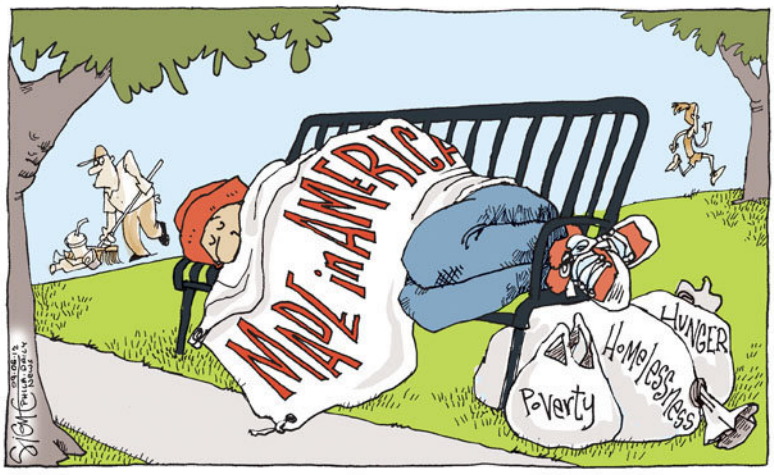
# Machine Learning Model Analysis to Predict Poverty Rates in the United States of America

Sunday, December 11th 2022

## Contents

**Rene Guerra**

**Fall 2022, UCSB**




### Poverty in America

After the second half of the twentieth century, the United States faced significant changes in demographic economics. Increasing birth rates and immigration benefited the labor force since there was a labor demand for agriculture, manufacturing, and other service industries. However, most of these communities were alienated from the American identity and showed an effect in many aspects, particularly income. Manual labor-driven communities included young American citizens and foreign workers who pursued a better standard of living. Many employers did not fulfill these conditions, making low-income communities experience wealth inequality. In this project, I will evaluate patterns on income data that support observations that are prevalent today since the mid-twentieth century. Also, I will predict potential current poverty rates based on regression models and will determine the best performing model to evaluate the data set.

### Project Significance in Statistical Labor Economics

Governmental institutions, such as the Bureau of Labor Statistics and the Census Bureau, research the impact of demographics in the labor force and consumption spending. Previous results indicate that there is a correlation between economic activity and specific communities in the United States. This project supports the discourse community and visualizes how the economic trends have performed for forty years. Also, prediction models can be useful for future estimations and retroactive comparisons of poverty rates in geographical or ethnic categories. Even though the best-performing model for this project may not be the best for other data sets, the performance of each model indicate potential optimization of outcome prediction. Thus, this project is a great tool for demographic analysis and labor force predictions.

**Project Performance Process**

This project will evaluate poverty rates across the United States based on `Age`, `Family Type`, `Ethnic Background`, `Region`, and `State` to predict what elements from these categories will have the highest rates of poverty in 2019 from a 40-year rate census. All observations are independent, results of an specific observation do not indicate dependence on a different category. Also, the data comes from the non-partisan government data reporting organization USAFACTS that gathered poverty rates information since 1950s. I will predict new rates based on the comprehensive information from 1980 to 2019.

There will be a total of 39 predictors (the years 1980 to 2018) that will attempt to predict the actual poverty rates in 2019 by each community. The variable names will be defined as `x1980` to `x2018` and the outcome `x2019`. There will be six regression models that will predict poverty rates in the training set and the best model will evaluate the testing set to conclude if the observed results match up will the regression results.

The first step will be an Exploratory Data Analysis of the relationships between predictors and outcome through visual resources and calculations between their correlations. This step will help the reader understand the consensus approach in the field and supportive results that will facilitate understanding how accurate are the predictions by model.

Next, I will split the data into training and testing sets by an established proportion. All models with be performed on the training data and the model with highest accuracy by rsq metric will evaluate a prediction on the testing data. Then, I will evaluate the accuracy of the testing results and state a reasoning behind the outcome.

After evaluating all the models on the training set and analyzing the best on the testing set, I will conclude the project with overall results, interesting observations and what I liked the most about this practice!

**Set Up Libraries**

There are some important libraries that will help run models, visualize results, and organize the prediction process.

```
library(readxl) #Read the original Excel data file
library(tidyverse) #Collection to load other packages
library(tidymodels) #Perform all models
library(ggplot2) #Create plots to visualize results
library(corrplot) #Crate correlation matrix between variables
library(rpart.plot) #Create tree plots for regression
library(ggthemes) #Make plots look more fun
library(cowplot) #Merge plots to evaluate results
library(glmnet) #Engine for ridge regression
library(vip) #Variable importance plot
library(janitor) #Clean column names
```

```
tidymodels_prefer() # Prioritize tidy models over other packages
```

### Data Cleaning

Let's import the original data file downloaded from USAFACTS and adjust it according to the purpose of this assessment.

```
Poverty1 <- read_excel("FinalProjectData.xlsx") %>%
  clean_names()
```

The category rows do not have data so we will remove those rows and group all the elements. Also, we want to change the header of the variables from "Years" to `Elements` since that column describes the elements in each category. Finally, I will consider the most recent and complete data for the period 1980-2018 to predict the outcome of poverty rates in 2019. Forty years of comprehensive data is sufficient to predict rates in the latest year. Thus, incomplete columns that not assert significant data will be removed.

Finally, the column names are numbers and R has trouble identifying them, instead it recognizes them as numeric values. Thus, I'll clean column names to prevent this confusion.

```
Poverty2 <- Poverty1[-c(2, 6, 12, 18, 23), -c(2:22, 63)]

Poverty <- Poverty2
colnames(Poverty)[colnames(Poverty) == "years"] <- "Elements"

View(Poverty)
```

We also need to assess the data by the categories `Age`, `Family Type`, `Ethnic Background`, `Region`, and `State`. Also, impute NA values for `Asian` with the first observed value, this is in 1987. I took this approach because imputing with other measurements such as mean of all observations in that row would consider a lower poverty rate than the estimated since current rates are much lower that overall rates in the last century. Thus, an imputation with the first observation means that the actual poverty rates for that population might be closer.

