR_BUILT               -1.898e-01  1.554e-02 -12.210  < 2e-16 ***
#> poly(GROSS_AREA, 2)1    1.245e+03  3.420e+02   3.641 0.000271 ***
#> poly(GROSS_AREA, 2)2    2.888e+02  1.887e+02   1.531 0.125885
#> poly(LIVING_AREA, 2)1  -1.528e+04  3.970e+02 -38.492  < 2e-16 ***
#> poly(LIVING_AREA, 2)2   4.221e+03  1.904e+02  22.162  < 2e-16 ***
#> poly(NUM_FLOORS, 1)     2.326e+03  1.570e+02  14.817  < 2e-16 ***
#> poly(R_BDRMS, 1)       -9.718e+02  1.459e+02  -6.663 2.71e-11 ***
#> poly(R_FULL_BTH, 1)     4.092e+03  1.373e+02  29.810  < 2e-16 ***
#> poly(R_KITCH, 1)       -1.159e+03  1.361e+02  -8.515  < 2e-16 ***
#> R_FPLACE                2.975e+01  6.075e-01  48.973  < 2e-16 ***
#> YR_LAST_UPGRA           3.276e-01  9.536e-03  34.351  < 2e-16 ***
#> ZIP_GROUP2              7.982e+02  7.127e+00 112.006  < 2e-16 ***
#> ZIP_GROUP3              5.872e+00  1.606e+00   3.656 0.000257 ***
#> ZIP_GROUP4              4.932e+02  3.793e+00 130.036  < 2e-16 ***
#> poly(DIST_FROM_DT, 2)1 -8.388e+03  1.524e+02 -55.047  < 2e-16 ***
#> poly(DIST_FROM_DT, 2)2  1.485e+03  1.100e+02  13.503  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 84.47 on 42910 degrees of freedom
#> Multiple R-squared:  0.6957, Adjusted R-squared:  0.6956
#> F-statistic:  5450 on 18 and 42910 DF,  p-value: < 2.2e-16
```

Now, assess both model 2 (stricter weights) and model 6 (moderated weights) using best subset selection, using `leaps::regsubsets()`. Then, the adjusted R^2 and BIC would be measured. The plots for adjusted R^2 and BIC below shows the terms to be added one-by-one, from the simplest model to the full polynomial model.

Under "strict" weights for zipcode clustering, polynomial model 2 (lowest loss) has an adjusted R^2 (the higher, the better) of 0.70 and a BIC (the lower, the better) of about -73000. For both assessment metrics, the first variable to be added to the model is the coefficient `ZIP_GROUP` with treatment contrasts, followed by `GROSS_AREA`, `LIVING_AREA`, and `DIST_FROM_DT` (the best predictors of property value per sq ft). For some variables, such as `LIVING_AREA`, including the polynomial terms before some other variables shows that some variables are much more important to the model than other variables. For example, including the squared `LIVING_AREA` is more important in model 2 than `NUM_FLOORS`.

