Predicting Total Wealth: A Predictive Analysis Using the 1991 SIPP Data

Xueshan (Kevin) Peng

Introduction

Loading and Inspecting the Data

Let's take a look at the first 6 rows of the data.

```
data <- read.table('data_tr.txt', head = T)[,-1]
head(data)
```

```
##       tw  ira e401  nifa   inc hmort   hval hequity educ male twoearn nohs hs
## 1  53550    0    0   100 28146 60150  69000    8850   12    0       0    0  1
## 2 124635    0    0 61010 32634 20000  78000   58000   16    0       0    0  0
## 3 192949 1800    0  7549 52206 15900 200000  184100   11    1       1    1  0
## 4   -513    0    0  2487 45252     0      0       0   15    0       1    0  0
## 5 212087    0    0 10625 33126 90000 300000  210000   12    0       0    0  1
## 6  24400    0    0  9000 76860 99600 120000   20400   15    0       1    0  0
##   smcol col age fsize marr
## 1     0   0  31     5    1
## 2     0   1  52     5    0
## 3     0   0  50     3    1
## 4     1   0  28     4    1
## 5     0   0  42     3    0
## 6     1   0  49     6    1
```

The variables in this dataset is defined as follows:

- tw: Total wealth (in US $), which is defined as "net financial assets, including Individual Retirement Account (IRA) and 401(k) assets, plus housing equity plus the value of business, property, and motor vehicles."
- ira: individual retirement account (IRA) balance (in US $).
- e401: Binary variable, where 1 indicates eligibility for a 401(k)-retirement plan, and 0 indicates otherwise.
- nifa: Non-401k financial assets (in US $).
- inc: Income (in US $).
- hmort: Home mortgage (in US $).
- hval: Home value (in US $).
- hequity: Home value minus home mortgage.
- educ: Education (in years).
- male: Binary variable, where 1 indicates male and 0 indicates otherwise.
- twoearn: Binary variable, where 1 indicates two earners in the household, and 0 indicates otherwise.

- nohs, hs, smcol, col: Dummy variables for education levels - no high school, high school, some college, college.
- age: Age.
- fsize: Family size.
- marr: Binary variable, where 1 indicates married and 0 indicates otherwise.

```
colSums(is.na(data))
```

```
##      tw     ira    e401    nifa     inc   hmort    hval hequity    educ    male
##       0       0       0       0       0       0       0       0       0       0
## twoearn    nohs      hs   smcol     col     age   fsize    marr
##       0       0       0       0       0       0       0       0
```

```
any(duplicated(data))
```

```
## [1] FALSE
```

We can see that the data is in good shape, where categorical variables are already transformed into dummy variables. We can also see that there exists multi-collinearity in education levels (**nohs**, **hs**, **smcol**, **col**) and home-ownership-related variables (**hmort**, **hval**, and **hequity**).

```
summary(data)
```

```
##        tw                ira              e401             nifa
##  Min.   :-502302   Min.   :     0   Min.   :0.0000   Min.   :      0
##  1st Qu.:   3246   1st Qu.:     0   1st Qu.:0.0000   1st Qu.:    200
##  Median :  25225   Median :     0   Median :0.0000   Median :   1687
##  Mean   :  63629   Mean   :  3471   Mean   :0.3714   Mean   :  13611
##  3rd Qu.:  82173   3rd Qu.:     0   3rd Qu.:1.0000   3rd Qu.:   8875
##  Max.   :1887115   Max.   :100000   Max.   :1.0000   Max.   :1425115
##       inc              hmort             hval           hequity
##  Min.   :    -9   Min.   :     0   Min.   :     0   Min.   :-40000
##  1st Qu.: 19413   1st Qu.:     0   1st Qu.:     0   1st Qu.:     0
##  Median : 31575   Median :  8000   Median : 50000   Median : 10000
##  Mean   : 37177   Mean   : 30207   Mean   : 63965   Mean   : 33757
##  3rd Qu.: 48615   3rd Qu.: 52000   3rd Qu.: 95000   3rd Qu.: 48000
##  Max.   :242124   Max.   :150000   Max.   :300000   Max.   :300000
##       educ            male            twoearn            nohs
##  Min.   : 1.0   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:12.0   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :12.0   Median :0.0000   Median :0.0000   Median :0.0000
##  Mean   :13.2   Mean   :0.2018   Mean   :0.3808   Mean   :0.1277
##  3rd Qu.:15.0   3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:0.0000
##  Max.   :18.0   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##        hs              smcol             col              age
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :25.00
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:32.00
##  Median :0.0000   Median :0.0000   Median :0.0000   Median :40.00
##  Mean   :0.3819   Mean   :0.2422   Mean   :0.2482   Mean   :41.08
##  3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:48.00
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :64.00
##      fsize            marr
```

```
##  Min.   : 1.00   Min.   :0.0000
##  1st Qu.: 2.00   1st Qu.:0.0000
##  Median : 3.00   Median :1.0000
##  Mean   : 2.87   Mean   :0.6075
##  3rd Qu.: 4.00   3rd Qu.:1.0000
##  Max.   :13.00   Max.   :1.0000
```

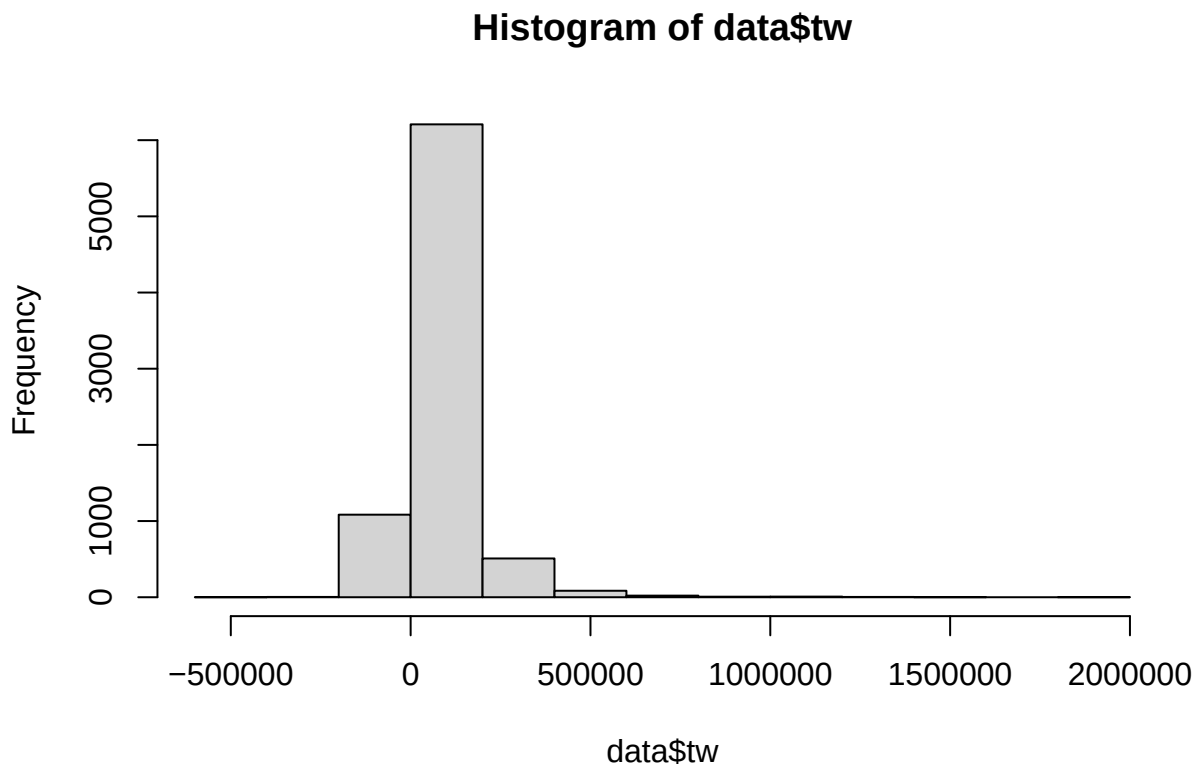
While there exist observations where total wealth is negative, it should be noted that the variable includes home equity, which can be negative, so it does not necessarily indicate that there are incorrect data entries.