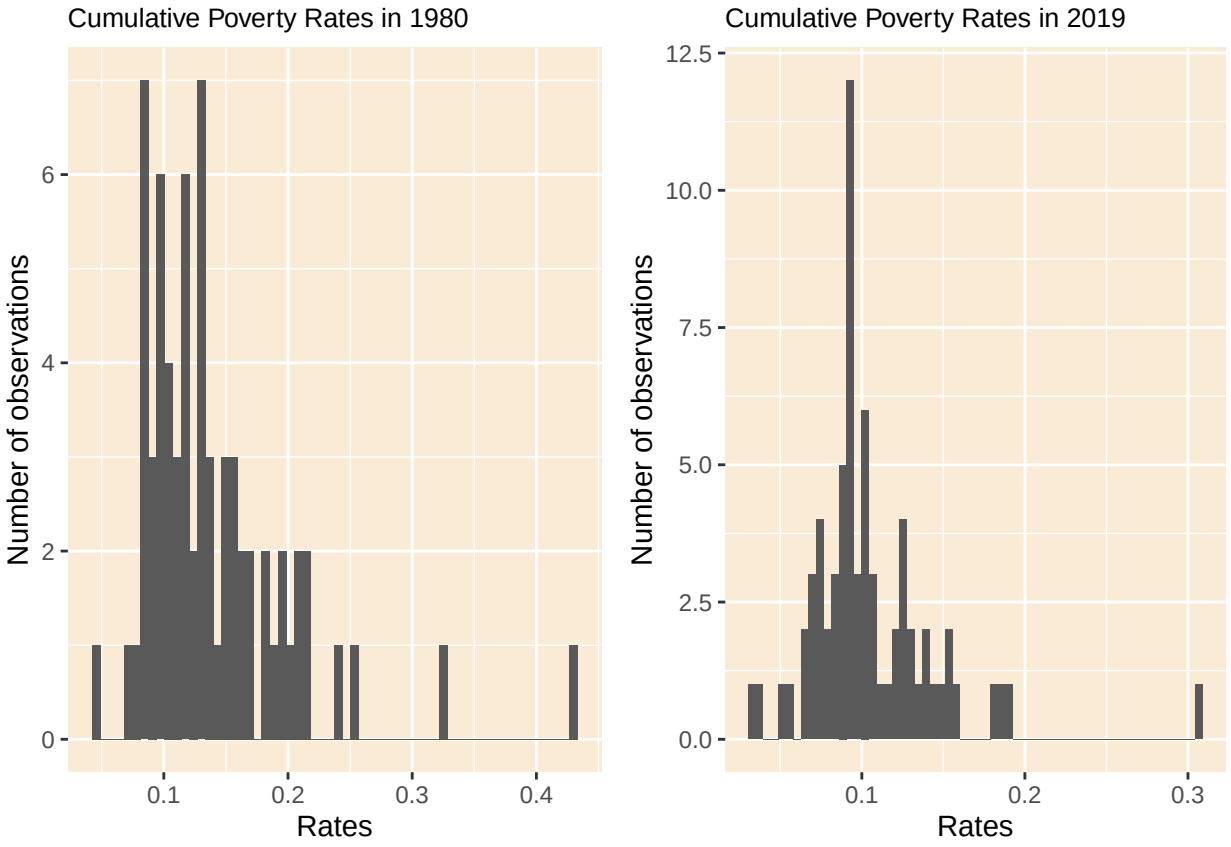
```
Age <- Poverty[2:4,]

FamilyType <- Poverty[5:9,]

Ethnicity <- Poverty[10:14,]

Ethnicity[is.na(Ethnicity)] <-  as.numeric(Ethnicity[3,9])

Poverty[is.na(Poverty)] <- as.numeric(Ethnicity[3,9])

Region <- Poverty[15:18,]

State <- Poverty[19:69,]
```

### Exploratory Data Analysis

Before performing the regression models, it is important to understand the relationship between the variables, the outcome and the categories that define each observation. Let's evaluate how poverty has changed in the last forty years in the United States.

```
Poverty1980 <- Poverty %>%
  ggplot(aes(x= x1980)) + geom_histogram(bins= 60) +
  labs(title= "Cumulative Poverty Rates in 1980",
       x= "Rates", y= "Number of observations" ) +
```

```
  theme(plot.title= element_text(size= 10),
        panel.background= element_rect(fill= "antiquewhite"))

Poverty2019 <- Poverty %>%
  ggplot(aes(x= x2019)) + geom_histogram(bins=60) +
  labs(title= "Cumulative Poverty Rates in 2019",
       x= "Rates", y= "Number of observations" ) +
  theme(plot.title= element_text(size= 10),
        panel.background= element_rect(fill= "antiquewhite"))

plot_grid(Poverty1980, Poverty2019)
```




The distribution of poverty rates in 1980 is in a broad range, most observations are between 9% and 15% in poverty. Also, there is one observation with a poverty rate of approximately 43%, almost half of that group did not meet the livable standard of living threshold in the United States! Surprisingly there was a group with a poverty rate of 4.5%, relatively low compared to other communities during that time.

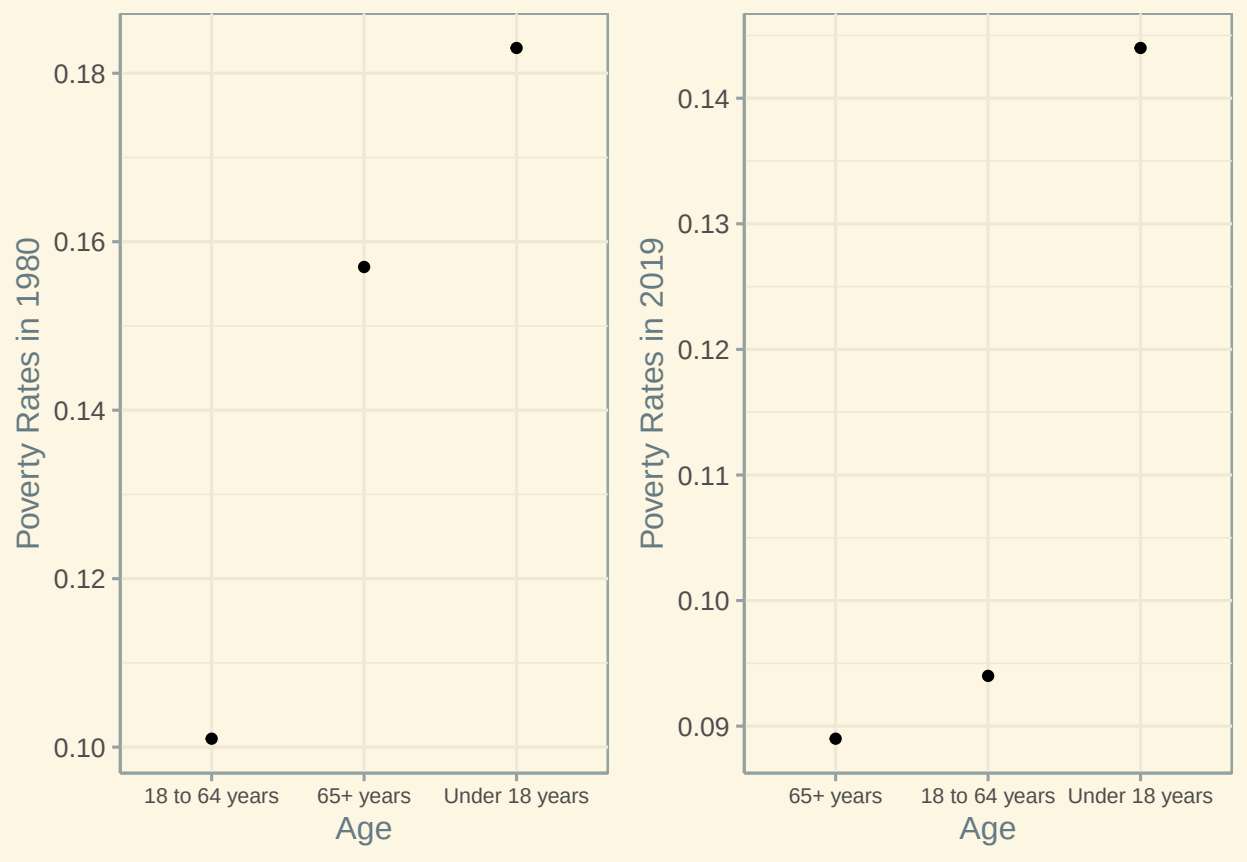
On the other hand, poverty decreased significantly by 2019. Most observations are between 6% and 12% in poverty and the highest is approximately 31%. There is a noticeable spike at around 8% where 12 population groups stand. Even though overall population poverty rates shifted to the left, the pattern is almost the same. We can see a singular observation all the way to right in both graphs and a similar lower bound for both years despite a 39-year difference.