Under "moderated" weights for zipcode clustering, polynomial model 6 (lowest loss) has an adjusted R^2 (the higher, the better) of 0.71 and a BIC (the lower, the better) of about -76000. For both assessment

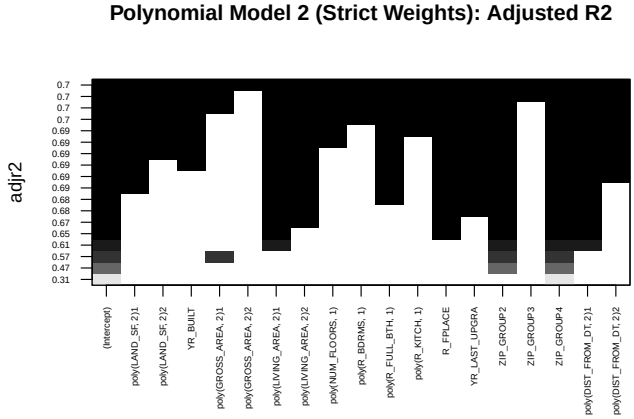

Figure 2: Best Subset Selection

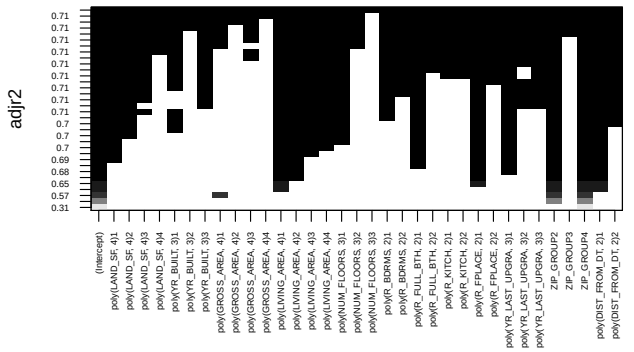

Figure 3: Best Subset Selection

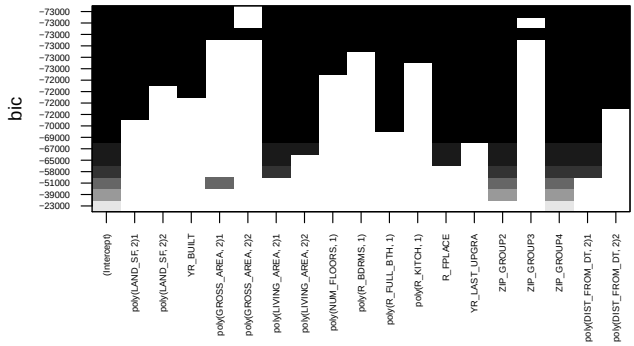

Figure 4: Best Subset Selection

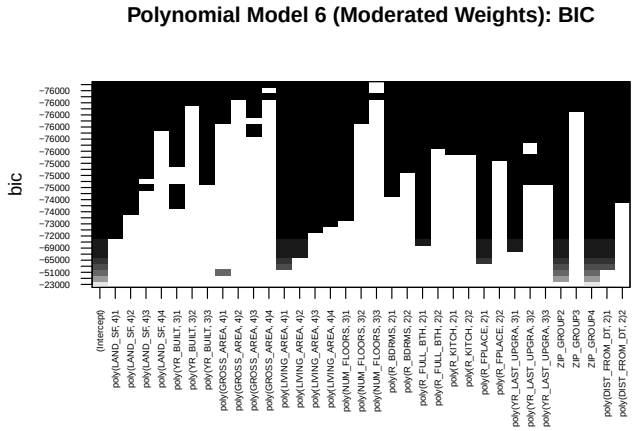

Figure 5: Best Subset Selection

metrics, the first variable to be added to the model is also the coefficient `ZIP_GROUP` with treatment contrasts, followed by `GROSS_AREA`, `LIVING_AREA`, and `DIST_FROM_DT` (the best predictors of property value per sq ft). For some variables, such as `LIVING_AREA`, including the polynomial terms before some other variables shows that some variables are much more important to the model than other variables. For example, including the 4th-power `LIVING_AREA` is more important in model 6 than `NUM_FLOORS`.

Generalized Additive Models

Another model to use on this dataset is a generalized additive model (GAM), where each predictor has a different function that best fits the data to give better estimates of `AV_TOTAL_SQFT`. To randomize the functions (by not prescribing a specific type/shape of a function), splines will be used with accompanying degrees of freedom, with the help of the `{splines}` library.

```
# GAM with varying degrees of freedom: use `splines::ns()` function
gams <- list()
for (k in 1:4) {
  gams[[k]] <- lm(AV_TOTAL_SQFT ~ ns(LAND_SF, 2*k) + ns(YR_BUILT, 2*k)
                  + ns(GROSS_AREA, 2*k) + ns(LIVING_AREA, 2*k)
                  + ns(NUM_FLOORS, 2*k) + ns(R_BDRMS, 2*k) + ns(R_FULL_BTH, 2*k)
                  + ns(R_KITCH, 2*k) + ns(R_FPLACE, 2*k) + ns(YR_LAST_UPGRA, 2*k)
                  + ZIP_GROUP + ns(DIST_FROM_DT, 2*k), data = df[idx[k,],])
}
```

The GAM models will again be fit into test data, and the same loss function will be used.

The GAM model assessment will also include "*strict*" weights, as well as "*moderated*" weights.