The variables **ira**, **nohs**, **smcol**, **col**, and **male** exhibited a value of 0 at the 3rd quantile. They are probably a significant number of data points taking on the value of 0. Since **male** is on the list, it should also be noted that most observations are associated with female participants.

Also, the variable **tw**, **nifa**, **hmort**, and **hequity** have means that are much greater than medians, showing signs of large outliers.

In the histogram below, we can visualize the existence of outliers with enormous wealth.

```
hist(data$tw)
```



Using the graph, we can determine that removing the outliers with **tw** above $1,000,000 would be appropriate.

```
data = subset(data, data$tw<1000000)
```

Testing and Removing Multi-collinearity

Let's test whether removing different educational level predictors affect my model's performance, gauged by (MSPE). For simplicity sake, I did not use k-fold cross validation.

```
k <- 10
set.seed(123)
rand <- sample(nrow(data), floor(nrow(data)/k))
train <- setdiff(c(1:nrow(data)), rand)
y_rand <- data$tw[rand]

regnohs <- lm(tw ~ 1 + hs + smcol + col, data = data[train,])
reghs <- lm(tw ~ 1 + nohs + smcol + col, data = data[train,])
regsmcol <- lm(tw ~ 1 + nohs + hs + col, data = data[train,])
regcol <- lm(tw ~ 1 + nohs + hs + smcol, data = data[train,])

prnohs <- predict(regnohs, newdata = data[rand,])
prhs <- predict(reghs, newdata = data[rand,])
prsmcol <- predict(regsmcol, newdata = data[rand,])
prcol <- predict(regcol, newdata = data[rand,])

MSEnohs <- mean((y_rand-prnohs)^2)
MSEhs <- mean((y_rand-prhs)^2)
MSEsmcol <- mean((y_rand-prsmcol)^2)
MSEcol <- mean((y_rand-prcol)^2)

c(MSEnohs, MSEhs, MSEsmcol, MSEcol)
```

```
## [1] 9119936474 9119936474 9119936474 9119936474
```

No difference in performance is found between removing different terms for multi-collinearity. For interpretability, we choose to remove **hs** for education level.

Removing Redundancies

Since **hequity** represents home value minus home mortgage, it is intuitively a better predictor of total wealth than **hval** or **hmort** itself. Hence, choosing **hequity** over **hval** and **hmort** is the more sensible choice.

Including years of education (**educ**) along with education levels is redundant. Considering that diplomas are usually much more important than years of education, prioritizing education level over years of education is appropriate.

```
data <- data[, !(names(data) %in% c("hs", "hval", "hmort", "educ"))]
```

Creating Linear Models

Creating a Simple Linear Model and Setting a Baseline

Let's create a simple linear model. This will serve as a baseline to compare to when we later filter through features, add nonlinear transformations, and add interaction terms.

```r
baseline <- lm(tw ~., data=data)
summary(baseline)
```

```
##
## Call:
## lm(formula = tw ~ ., data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -514960  -12366   -3165    4239  661628
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.681e+04  2.344e+03  -7.171 8.14e-13 ***
## ira          1.612e+00  5.112e-02  31.534  < 2e-16 ***
## e401         8.286e+03  9.252e+02   8.956  < 2e-16 ***
## nifa         1.103e+00  1.447e-02  76.251  < 2e-16 ***
## inc          2.572e-01  2.481e-02  10.366  < 2e-16 ***
## hequity      1.082e+00  9.622e-03 112.471  < 2e-16 ***
## male         3.831e+03  1.162e+03   3.297 0.000981 ***
## twoearn     -6.510e+03  1.192e+03  -5.462 4.86e-08 ***
## nohs         2.518e+02  1.407e+03   0.179 0.857962
## smcol        1.434e+03  1.116e+03   1.285 0.198841
## col         -2.560e+02  1.190e+03  -0.215 0.829718
## age          2.553e+02  4.634e+01   5.510 3.71e-08 ***
## fsize       -8.385e+01  3.464e+02  -0.242 0.808775
## marr         2.652e+03  1.341e+03   1.978 0.048002 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 37700 on 7905 degrees of freedom
## Multiple R-squared:  0.8455, Adjusted R-squared:  0.8453
## F-statistic:  3329 on 13 and 7905 DF,  p-value: < 2.2e-16
```

Running a OLS regression on all of the features yields the above result. Surprisingly, after removing some of features which we considered to be redundant to predicting **tw**, most features are marked as significant.

Let's now assess the predictive performance of such a model.

For better accuracy, I employed 10-fold cross validation. Leave-one-out cross validation would yield a even more accurate result, but doing so on a dataset containing 7919 observations would take too much computational power.

In this Markdown file, I have saved complex/reusable code as functions in separate R Script files, which I will import to have them utilized. Feel free to check them out!

```r
source("KfoldCVFunctions.R")

mean_mspe_ols <- compute_ols_mspe(data, response_var = "tw")

cat("Mean MSPE of this model is", mean_mspe_ols, ".\n")
```

```
## Mean MSPE of this model is 1428001865 .
```

The mean MSPE of our OLS regression will than serve as our benchmark, which we will strive to improve upon when selecting features.

```
baseline <- benchmark <- mean_mspe_ols
```

Using Lasso and Forward/Backward Stepwise Selection

The baseline model, including all features, has overfitting issues. Hence, we should omit some features from our model to improve the predictive performance.

For this approach, we are going to include all the features in the dataset. We will let the feature selection algorithms, Lasso and Stepwise Selection, to select the features for us.

```
source("KfoldCVFunctions.R")

mean_mspe_lasso <- compute_lasso_mspe(data, response_var = "tw")
mean_mspe_forward <- compute_forward_stepwise_mspe(data, response_var = "tw")
mean_mspe_backward <- compute_backward_stepwise_mspe(data, response_var = "tw")

# Print results
print(c(Baseline = benchmark,
        Lasso = mean_mspe_lasso,
        Forward_Stepwise = mean_mspe_forward,
        Backward_Stepwise = mean_mspe_backward))
```

```
##          Baseline             Lasso  Forward_Stepwise Backward_Stepwise
##        1428001865        1427662119        1427849850        1427849850
```

As shown in the results above, all three feature selection methods helped us to obtain a better result. Lasso yielded a lower MSPE than forward or backward stepwise selection.

Let's now inspect the coefficients that Lasso chose and their associated p-values.

Since Lasso performs both variable selection and shrinkage, leading to biased coefficient estimates, traditional significance tests for coefficients (like p-values) are not straightforwardly available. Therefore, we use the *hdi* (High Dimensional Inference) package to approximate the p-values.

```
library(hdi)

response_var <- "tw"
y <- data[[response_var]]
X <- as.matrix(data[ , !(names(data) %in% response_var)])

lasso_cv <- cv.glmnet(X, y, alpha = 1)
best_lambda <- lasso_cv$lambda.min

lasso_model <- glmnet(X, y, lambda = best_lambda, alpha = 1)

lasso_inference <- hdi::lasso.proj(X, y)

print(coef(lasso_model))
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
```

```
##                        s0
## (Intercept) -1.576371e+04
## ira          1.606418e+00
## e401         7.935739e+03
## nifa         1.102051e+00
## inc          2.500733e-01
## hequity      1.081333e+00
## male         3.169643e+03
## twoearn     -5.386380e+03
## nohs         .
## smcol        1.006324e+03
## col          .
## age          2.470886e+02
## fsize        .
## marr         1.473061e+03
```

As shown above, Lasso has selected all of the features except nohs, col, and fsize. The coefficients are shown above.

```
print(lasso_inference$pval)
```

```
##           ira          e401          nifa           inc       hequity
## 5.152649e-198  1.807363e-17  0.000000e+00  1.982564e-23  0.000000e+00
##          male       twoearn          nohs         smcol           col
##  1.683754e-03  1.802943e-07  8.852471e-01  2.187133e-01  8.723161e-01
##           age         fsize          marr
##  1.172631e-07  7.921987e-01  5.334928e-02
```

Above are the approximated p-values the coefficients. We can gauge how strong of a predictor each feature is. As expected, ira and e401 have a really small p-value as they are literally a part of **tw**.