In the previous analysis we found the position of each observation, let's now identify them.

```
AgeOrder1 <- factor(Age$Elements, levels= Age$Elements[order(Age$x1980)])
Age1980 <- ggplot(Age, aes(x= AgeOrder1, y= x1980, group= Elements)) +
```

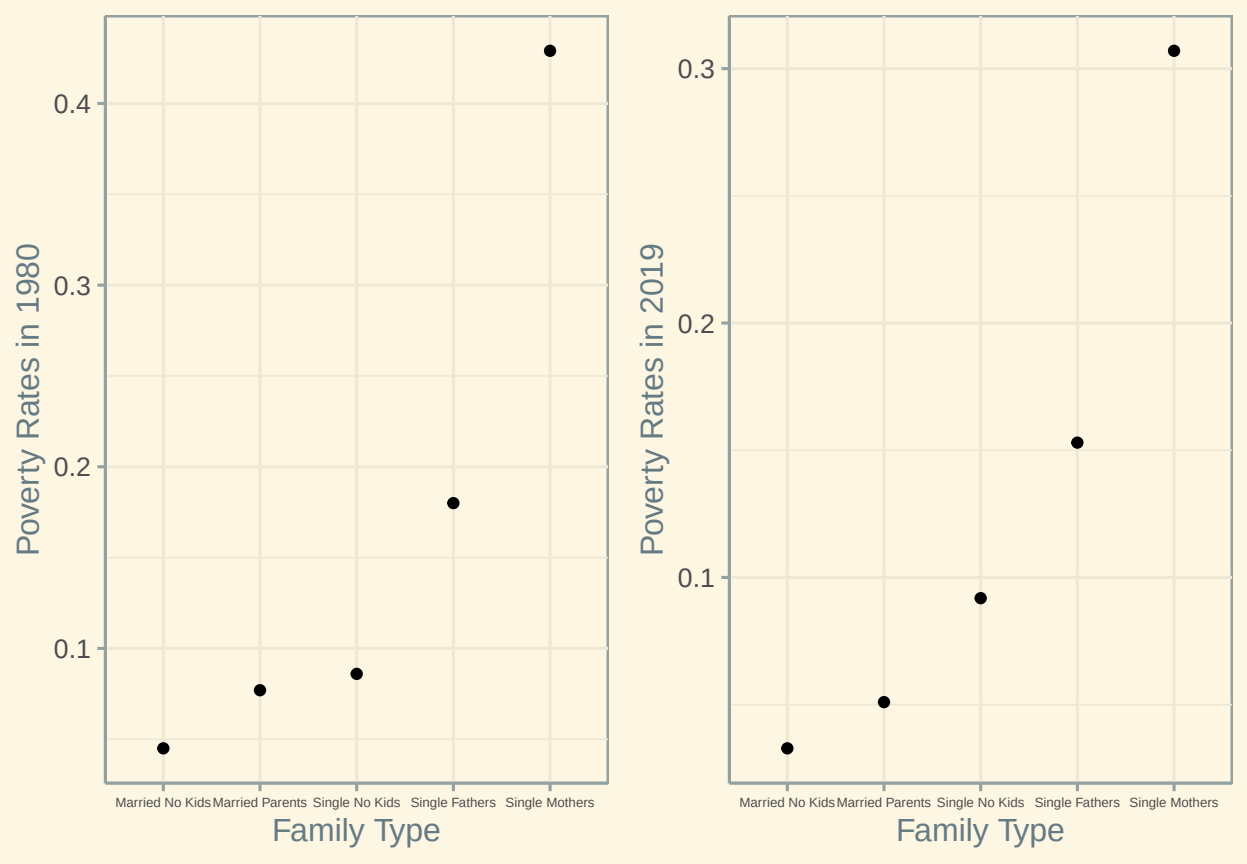
```
  geom_point() + theme_solarized() + labs(x= "Age", y= "Poverty Rates in 1980") +
  theme(axis.text.x= element_text(size= 8))

AgeOrder2 <- factor(Age$Elements, levels= Age$Elements[order(Age$x2019)])
Age2019 <- ggplot(Age, aes(x= AgeOrder2, y= x2019, group= Elements)) +
  geom_point() + theme_solarized() + labs(x= "Age", y= "Poverty Rates in 2019") +
  theme(axis.text.x= element_text(size= 8))


FamilyOrder1 <- factor(FamilyType$Elements, levels= FamilyType$Elements[order(FamilyType$x1980)])
Family1980 <- ggplot(FamilyType, aes(x= FamilyOrder1, y= x1980, group= Elements)) +
  geom_point() + theme_solarized() + labs(x= "Family Type", y= "Poverty Rates in 1980") +
  theme(axis.text.x= element_text(size= 5))

FamilyOrder2 <- factor(FamilyType$Elements, levels= FamilyType$Elements[order(FamilyType$x2019)])
Family2019 <- ggplot(FamilyType, aes(x= FamilyOrder2, y= x2019, group= Elements)) +
  geom_point() + theme_solarized() + labs(x= "Family Type", y= "Poverty Rates in 2019") +
  theme(axis.text.x= element_text(size= 5))


EthnicityOrder1 <- factor(Ethnicity$Elements, levels= Ethnicity$Elements[order(Ethnicity$x1980)])
Ethnicity1980 <- ggplot(Ethnicity, aes(x= EthnicityOrder1, y= x1980, group= Elements)) +
  geom_point() + theme_solarized() + labs(x= "Ethnicity", y= "Poverty Rates in 1980") +
  theme(axis.text.x= element_text(size= 5))

EthnicityOrder2 <- factor(Ethnicity$Elements, levels= Ethnicity$Elements[order(Ethnicity$x2019)])
Ethnicity2019 <- ggplot(Ethnicity, aes(x= EthnicityOrder2, y= x2019, group= Elements)) +
  geom_point() + theme_solarized() + labs(x= "Ethnicity", y= "Poverty Rates in 2019") +
  theme(axis.text.x= element_text(size= 5))


RegionOrder1 <- factor(Region$Elements, levels= Region$Elements[order(Region$x1980)])

Region1980 <- ggplot(Region, aes(x= RegionOrder1, y= x1980, group= Elements)) +
  geom_point() + theme_solarized() + labs(x= "Region", y= "Poverty Rates in 1980")

RegionOrder2 <- factor(Region$Elements, levels= Region$Elements[order(Region$x2019)])
Region2019 <- ggplot(Region, aes(x= RegionOrder2, y= x2019, group= Elements)) +
  geom_point() + theme_solarized() + labs(x= "Region", y= "Poverty Rates in 2019")


StateOrder1 <- factor(State$Elements, levels= State$Elements[order(State$x1980)])

State1980 <- ggplot(State, aes(x= StateOrder1, y= x1980, group= Elements)) +
  geom_point() + theme_solarized() + labs(x= "State", y= "Poverty Rates in 1980") +
  theme(axis.text.y= element_text(size= 5)) + coord_flip()

StateOrder2 <- factor(State$Elements, levels= State$Elements[order(State$x2019)])
State2019 <- ggplot(State, aes(x= StateOrder2, y= x2019, group= Elements)) +
  geom_point() + theme_solarized() + labs(x= "State", y= "Poverty Rates in 2019") +
  theme(axis.text.y= element_text(size= 5)) + coord_flip()
```

```
plot_grid(Age1980, Age2019)
```