Using the strict weights $a = (0.5, 4, 0.9, 3)$, we have:

```
#> [1] "Loss for all 4 GAM models - strict weight (0.5, 4, 0.9, 3): "
#> [1] 8564.837 7762.325 8029.140 8378.913
```

Using the assigned weights above, it looks like **model 2** has the smallest loss. Note that due to applying "*strict*" weights, a simpler GAM model with smaller degrees of freedom is chosen.

Using the moderated weights $a = (0.7, 1.5, 1, 1.7)$, we have:

```
#> [1] "Loss for all 4 GAM models - moderated weight (0.7, 1.5, 1, 1.7): "
#> [1] 7332.265 6861.202 6840.008 7028.681
```

Using the assigned weights above, it looks like **model 3** has the smallest loss.

The coefficients of GAM model 2 (best model using strict weights) are shown below.

```
#>
#> Call:
#> lm(formula = AV_TOTAL_SQFT ~ ns(LAND_SF, 2 * k) + ns(YR_BUILT,
#>     2 * k) + ns(GROSS_AREA, 2 * k) + ns(LIVING_AREA, 2 * k) +
#>     ns(NUM_FLOORS, 2 * k) + ns(R_BDRMS, 2 * k) + ns(R_FULL_BTH,
#>     2 * k) + ns(R_KITCH, 2 * k) + ns(R_FPLACE, 2 * k) + ns(YR_LAST_UPGRA,
#>     2 * k) + ZIP_GROUP + ns(DIST_FROM_DT, 2 * k), data = df[idx[k,
#>     ], ])
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -639.96  -50.07   -3.43   45.87 2131.15
#>
#> Coefficients: (2 not defined because of singularities)
#>                            Estimate Std. Error t value Pr(>|t|)
#> (Intercept)                 809.318     49.673  16.293  < 2e-16 ***
#> ns(LAND_SF, 2 * k)1          51.672      2.736  18.883  < 2e-16 ***
#> ns(LAND_SF, 2 * k)2         180.629      7.296  24.758  < 2e-16 ***
#> ns(LAND_SF, 2 * k)3         127.822     19.328   6.613 3.81e-11 ***
#> ns(LAND_SF, 2 * k)4          -8.033     40.167  -0.200   0.8415
#> ns(YR_BUILT, 2 * k)1       -357.972     24.918 -14.366  < 2e-16 ***
#> ns(YR_BUILT, 2 * k)2       -187.244     14.714 -12.726  < 2e-16 ***
#> ns(YR_BUILT, 2 * k)3       -676.807     50.139 -13.499  < 2e-16 ***
#> ns(YR_BUILT, 2 * k)4       -209.761     11.583 -18.109  < 2e-16 ***
#> ns(GROSS_AREA, 2 * k)1       59.898      7.772   7.707 1.32e-14 ***
#> ns(GROSS_AREA, 2 * k)2       66.473     14.686   4.526 6.02e-06 ***
#> ns(GROSS_AREA, 2 * k)3       50.285     45.607   1.103   0.2702
#> ns(GROSS_AREA, 2 * k)4      -97.482     88.862  -1.097   0.2726
#> ns(LIVING_AREA, 2 * k)1    -418.213      9.559 -43.753  < 2e-16 ***
#> ns(LIVING_AREA, 2 * k)2    -661.436     18.941 -34.921  < 2e-16 ***
#> ns(LIVING_AREA, 2 * k)3    -668.217     57.641 -11.593  < 2e-16 ***
#> ns(LIVING_AREA, 2 * k)4     203.511    110.236   1.846   0.0649 .
#> ns(NUM_FLOORS, 2 * k)1       50.048      2.468  20.277  < 2e-16 ***
#> ns(NUM_FLOORS, 2 * k)2       33.064      4.759   6.947 3.77e-12 ***
#> ns(NUM_FLOORS, 2 * k)3      251.763      9.203  27.356  < 2e-16 ***
#> ns(NUM_FLOORS, 2 * k)4      388.678     17.813  21.820  < 2e-16 ***
#> ns(R_BDRMS, 2 * k)1         -44.283      7.883  -5.618 1.95e-08 ***
#> ns(R_BDRMS, 2 * k)2         -80.273      6.497 -12.355  < 2e-16 ***
#> ns(R_BDRMS, 2 * k)3         -34.189     16.970  -2.015   0.0439 *
#> ns(R_BDRMS, 2 * k)4          95.333     12.905   7.387 1.52e-13 ***
#> ns(R_FULL_BTH, 2 * k)1      178.851     34.282   5.217 1.83e-07 ***
#> ns(R_FULL_BTH, 2 * k)2      260.584     25.810  10.096  < 2e-16 ***
#> ns(R_FULL_BTH, 2 * k)3      429.260     72.704   5.904 3.57e-09 ***
#> ns(R_FULL_BTH, 2 * k)4      121.991     65.480   1.863   0.0625 .