- tw: Total wealth (in US $), which is defined as "net financial assets, including Individual Retirement Account (IRA) and 401(k) assets, plus housing equity plus the value of business, property, and motor vehicles."

Let's also compute the MSPE for a simple OLS regression model and for a ridge regression model with the selected features for comparison.

```
data_subset <- data[, !(names(data) %in% c("nohs", "col", "fsize"))]

source("KfoldCVFunctions.R")
mean_mspe_ols <- compute_ols_mspe(data_subset, response_var = "tw")
mean_mspe_ridge <- compute_ridge_mspe(data_subset, response_var = "tw")

print(c(
  OLS_on_all_features = benchmark,
  OLS_on_selected_features = mean_mspe_ols,
  Ridge = mean_mspe_ridge,
  Lasso = mean_mspe_lasso,
  Forward_Stepwise = mean_mspe_forward,
  Backward_Stepwise = mean_mspe_backward
))
```

```
##      OLS_on_all_features OLS_on_selected_features                   Ridge
##                1428001865               1427051036              1449149272
##                    Lasso         Forward_Stepwise        Backward_Stepwise
##                1427662119               1427849850              1427849850
```

Surprisingly, a simple OLS regression on selected features yielded better results than ridge regression, Lasso, forward/backward stepwise selection.

The mean MSPE of our OLS regression will than serve as our new benchmark, which we will strive to improve upon when fitting nonlinear transformations.

```
benchmark = mean_mspe_ols
```

Finding Nonlinear Relationships and Applying Transformations

Inspecting the Relationships between tw and Features

After creating an appropriate linear model, the appropriate next step to improve predictive performance is through applying nonlinear transformations.

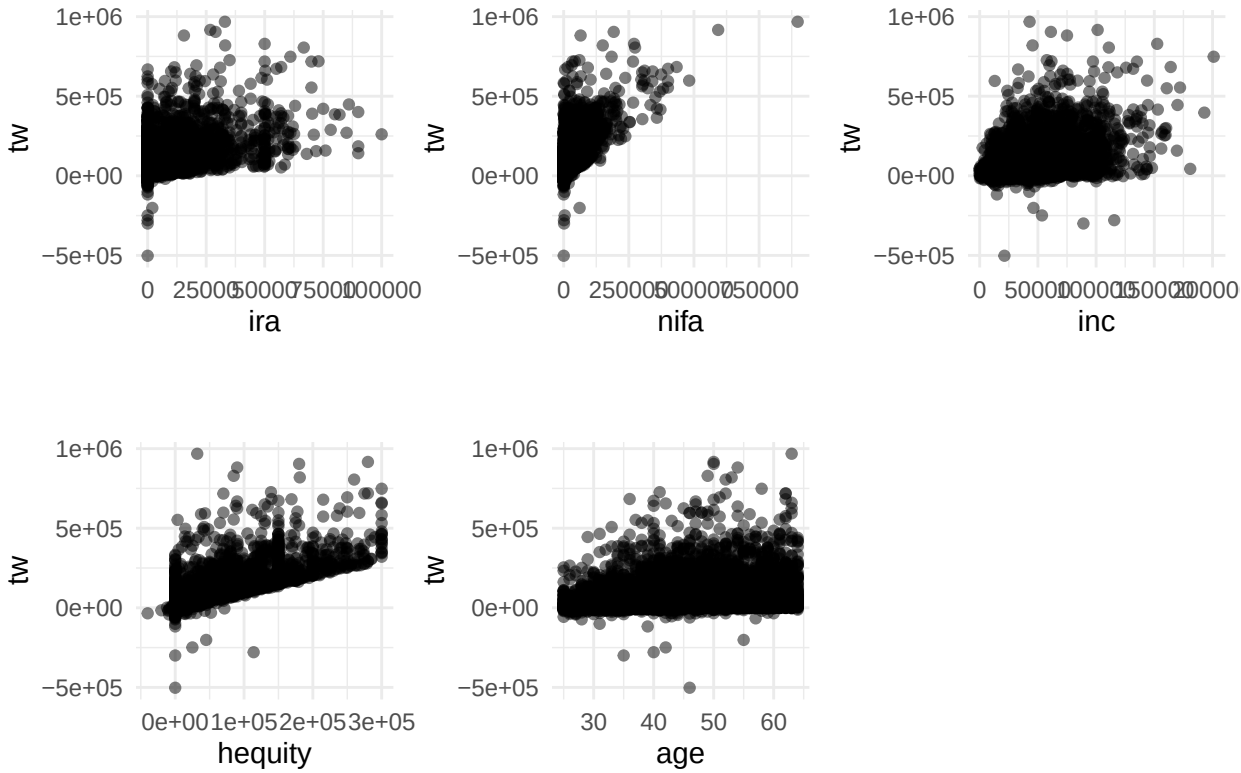
Let's first inspect the relationships between **tw** and all the quantitative (non-binary) features in the dataset. Nonlinear transformations are inappropriate for binary features because they only have two distinct values, making such transformations ineffective and potentially meaningless.

The below function compiles all the scatterplots into a list of *ggplot* objects.

```
library(tidyverse)
library(gridExtra)

plot_scatterplots <- function(data, response_var, feature_names) {

  plot_list <- list()

  for (feature in feature_names) {
    p <- ggplot(data = data) +
      geom_point(mapping = aes(x = !!sym(feature), y = !!sym(response_var)),
                 alpha = 0.5) +
      labs(x = feature, y = response_var) +
      theme_minimal() +
      theme(aspect.ratio = 1)
    plot_list[[feature]] <- p
  }

  return(plot_list)
}
```

We can then call the function and use the *gridExtra* package to arrange them.

```
plots <- plot_scatterplots(data_subset,
                           response_var = "tw",
                           c("ira", "nifa", "inc", "hequity", "age"))

do.call(grid.arrange, c(plots, ncol = 3))
```



Since **tw** is defined as net financial assets including **ira**, it is expected that there is a linear relationship between **tw** and **ira**, and this is what we can see on the scatterplot, too. The same goes with **hequity**.

Similarly, **nifa** is defined as non-401k financial assets and has a seemingly linear relationship with **tw** despite it having a more data points on the lower end than the higher end of **nifa**.

One that stands out as being nonlinear is the relationship between **age** and **tw**. In the area where there are more data points, represented by a darker color, there seems to be a sharper increase in **tw** at a younger age.

Applying Nonlinear Transformations to Age

For the following reasons, I will be using cubic splines as opposed to polynomials.

- Polynomials must use a high degree for flexible fits, but splines are able to do so with the degree fixed. This is likely to produce more stable estimates.
- Polynomials lack the ability to incorporate thresholds like splines, leading to undesirably global outcomes. In other words, observations within one range of the predictor strongly influence the model's behavior across different ranges.
- A polynomial fit is likely to produce undesirable results at the boundaries.