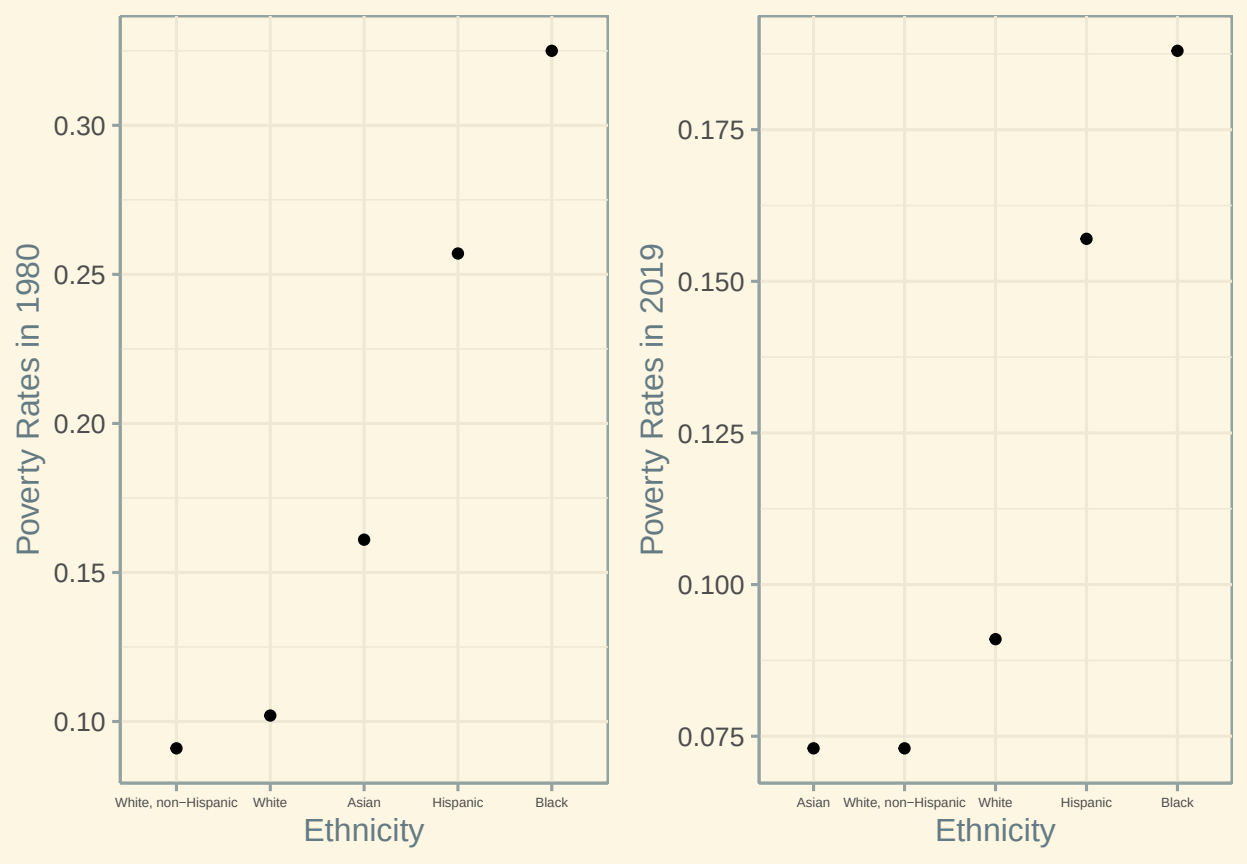

In both years individuals under 18 years were the poorest population in both 1980 and 2019 with similar rates. Under-aged people are a small portion of the labor force and most do not have income since they are dependent to their household. However, it surprises me that adults over the age of 64 were the population with the lowest poverty rate in 2019. This effect might be due to the increase in retirement, pension, and investment services that help adults finance their living after participating in the labor force. Finally, working adults between the ages of 18 and 64 years dropped to second place after having the lowest poverty rates in 1980.

```
plot_grid(Family1980, Family2019)
```