#> ns(R_KITCH, 2 * k)1         -19.285     18.613  -1.036   0.3002
#> ns(R_KITCH, 2 * k)2           8.657     12.911   0.671   0.5025
#> ns(R_KITCH, 2 * k)3           1.505     13.292   0.113   0.9098
#> ns(R_KITCH, 2 * k)4         -75.078     30.347  -2.474   0.0134 *
#> ns(R_FPLACE, 2 * k)1       -253.402     20.924 -12.111  < 2e-16 ***
#> ns(R_FPLACE, 2 * k)2         22.960     17.278   1.329   0.1839
#> ns(R_FPLACE, 2 * k)3             NA         NA      NA       NA
```

```
#> ns(R_FPLACE, 2 * k)4            NA         NA      NA       NA
#> ns(YR_LAST_UPGRA, 2 * k)1  135.020     26.686   5.060 4.22e-07 ***
#> ns(YR_LAST_UPGRA, 2 * k)2  100.821     23.186   4.348 1.37e-05 ***
#> ns(YR_LAST_UPGRA, 2 * k)3  271.747     54.746   4.964 6.94e-07 ***
#> ns(YR_LAST_UPGRA, 2 * k)4   95.213      6.812  13.978  < 2e-16 ***
#> ZIP_GROUP2                 757.806      7.214 105.050  < 2e-16 ***
#> ZIP_GROUP3                   8.375      1.576   5.315 1.07e-07 ***
#> ZIP_GROUP4                 470.346      3.872 121.482  < 2e-16 ***
#> ns(DIST_FROM_DT, 2 * k)1   -91.512      2.825 -32.398  < 2e-16 ***
#> ns(DIST_FROM_DT, 2 * k)2  -124.668      8.446 -14.761  < 2e-16 ***
#> ns(DIST_FROM_DT, 2 * k)3  -166.513     17.652  -9.433  < 2e-16 ***
#> ns(DIST_FROM_DT, 2 * k)4   -52.988     31.148  -1.701   0.0889 .
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 81.41 on 42883 degrees of freedom
#> Multiple R-squared:  0.7175, Adjusted R-squared:  0.7172
#> F-statistic:  2420 on 45 and 42883 DF,  p-value: < 2.2e-16
```

The adjusted R^2 value is pretty high, at **0.72**, showing that a GAM model with lower degrees of freedom is effective.

Assessment: Zillow Data

The chosen polynomial models (model 2 and 6) and GAM models (model 2 and 3) will be *tested* on one of the very beautiful and luxurious homes that is selling in Boston, as of 12/14/2020. The property address, located in the Back Bay, is:

> 61 Saint Botolph St, Boston, MA 02116.