I have constructed a function to find the optimal number of knots and also where to place them. The function returns a set of knots, boundary knots (endpoints of the feature), as well as a plot showing the MSE vs. Number of Knots.

```r
source("select_optimal_knots.R")

result = select_optimal_knots(data, "tw", "age", 20)
print(result$knots_chosen)
```

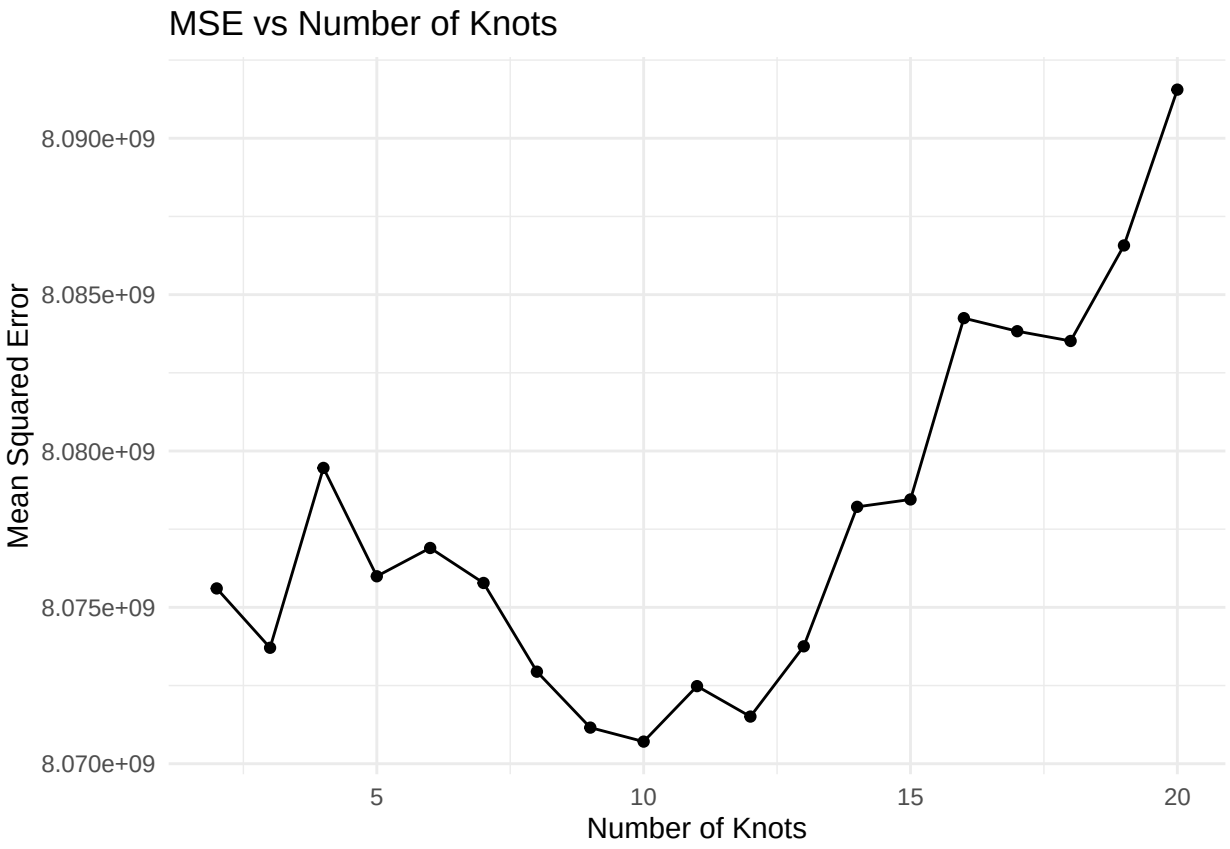
```
##  [1] 28.54545 32.09091 35.63636 39.18182 42.72727 46.27273 49.81818 53.36364
##  [9] 56.90909 60.45455
```

We can see that the algorithms returns the optimal set of knots, which, in this case, contains 10 items.

```r
print(result$boundary_knots)
```

```
## [1] 25 64
```

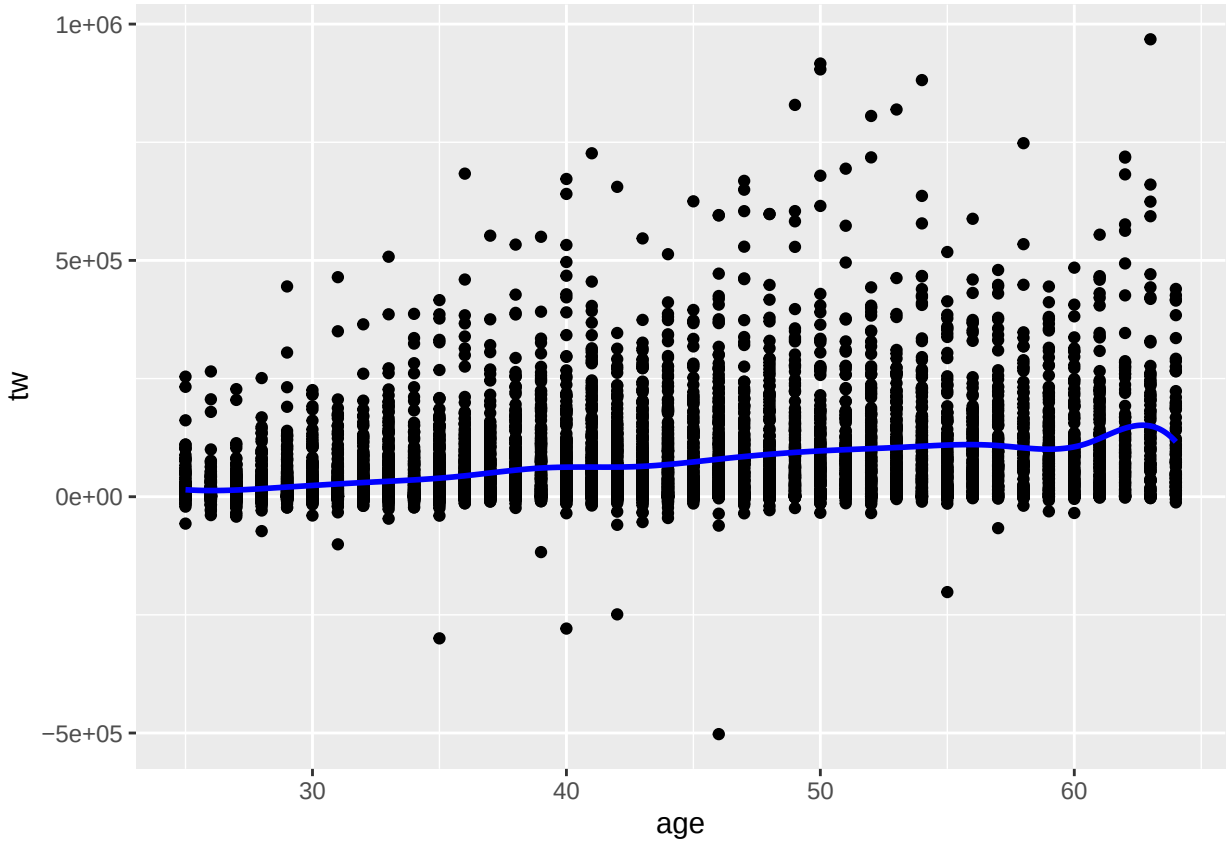
```r
print(result$plot)
```



We can see through the visualization that the MSE is the lowest when the number of knots is 10.

```r
age_spline_model <- lm(tw ~ bs(age,
                               knots = result$knots_chosen,
                               Boundary.knots = result$boundary_knots),
                       data = data)

x.grid <- seq(from = result$boundary_knots[1],
```

```
              to = result$boundary_knots[2],
              length.out = 1000)
pr <- predict(age_spline_model, newdata = data.frame(age = x.grid))

predictions <- data.frame(age = x.grid, tw = pr)

ggplot(data = data) +
  geom_point(mapping = aes(x = age, y = tw)) +
  geom_line(data = predictions,
            mapping = aes(x = age, y = tw),
            lwd = 1,
            color = "blue")
```



We can then use the previous output to construct a cubic spline model, from which predictions can be obtained and visualized through the graph.