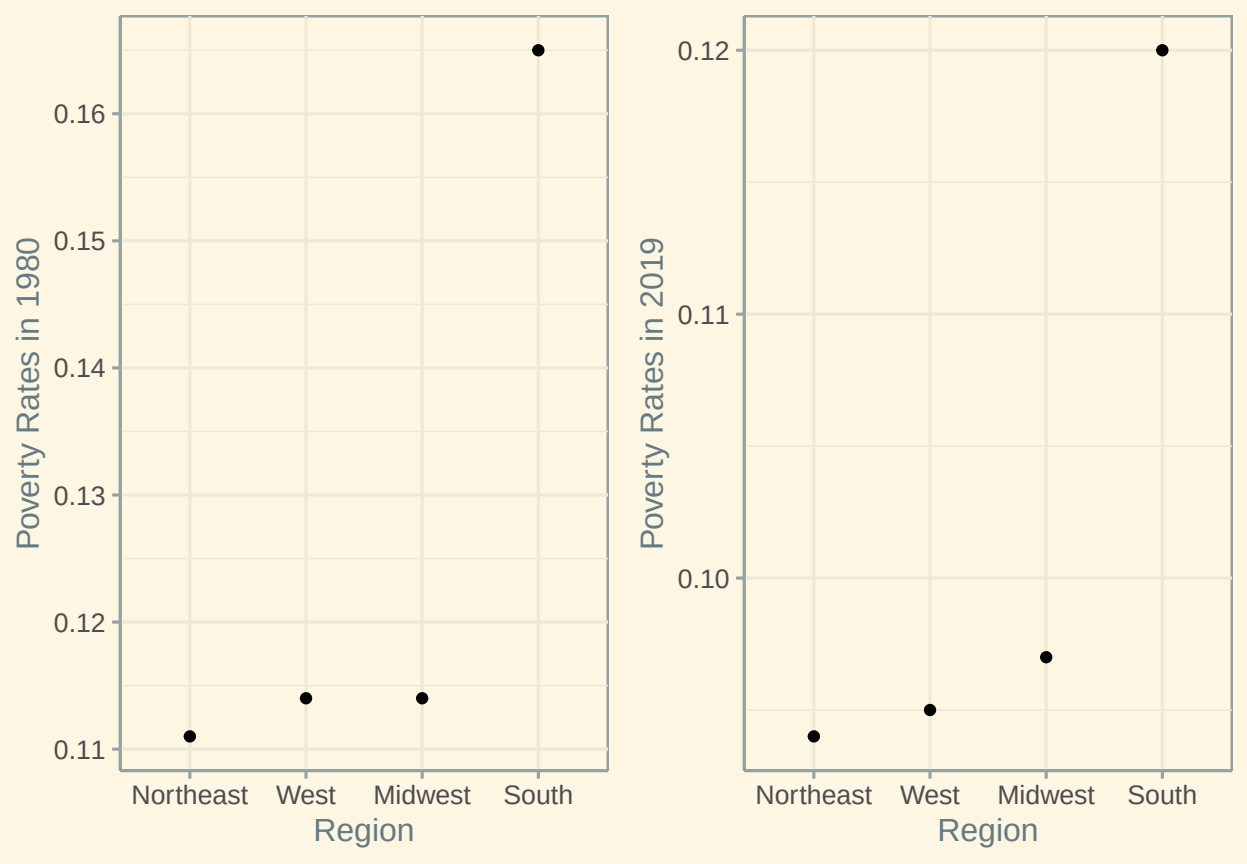

Family types have the same pattern for both years. Households with no kids had the lowest poverty rates, and we have found the highest poverty rate from all observations, single mothers. This effect might be due to the insufficient income from only one parent to support children, fact that affects multiple communities across the United States. It appears that the relationship by group is exponential where adults with no kids remain in the middle with relatively no significant change in poverty rate, remaining at approximately 9% for 40 years.

```
plot_grid(Ethnicity1980, Ethnicity2019)
```



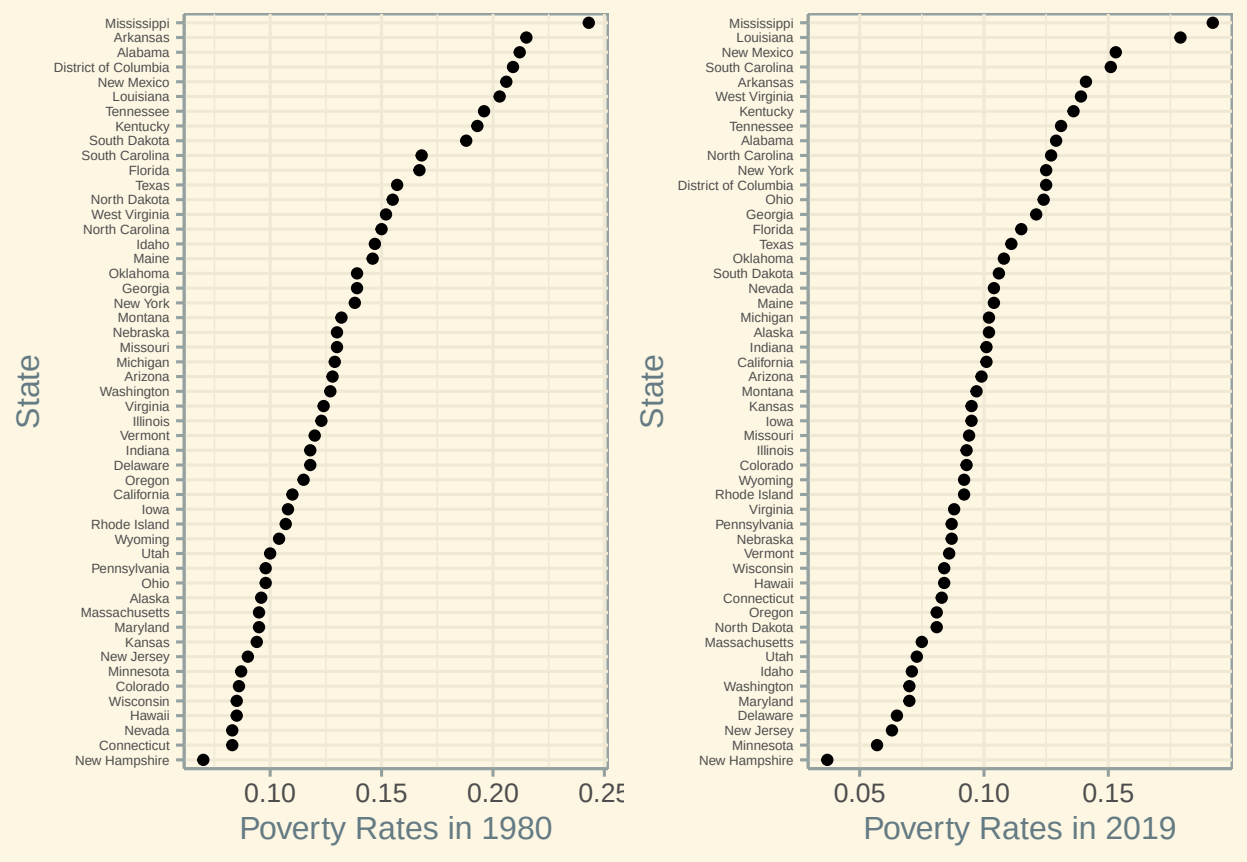

White population had the lowest poverty rate at 9% and black population the highest at 32%. Even though the Asian population's poverty rate was imputed, it is surprising how much it decreased, taking the lowest rates in 2019 at 7%. Even though poverty rates decreased enormously, communities of color continue be be affected by poverty for 40 years. It is important to analyze what is the direct effect on these groups that fail to meet livable life standards in the United States.

```
plot_grid(Region1980, Region2019)
```