The website is:

> https://www.zillow.com/homedetails/61-Saint-Botolph-St-Boston-MA-02116/59188464_zpid/

The attribute of this property is below (3 bathrooms - 2 full and 1 half):

- Current Selling Price `AV_TOTAL`: 2299000 USD
- `LIVING_AREA`: 2124 sq ft
- `ZIP_GROUP`: 4 (Downtown Area 02116)

Thus, `AV_TOTAL_SQFT` equals $1082.39 per sq ft. The other attributes, from Zillow, are shown below:

```r
# Data vector for this property
zillow <- data.frame(AV_TOTAL = 2299000, AV_TOTAL_SQ_FT = 2299000/2124,
                     LAND_SF = 2178, YR_BUILT = 1900, GROSS_AREA = 2178,
                     LIVING_AREA = 2124, NUM_FLOORS = 2, R_BDRMS = 3,
                     R_FULL_BTH = 2, R_KITCH = 1, R_FPLACE = 1,
                     YR_LAST_UPGRA = 2018, ZIP_GROUP = factor(4),
                     DIST_FROM_DT = 0)

# Predictions
print(paste0("Predicted value/sqft - Polynomial Model #2: $",
             round(predict(fit2, zillow), 2)))
#> [1] "Predicted value/sqft - Polynomial Model #2: $894.19"
print(paste0("Predicted value/sqft - Polynomial Model #6: $",
             round(predict(fit6, zillow), 2)))
#> [1] "Predicted value/sqft - Polynomial Model #6: $861.73"
```

```
print(paste0("Predicted value/sqft - GAM Model #2: $",
             round(predict(gams[[2]], zillow), 2)))
#> [1] "Predicted value/sqft - GAM Model #2: $870.18"
print(paste0("Predicted value/sqft - GAM Model #3: $",
             round(predict(gams[[3]], zillow), 2)))
#> [1] "Predicted value/sqft - GAM Model #3: $857.32"
```

For this real-world test observation, all of the chosen models **underestimated** the true value per area of $1082.39 / sqft, probably because the models cannot measure the extreme skewness of luxurious property values in the Back Bay. The models chosen by the loss function with strict weights (poly model 2 and GAM model 2), with the more simple functions, have a higher and thus closer estimate than the models chosen by the loss function with moderated weights.

Conclusions and Further Research

Therefore, this abstract shows the analysis of 2020 Boston Residential Property Values in terms of clustering to incorporate geographical domain knowledge, and in terms of nonlinear models, including generalized additive models. It is better for the models to test the value per square foot, rather than the total value itself, to give a better comparison of real estate values. Multivariate methods, such as k-means clustering, were used to show the relationships among certain property characteristics according to property value, property size, and property age. These characteristics were used to group zipcodes and neighborhoods that are similar to each other, and the clusters showed apparent geographical patterns, which provided powerful insights of this dataset. The main result was that distance from Downtown Boston was a huge factor in explaining the distribution of `AV_TOTAL_SQFT` and the huge contrasts in different neighborhoods.

Attempting to create non-linear models, such as polynomial models and generalized additive models (GAM) with increasing complexity, makes a difference when user inputs and domain knowledge is incorporated into their loss functions. When observations in expensive zipcodes are *penalized heavily*, simpler polynomial models and GAM models are used in favor of more complex models, since simpler models tend to have *lower loss* values with "*strict*" weights. However, if the weights are moderated (smaller penalty for expensive zipcode clusters), then the more complex models are favored. Using best subset selection, using all of the terms of a particular predictor may be used before using the first term of another predictor (such as using all terms of `LIVING_AREA` before using the first term of `NUM_FLOORS`). The best GAM model under strict weights has a relatively high adjusted R^2 value of 0.72. Assessments of using best subset selection include adjusted R^2 and BIC.

Finally, a real-world example of a nice but expensive property in the Back Bay neighborhood of Boston. The polynomial and GAM models, both chosen using strict and moderated user-input weights, slightly underestimate the property value per sqft in this example. However, the simpler models (chosen by strict weights) give a slightly more accurate estimate, since there is *less* overfitting of skewed data points.

For further research, datasets that include similar information, but from **different** suburbs and cities in the Boston metro area (including North Shore, South Shore, and MetroWest) can be incorporated to provide more geographic analysis and more domain knowledge beyond Boston's city boundaries, which will definitely make models more creative. Also, if datasets that include more **economic information** (i.e. types of sectors and jobs with various income distributions) in each neighborhood are used, that can provide a richer analysis of residential property values. For example, a neighborhood can have high property values per sqft mainly due to the large prescence of a growing sector. Also, if this research becomes more geared to real estate **investment management**, datasets of rental prices and rental/tenant history can also be incorporated to give real estate investors (hopefully me someday!) a better sense of cap rate (ratio of net operating income NOI from tenants to the original asset/property price) and investment success. Finally, in a more technical standpoint, research on **choosing weights** would help make the loss function and the optimization function be more flexible but *not too complex*, yet more accurate, without overfitting extremely skewed outlier data. This is because choosing weights on zipcode clusters requires *domain knowledge* and can affect the *penalty*

inflicted on specific observations. Another way to research the usefulness of domain knowledge on this dataset is to create a special **index** that *scores* each observation according to geographic and economic attractiveness, which can make these models more useful while raising the adjusted R^2 score.