However, the visualization is suggesting that there might be some overfitting of the data, as there is no reason for wealth to have a jump at the age of 62. We will validate this through the below analysis.

```
source("spline_transform_dataset.R")

transformed_data <- spline_transform_dataset(data,
                                             "age",
                                             result$knots_chosen,
                                             result$boundary_knots)
transformed_data_sub <- spline_transform_dataset(data_subset,
```

```
                                                "age",
                                                result$knots_chosen,
                                                result$boundary_knots)

source("KfoldCVFunctions.R")

mean_mspe_lasso <- compute_lasso_mspe(transformed_data, response_var = "tw")
mean_mspe_forward <- compute_forward_stepwise_mspe(transformed_data, response_var = "tw")
mean_mspe_backward <- compute_backward_stepwise_mspe(transformed_data, response_var = "tw")
mean_mspe_ols <- compute_ols_mspe(transformed_data_sub, response_var = "tw")
mean_mspe_ridge <- compute_ridge_mspe(transformed_data_sub, response_var = "tw")

print(c(
  OLS_without_transformations = benchmark,
  OLS = mean_mspe_ols,
  Ridge = mean_mspe_ridge,
  Lasso = mean_mspe_lasso,
  Forward_Stepwise = mean_mspe_forward,
  Backward_Stepwise = mean_mspe_backward
))
```

```
## OLS_without_transformations                         OLS
##                  1427051036                  1430609484
##                       Ridge                       Lasso
##                  1452436233                  1431052699
##           Forward_Stepwise           Backward_Stepwise
##                  1432084379                  1432221028
```

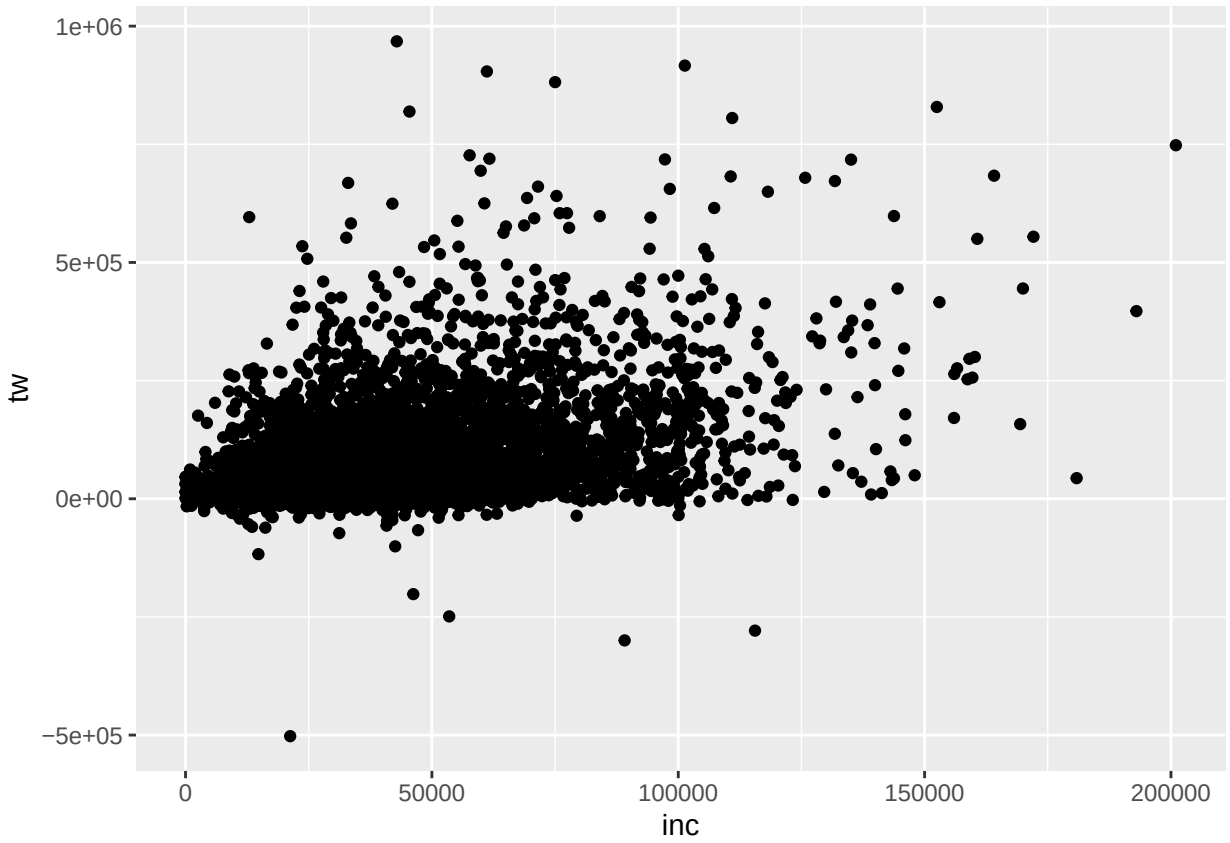
Applying the transformation to **age** and incorporating it into the dataset, we can now use General Additive Method (GAM) to compare the model's performance with the benchmark.

Confirming our previous guess about overfitting, the original linear model yielded a better performance, suggesting that the relationship between **tw** and **age** is approximately linear.

Applying Nonlinear Transformations to Income (inc)

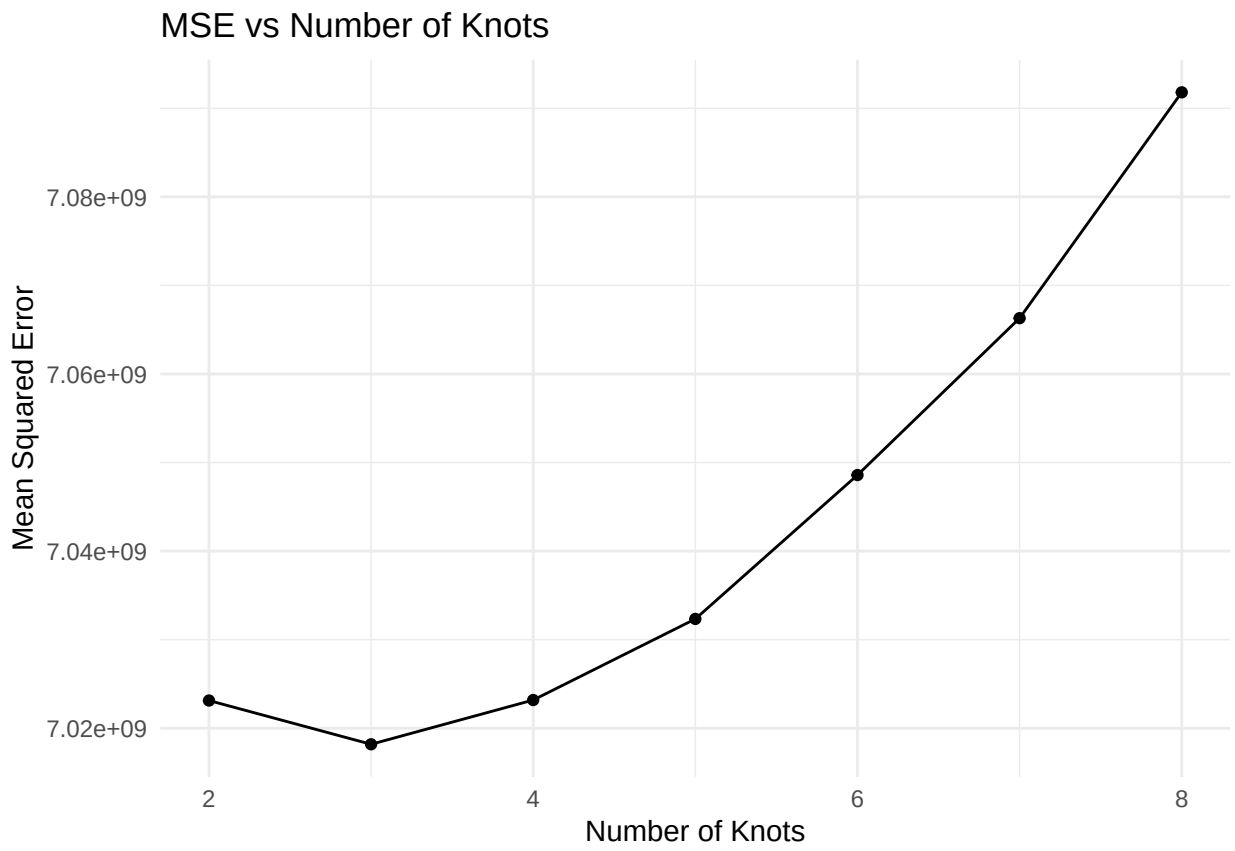
Let's now inspect the relationship between **tw** and **inc**, which looks a giant lump. This might suggest a nonlinear relationship.

```
ggplot(data = data) +
  geom_point(mapping = aes(x = inc, y = tw))
```