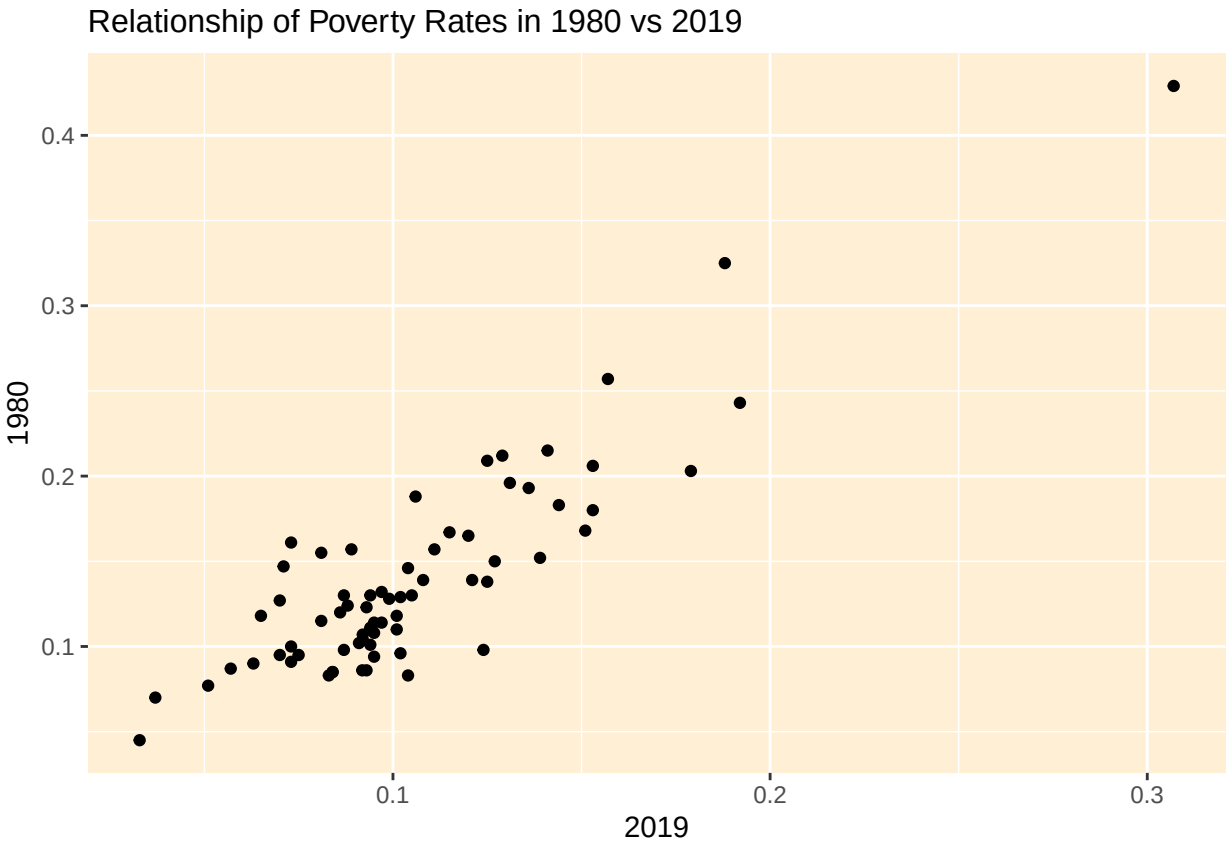

All regions in the United States had a significant decline in poverty rates in 2019. The pattern is the same, the northeast had the lowest rates in both years whereas the south had the highest rates by far. This reaction might be due to a limited domestic product in the southern states and low labor demand for this area. Another interesting observation is that the west outperformed the mid-west, moving from an equal rate at 11.4% to a 9.5% in 2019.

```
plot_grid(State1980, State2019)
```




These plots summarize poverty rates by state. There is a similar pattern for both years, and we see that the bounds are the same, New Hampshire with the lowest rate and Mississippi with the highest rate. However, states in between have changed. California, for example, decreased poverty rates in 2019, but was outperformed by other western states such as Oregon. This result correlates with the observations by region since most southern states remain at the upper bound of poverty, whereas northeastern states continue to have the lowest poverty rates in the country.

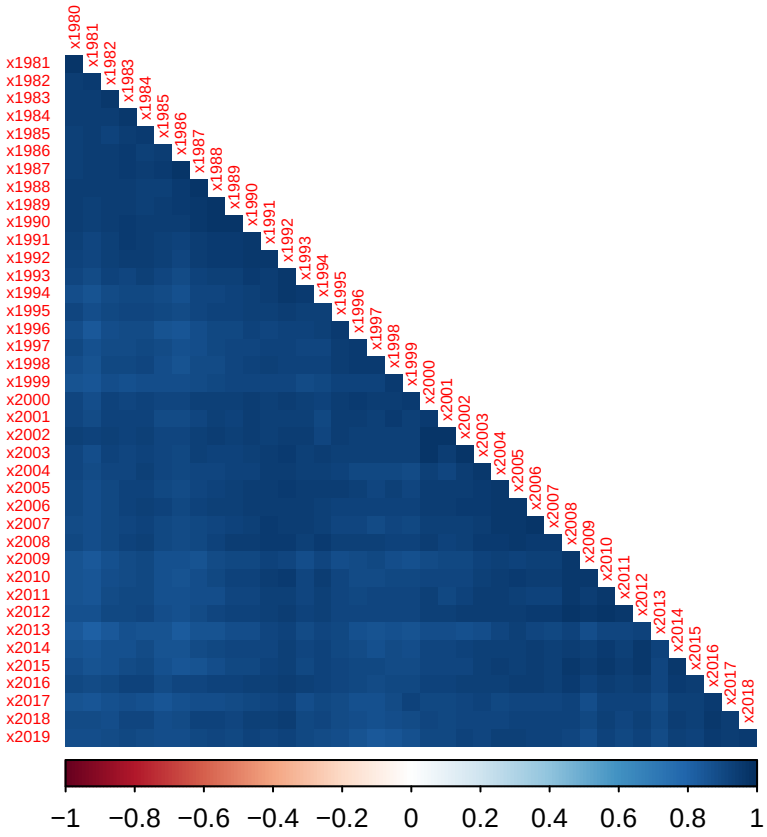
```
Poverty %>%
  ggplot(aes(x= x2019, y= x1980)) + geom_point() +
  labs(title= "Relationship of Poverty Rates in 1980 vs 2019", x= "2019", y= "1980" ) +
  theme(plot.title= element_text(size= 12), panel.background= element_rect(fill= "papayawhip"))
```




There is a direct relationship in poverty between 1980 and 2019. The lowest rates in 1980 were also the lowest rates in 2019. Likewise, we can spot the extremely high rate for both years, inducing that high rates in 1980 were equivalent in 2019.

Let's see the actual correlations between the variables.

```
Poverty %>%
  select(where(is.numeric)) %>%
  cor() %>%
  corrplot(method= "shade", type= "lower", diag= FALSE, tl.cex= 0.5)
```




All variables have positive correlations at different rates with the rest of variables. This result supports my observations from previous analysis, there is an impact in the observation for a year based on the rest. However, this means that variables bring similar information to the model. The high correlation between each pair of variables imply that a change in one will easily fluctuate to the other, thus affecting the outcome predictions for the project.

I will take appropriate steps later in the modeling process to address this detail to ensure prediction results are optimal.