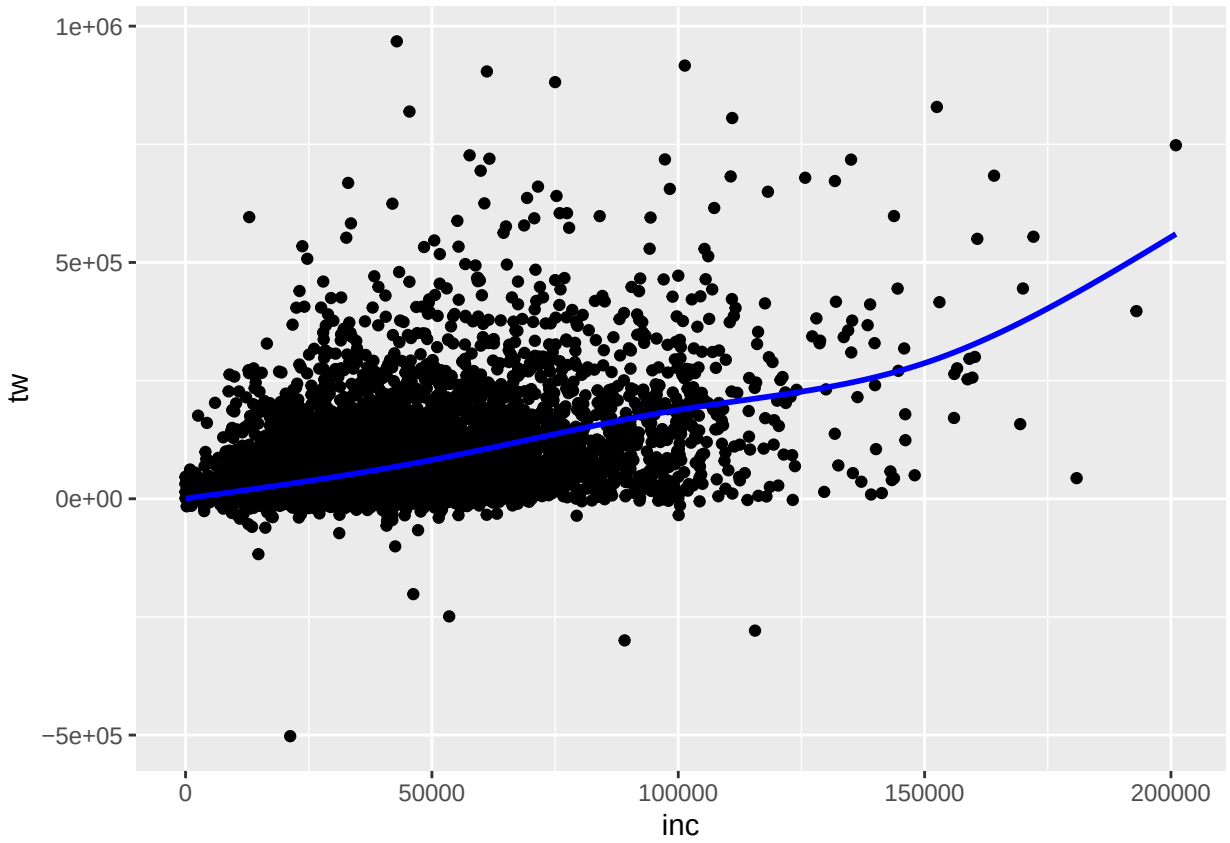
Notice that there are less observations on the higher end of the variable **inc**. Hence, a natural cubic spline would help reduce the variance which is otherwise substantial when **inc** takes on a large value. In the region where X is smaller than the smallest knot, or larger than the largest knot, a natural cubic spline constrains the fit to be linear.

```
source("select_optimal_knots_natural.R")

result = select_optimal_knots_natural(data, "tw", "inc", 8)
print(result$plot)
```



The optimal number of knots is 3.

```
inc_spline_model <- lm(tw ~ ns(inc,
                               knots = result$knots_chosen,
                               Boundary.knots = result$boundary_knots),
                       data = data)

x.grid <- seq(from = result$boundary_knots[1],
              to = result$boundary_knots[2],
              length.out = 1000)
pr <- predict(inc_spline_model, newdata = data.frame(inc = x.grid))

predictions <- data.frame(inc = x.grid, tw = pr)

ggplot(data = data) +
  geom_point(mapping = aes(x = inc, y = tw)) +
  geom_line(data = predictions,
            mapping = aes(x = inc, y = tw),
            lwd = 1,
            color = "blue")
```



The line fitted by the natural spline model is shown above.

```
source("natural_spline_transform_dataset.R")

transformed_data <- natural_spline_transform_dataset(data,
                                                     "inc",
                                                     result$knots_chosen,
                                                     result$boundary_knots)
transformed_data_sub <- natural_spline_transform_dataset(data_subset,
                                                         "inc",
                                                         result$knots_chosen,
                                                         result$boundary_knots)

source("KfoldCVFunctions.R")

mean_mspe_lasso <- compute_lasso_mspe(transformed_data, response_var = "tw")
mean_mspe_forward <- compute_forward_stepwise_mspe(transformed_data, response_var = "tw")
mean_mspe_backward <- compute_backward_stepwise_mspe(transformed_data, response_var = "tw")
mean_mspe_ols <- compute_ols_mspe(transformed_data_sub, response_var = "tw")
mean_mspe_ridge <- compute_ridge_mspe(transformed_data_sub, response_var = "tw")

print(c(
  OLS_without_transformations = benchmark,
  OLS = mean_mspe_ols,
  Ridge = mean_mspe_ridge,
  Lasso = mean_mspe_lasso,
```

```
  Forward_Stepwise = mean_mspe_forward,
  Backward_Stepwise = mean_mspe_backward
))
```

```
## OLS_without_transformations                         OLS
##                  1427051036                  1430497167
##                       Ridge                       Lasso
##                  1449872475                  1429314763
##           Forward_Stepwise          Backward_Stepwise
##                  1429742512                  1429742512
```

Once again, the original linear model yielded a better performance, suggesting that the relationship between **tw** and **inc** is approximately linear.

Conclusion for Finding Nonlinearity

Using the definition of **tw**, we have ascertained the linearity of its relationship with each of the following:

- **ira**
- **nifa**
- **hequity**

Also, applying General Additive Method and spline basis representation did not yield a better predictive performance, implying that the relationship of **tw** with **age** and **inc** are approximately linear as well.

We can therefore conclude that we have encountered bad luck with this dataset, where there are almost no room for nonlinear models to help improve predictive power.

Using Interaction Terms to Improve Performance

Adding interaction terms to a model can improve its predictive performance, especially if there are meaningful interactions between variables.

Selecting Interaction Terms

As mentioned before, based on the definition of **tw**, **tw** contains **ira**, **nifa**, and **hequity**. Hence, the coefficient to these variables should be close to 1 and shall not be influenced by other variables. Hence we will not consider these three variables for interaction terms.

We will be adding all possible interaction terms excluding those relevant to the aforementioned variables and then using feature selections methods to prune them to yield two important benefits:

- Ensuring that we don't miss any potentially important interaction terms
- Having less bias than manual selection

Using a custom-built function, we can input the variable we are predicting and the variables we want to exclude to obtain a dataset with the desired interaction terms.

```r
source("create_interactions.R")

data_with_interactions <- create_interactions(data_subset, "tw", c("ira", "nifa", "hequity"))

print(colnames(data_with_interactions))
```

```
##  [1] "tw"            "ira"           "e401"          "nifa"
##  [5] "inc"           "hequity"       "male"          "twoearn"
##  [9] "smcol"         "age"           "marr"          "e401:inc"
## [13] "e401:male"     "e401:twoearn"  "e401:smcol"    "e401:age"
## [17] "e401:marr"     "inc:male"      "inc:twoearn"   "inc:smcol"
## [21] "inc:age"       "inc:marr"      "male:twoearn"  "male:smcol"
## [25] "male:age"      "male:marr"     "twoearn:smcol" "twoearn:age"
## [29] "twoearn:marr"  "smcol:age"     "smcol:marr"    "age:marr"
```

Let's check whether we have the right columns. It seems that the function has done its job, obtaining $11 + (6 + 5 + 4 + 3 + 2 + 1) = 32$ columns.

```r
source("KfoldCVFunctions.R")

mean_mspe_lasso <- compute_lasso_mspe(data_with_interactions, response_var = "tw")
mean_mspe_forward <- compute_forward_stepwise_mspe(data_with_interactions, response_var = "tw")
mean_mspe_backward <- compute_backward_stepwise_mspe(data_with_interactions, response_var = "tw")

print(c(
  OLS_without_interactions = benchmark,
  Lasso = mean_mspe_lasso,
  Forward_Stepwise = mean_mspe_forward,
  Backward_Stepwise = mean_mspe_backward
))
```

```
## OLS_without_interactions                    Lasso         Forward_Stepwise
##               1427051036               1399701997               1399395813
##        Backward_Stepwise
##               1397018432
```

Based on the output above, we can conclude that adding interaction terms is the right choice. Using all three feature selection methods, we are able to obtain a better predictive performance than the benchmark.

Backward stepwise selection has yielded the best performance. Let's check what interaction terms it has chosen for our model.

```r
full <- lm(as.formula(paste(response_var, "~ .")), data=data_with_interactions)
null <- lm(as.formula(paste(response_var, "~ 1")), data=data_with_interactions)

backward <- stepAIC(full, scope=list(lower=null, upper=full), trace = FALSE, direction='backward')

summary(backward)
```

```
##
## Call:
## lm(formula = tw ~ ira + e401 + nifa + inc + hequity + male +
```

```
##     twoearn + age + marr + `e401:inc` + `e401:age` + `inc:male` +
##     `inc:twoearn` + `inc:age` + `inc:marr` + `male:smcol` + `twoearn:age`,
##     data = data_with_interactions)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -510126  -10679   -2669    2404  637032
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.327e+04  3.406e+03   3.897 9.80e-05 ***
## ira            1.569e+00  5.086e-02  30.843  < 2e-16 ***
## e401          -1.776e+04  4.043e+03  -4.393 1.13e-05 ***
## nifa           1.080e+00  1.442e-02  74.946  < 2e-16 ***
## inc           -7.064e-01  9.862e-02  -7.163 8.61e-13 ***
## hequity        1.076e+00  9.495e-03 113.317  < 2e-16 ***
## male          -6.836e+03  2.037e+03  -3.356 0.000793 ***
## twoearn        1.514e+04  4.339e+03   3.489 0.000487 ***
## age           -2.622e+02  7.467e+01  -3.511 0.000449 ***
## marr          -4.403e+03  2.099e+03  -2.097 0.035995 *
## `e401:inc`     2.879e-01  3.767e-02   7.642 2.39e-14 ***
## `e401:age`     3.657e+02  9.183e+01   3.982 6.89e-05 ***
## `inc:male`     2.883e-01  4.849e-02   5.946 2.87e-09 ***
## `inc:twoearn` -1.420e-01  4.845e-02  -2.930 0.003395 **
## `inc:age`      1.614e-02  2.068e-03   7.807 6.60e-15 ***
## `inc:marr`     2.284e-01  6.032e-02   3.787 0.000154 ***
## `male:smcol`   6.437e+03  2.278e+03   2.826 0.004725 **
## `twoearn:age` -3.786e+02  9.715e+01  -3.897 9.81e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 37210 on 7901 degrees of freedom
## Multiple R-squared:  0.8496, Adjusted R-squared:  0.8493
## F-statistic:  2626 on 17 and 7901 DF,  p-value: < 2.2e-16
```

It is looking promising as most of the features are highly significant with very small p-values.

The choice of interaction terms is intuitive, too. Just as an example, if a person is eligible for 401k retirement plan, which means the value of **e401** is 1, he/she is likely to accumulate more wealth with age than a person who is not eligible for the retirement plan. Hence, the coefficient of the interaction term **e401:age** would naturally be positive.

Visualizing Interaction between Features

Let's visualize this on a graph.

```
ggplot(data, aes(x = age, y = tw, color = factor(e401))) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(title = "Interaction between E401 and Age on TW")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

This is just one of the examples. All other interactions in the model make intuitive sense if you think hard enough. I am going to skip the explanations for the purpose of not making this Markdown file too long.

Assessing Performance Improvement

Let's now try the selected features with OLS and Ridge Regression.

```
data_with_selected_interactions <- data_subset %>%
    mutate(
      `e401:inc` = e401 * inc,
      `e401:age` = e401 * age,
      `inc:male` = inc * male,
      `inc:twoearn` = inc * twoearn,
      `inc:age` = inc * age,
      `inc:marr` = inc * marr,
      `male:smcol` = male * smcol,
      `twoearn:age` = twoearn * age
    ) %>%
    select(-smcol)

mean_mspe_ols <- compute_ols_mspe(data_with_selected_interactions, response_var = "tw")
mean_mspe_ridge <- compute_ridge_mspe(data_with_selected_interactions, response_var = "tw")

print(c(
  OLS_without_interactions = benchmark,
```

```
  OLS = mean_mspe_ols,
  Ridge = mean_mspe_ridge,
  Lasso = mean_mspe_lasso,
  Forward_Stepwise = mean_mspe_forward,
  Backward_Stepwise = mean_mspe_backward
))
```

```
## OLS_without_interactions                   OLS                 Ridge
##               1427051036            1394133060            1427462506
##                    Lasso       Forward_Stepwise      Backward_Stepwise
##               1399701997            1399395813            1397018432
```

With the selected features, our OLS model is once again able to lower the MSPE obtained through the 10-fold cross validation process.

This will conclude our selection for interaction terms. Let's now construct the final model.

Conclusion

Constructing the Final Model

```
formula <- as.formula(paste(response_var, "~ ."))
final_model <- lm(formula, data=data_with_selected_interactions)
summary(final_model)
```

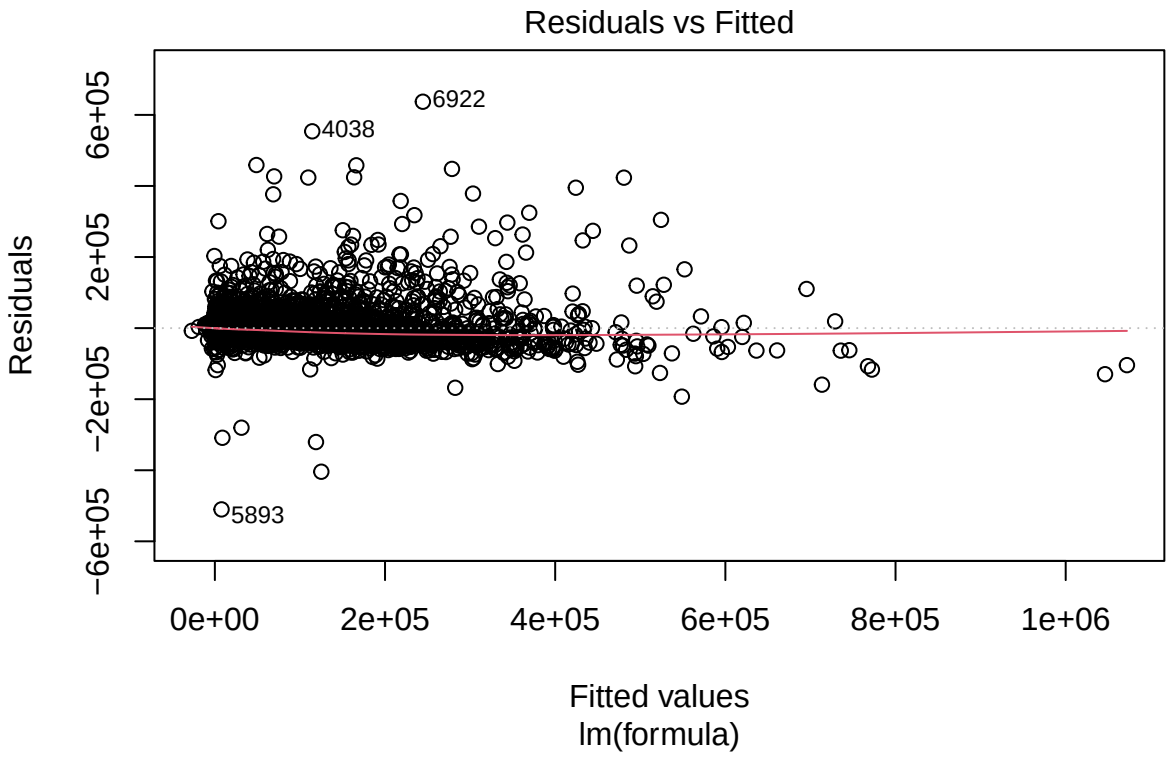
```
##
## Call:
## lm(formula = formula, data = data_with_selected_interactions)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -510126  -10679   -2669    2404  637032
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.327e+04  3.406e+03   3.897 9.80e-05 ***
## ira             1.569e+00  5.086e-02  30.843  < 2e-16 ***
## e401           -1.776e+04  4.043e+03  -4.393 1.13e-05 ***
## nifa            1.080e+00  1.442e-02  74.946  < 2e-16 ***
## inc            -7.064e-01  9.862e-02  -7.163 8.61e-13 ***
## hequity         1.076e+00  9.495e-03 113.317  < 2e-16 ***
## male           -6.836e+03  2.037e+03  -3.356 0.000793 ***
## twoearn         1.514e+04  4.339e+03   3.489 0.000487 ***
## age            -2.622e+02  7.467e+01  -3.511 0.000449 ***
## marr           -4.403e+03  2.099e+03  -2.097 0.035995 *
## `e401:inc`      2.879e-01  3.767e-02   7.642 2.39e-14 ***
## `e401:age`      3.657e+02  9.183e+01   3.982 6.89e-05 ***
## `inc:male`      2.883e-01  4.849e-02   5.946 2.87e-09 ***
## `inc:twoearn` -1.420e-01  4.845e-02  -2.930 0.003395 **
## `inc:age`       1.614e-02  2.068e-03   7.807 6.60e-15 ***
```