### Data Splitting

Performing a split in the data helps obtain the best performing regression from all the models evaluating the training set. The proportion will be 70% of the data to the training set and 30% to the testing set. Also, a seed will be set to prevent changes in the randomization of the split every time I run the code.

```
set.seed(2022)

Poverty_split <- initial_split(Poverty,  prop= 0.7)

Poverty_train <- training(Poverty_split)
dim(Poverty_train)
```

```
## [1] 48 41
```

```
Poverty_train
```

```
## # A tibble: 48 x 41
##    Elements    x1980 x1981 x1982 x1983 x1984 x1985 x1986 x1987 x1988 x1989 x1990
##    <chr>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
```

```
##  1 New York    0.138 0.144 0.148 0.16  0.16  0.158 0.132 0.143 0.134 0.126 0.143
##  2 Oklahoma    0.139 0.138 0.154 0.17  0.133 0.16  0.147 0.17  0.173 0.147 0.156
##  3 Married Pa~ 0.077 0.087 0.098 0.101 0.094 0.089 0.08  0.077 0.072 0.073 0.078
##  4 Utah        0.1   0.122 0.145 0.135 0.111 0.109 0.126 0.102 0.098 0.082 0.082
##  5 White, non~ 0.091 0.099 0.106 0.108 0.1   0.097 0.094 0.087 0.084 0.083 0.088
##  6 Oregon      0.115 0.123 0.139 0.161 0.128 0.119 0.123 0.142 0.104 0.112 0.092
##  7 Single No ~ 0.086 0.114 0.115 0.119 0.106 0.1   0.092 0.098 0.089 0.084 0.085
##  8 Georgia     0.139 0.163 0.196 0.188 0.169 0.177 0.146 0.146 0.14  0.15  0.158
##  9 New Mexico  0.206 0.186 0.224 0.245 0.195 0.185 0.213 0.194 0.23  0.195 0.209
## 10 Wisconsin   0.085 0.081 0.095 0.107 0.155 0.116 0.107 0.09  0.078 0.084 0.093
## # ... with 38 more rows, and 29 more variables: x1991 <dbl>, x1992 <dbl>,
## #   x1993 <dbl>, x1994 <dbl>, x1995 <dbl>, x1996 <dbl>, x1997 <dbl>,
## #   x1998 <dbl>, x1999 <dbl>, x2000 <dbl>, x2001 <dbl>, x2002 <dbl>,
## #   x2003 <dbl>, x2004 <dbl>, x2005 <dbl>, x2006 <dbl>, x2007 <dbl>,
## #   x2008 <dbl>, x2009 <dbl>, x2010 <dbl>, x2011 <dbl>, x2012 <dbl>,
## #   x2013 <dbl>, x2014 <dbl>, x2015 <dbl>, x2016 <dbl>, x2017 <dbl>,
## #   x2018 <dbl>, x2019 <dbl>
```

```
Poverty_test <- testing(Poverty_split)
dim(Poverty_test)
```

```
## [1] 21 41
```

```
Poverty_test
```

```
## # A tibble: 21 x 41
##    Elements    x1980 x1981 x1982 x1983 x1984 x1985 x1986 x1987 x1988 x1989 x1990
##    <chr>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 Single Mot~ 0.429 0.443 0.478 0.471 0.457 0.454 0.46  0.455 0.447 0.428 0.445
##  2 West        0.114 0.127 0.141 0.147 0.131 0.13  0.132 0.126 0.127 0.125 0.13
##  3 South       0.165 0.174 0.181 0.172 0.162 0.16  0.161 0.161 0.161 0.154 0.158
##  4 California  0.11  0.133 0.141 0.149 0.132 0.136 0.127 0.123 0.132 0.129 0.139
##  5 Connecticut 0.083 0.082 0.081 0.088 0.069 0.076 0.06  0.066 0.04  0.029 0.06
##  6 Delaware    0.118 0.123 0.115 0.086 0.103 0.114 0.124 0.066 0.086 0.1   0.069
##  7 Hawaii      0.085 0.113 0.132 0.129 0.093 0.107 0.107 0.088 0.111 0.113 0.11
##  8 Indiana     0.118 0.126 0.126 0.161 0.129 0.12  0.127 0.111 0.101 0.137 0.13
##  9 Kansas      0.094 0.117 0.104 0.136 0.107 0.138 0.111 0.092 0.081 0.108 0.103
## 10 Kentucky    0.193 0.193 0.162 0.179 0.191 0.194 0.177 0.173 0.176 0.161 0.173
## # ... with 11 more rows, and 29 more variables: x1991 <dbl>, x1992 <dbl>,
## #   x1993 <dbl>, x1994 <dbl>, x1995 <dbl>, x1996 <dbl>, x1997 <dbl>,
## #   x1998 <dbl>, x1999 <dbl>, x2000 <dbl>, x2001 <dbl>, x2002 <dbl>,
## #   x2003 <dbl>, x2004 <dbl>, x2005 <dbl>, x2006 <dbl>, x2007 <dbl>,
## #   x2008 <dbl>, x2009 <dbl>, x2010 <dbl>, x2011 <dbl>, x2012 <dbl>,
## #   x2013 <dbl>, x2014 <dbl>, x2015 <dbl>, x2016 <dbl>, x2017 <dbl>,
## #   x2018 <dbl>, x2019 <dbl>
```

```
Poverty_fold <- vfold_cv(Poverty_train, v= 10)
```

We are going to set up some recipes that will be used in modeling. For polynomial regression, I have set degree to 5, since there are many predictors and supplanting them with a degree is not optimal since high degree polynomials approach to overfitting and results do not show significant change. Also, we are going to take rates from the middle of the data, this is poverty rates in `x1999` to generate a polynomial to predict the final outcome.

### Model Building

There will be six machine learning models that will predict all poverty rates in `x2019` using different parameters, engines, and some will use a different recipe due to a change in the variables used. This is the case of Polynomial Regression and Regression Spline were only one variable will predict the outcome.

The steps for the modeling process are similar for all. First, a recipe will be set will corresponding variables and steps. Since the evaluating data only has numerical predictors, categorical steps are not required. Then, the model will use the recipe and a regression engine to use in a workflow and eventually fit the best results to the training set.

For some models, a grid will be created to tune the workflow and finally visualize the accuracy results of their performance following the numerical value derived from the rsq metric.

**First Model: Linear Regression** Let's create the recipe for this model, where variables will be scaled, centered and the principal component analysis step will deal with the high correlation we saw previously.