```
## `inc:marr`     2.284e-01  6.032e-02   3.787 0.000154 ***
## `male:smcol`   6.437e+03  2.278e+03   2.826 0.004725 **
## `twoearn:age` -3.786e+02  9.715e+01  -3.897 9.81e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 37210 on 7901 degrees of freedom
## Multiple R-squared:  0.8496, Adjusted R-squared:  0.8493
## F-statistic:  2626 on 17 and 7901 DF,  p-value: < 2.2e-16
```

Here we have the final model. It is unfortunate that we are unable to include any transformations for nonlinearity without impacting the predictive performance, but doing so is more suitable accounting for the linearity of the relationship between **tw** and features.

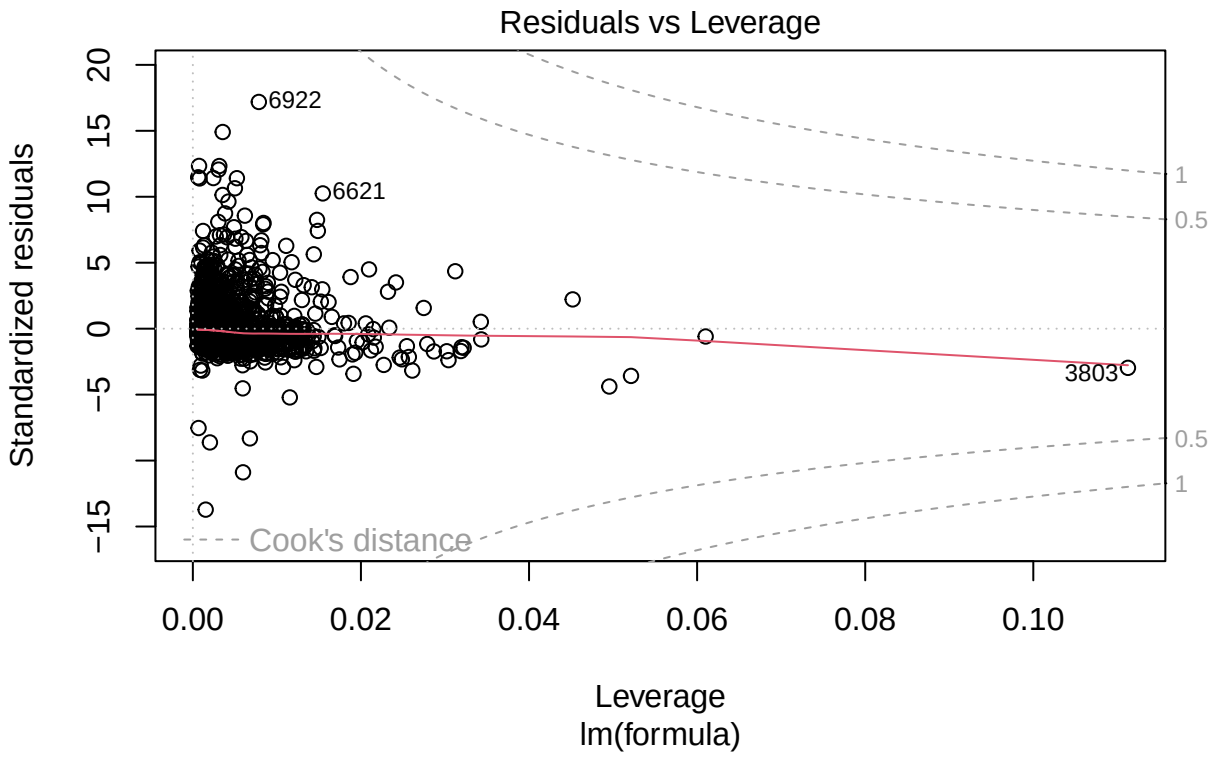
The coefficients and the associated p-values are shown above.

```
plot(final_model, which = 1)
```



The model demonstrates a strong fit to the data, as evidenced by the mostly flat residuals versus fitted values graph. This indicates that the model performs well across all areas of the data.

```
plot(final_model, which = 5)
```



Our model effectively identifies and excludes the correct outliers, as all data points fall well within a Cook's distance of 0.5. This indicates that no single point is unduly influencing the model's results.

Loading the Testing Data and Generating Predictions

With the model constructed, we can now create predictions. Let's prepare the dataset for predictions.

```
data_te <- read.table("data_for_prediction.txt", header = TRUE, sep = "\t", dec = ".")[,-1]

data_te <- data_te %>%
    mutate(
      `e401:inc` = e401 * inc,
      `e401:age` = e401 * age,
      `inc:male` = inc * male,
      `inc:twoearn` = inc * twoearn,
      `inc:age` = inc * age,
      `inc:marr` = inc * marr,
      `male:smcol` = male * smcol,
      `twoearn:age` = twoearn * age
    )

print(colnames(data_te))
```

```
##  [1] "ira"         "e401"        "nifa"        "inc"         "hmort"
##  [6] "hval"        "hequity"     "educ"        "male"        "twoearn"
```

```
## [11] "nohs"         "hs"          "smcol"       "col"         "age"
## [16] "fsize"        "marr"        "e401:inc"    "e401:age"    "inc:male"
## [21] "inc:twoearn" "inc:age"     "inc:marr"    "male:smcol"  "twoearn:age"
```

The prepared dataset now matches all the features from our model. Additional features that were not part of the model will not affect the prediction if they are not included in the model.

```
predictions <- predict(final_model, newdata=data_te)
data_te <- cbind(predicted_tw = predictions, data_te)
write.csv(data_te, "Predictions.csv")
write.table(predictions, file = 'Predictions.txt')
```

We can now use the model to create predictions and save it in a file.

Assessing Performance Improvements over Baseline

Let's assess our model's performance against that of the baseline model.

```
percentage_reduction <- round(((benchmark - mean_mspe_ols) / benchmark) * 100, 1)

print(c(
  MSPE_Basline_Model = baseline,
  MSPE_Final_Model = mean_mspe_ols,
  Percentage_Reduction = percentage_reduction
))
```

```
##   MSPE_Basline_Model     MSPE_Final_Model Percentage_Reduction
##         1428001864.6         1394133060.5                  2.3
```