`Elements` is not included since it is a categorical variable that does not impact the data over the years.

```
lm_Poverty_recipe <- recipe(x2019 ~ x1980 + x1981 + x1982 + x1983 + x1984
                              + x1985 + x1986 + x1987 + x1988 + x1989 + x1990
                              + x1991 + x1992 + x1993 + x1994 + x1995 + x1996
                              + x1997 + x1998 + x1999 + x2000 + x2001 + x2002
                              + x2003 + x2004 + x2005 + x2006 + x2007 + x2008
                              + x2009 + x2010 + x2011 + x2012 + x2013 + x2014
                              + x2015 + x2016 + x2017 + x2018,
                            data= Poverty_train) %>%
  step_normalize(all_predictors()) %>%
  step_pca(all_predictors())
```

```
estimate_pca <- prep(lm_Poverty_recipe, training= Poverty_train)
tidy(estimate_pca, number= 2)
```

```
## # A tibble: 1,521 x 4
##    terms value component id
##    <chr> <dbl> <chr>     <chr>
##  1 x1980 0.156 PC1       pca_5WhVq
##  2 x1981 0.150 PC1       pca_5WhVq
##  3 x1982 0.159 PC1       pca_5WhVq
##  4 x1983 0.161 PC1       pca_5WhVq
##  5 x1984 0.159 PC1       pca_5WhVq
##  6 x1985 0.155 PC1       pca_5WhVq
##  7 x1986 0.153 PC1       pca_5WhVq
##  8 x1987 0.158 PC1       pca_5WhVq
##  9 x1988 0.161 PC1       pca_5WhVq
## 10 x1989 0.163 PC1       pca_5WhVq
## # ... with 1,511 more rows
```

There were a total of 26 principal components, each with all variables. This recipe will be used for other models so components are the same for those.

Following th recipe for the model, set up the regression model, workflow and fit, all on the training set.

```
lm_Poverty <- linear_reg() %>%
  set_mode("regression") %>%
  set_engine(("lm"))

lm_Poverty_wf <- workflow() %>%
```

```
  add_model(lm_Poverty) %>%
  add_recipe(lm_Poverty_recipe)

lm_Poverty_fit <- fit(lm_Poverty_wf, Poverty_train)
```

There are 5 principal component terms derived from the linear regression model.

```
lm_Poverty_fit %>%
  extract_fit_parsnip() %>%
  tidy()
```

```
## # A tibble: 6 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  0.101    0.00147     68.8   8.14e-45
## 2 PC1          0.00481  0.000252    19.1   2.56e-22
## 3 PC2         -0.00187  0.00143     -1.31  1.97e- 1
## 4 PC3          0.00787  0.00213      3.70  6.28e- 4
## 5 PC4         -0.00317  0.00258     -1.23  2.25e- 1
## 6 PC5          0.00109  0.00288      0.380 7.06e- 1
```

Let's find the predicted results of `x2019` and compare to the actual poverty rates in 2019.

```
Class_linear <- predict(lm_Poverty_fit, Poverty_train) %>%
  bind_cols(Poverty_train %>% select(x2019))
```

Now that predictions for all elements in 2019 are set, the best result for the "rsq" metric can determine the accuracy of the model.

```
linear_Regression <- Class_linear %>%
  ggplot(aes(x= .pred, y= x2019)) +
  geom_point(alpha= 0.6) +
  geom_abline(lty= 4) +
  coord_obs_pred() + labs(title= "Linear Regression", x= "Prediction", y= "2019" ) +
  theme(plot.title= element_text(size= 10),
        panel.background= element_rect(fill= "lightgoldenrodyellow", color= "black"))

Best_linear <- rsq(Class_linear, truth= x2019, estimate= .pred)

Best_linear
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.901
```

The model performed very well, meaning that 90% in the variations in the poverty rates from 1980 to 2018 are explained by 2019. This makes sense since the many predictors we have show a linear relationship with the outcome.

**Second Model: Ridge Regression** Ridge can be used for regression models since we will continue to use a linear regression model similar to the previous one.

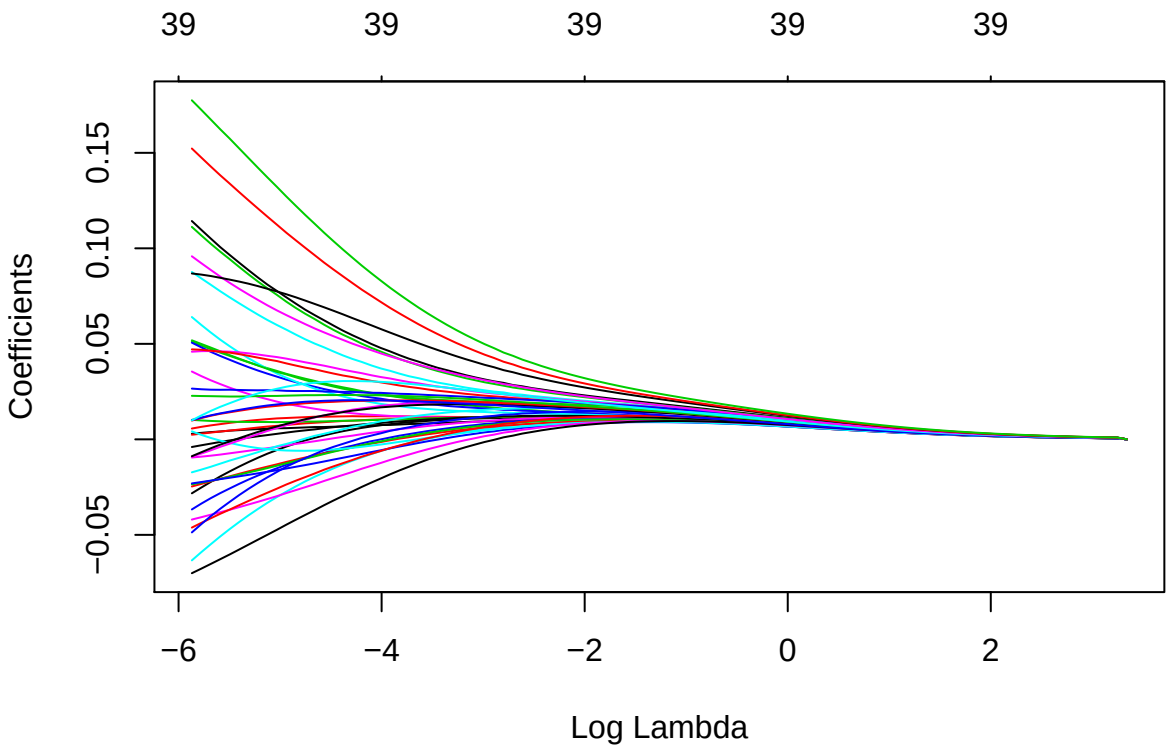
```
ridge_Poverty_recipe <- recipe(x2019 ~ x1980 + x1981 + x1982 + x1983 + x1984
                               + x1985 + x1986 + x1987 + x1988 + x1989 + x1990
                               + x1991 + x1992 + x1993 + x1994 + x1995 + x1996
                               + x1997 + x1998 + x1999 + x2000 + x2001 + x2002
```

```
                              + x2003 + x2004 + x2005 + x2006 + x2007 + x2008
                              + x2009 + x2010 + x2011 + x2012 + x2013 + x2014
                              + x2015 + x2016 + x2017 + x2018,
                              data= Poverty_train) %>%
  step_normalize(all_predictors()) %>%
  step_pca(all_predictors())
```

Before tuning hyper parameters, set penalty and mixture to 0.

```
ridge_Poverty <- linear_reg(penalty= 0, mixture= 0) %>%
  set_mode("regression") %>%
  set_engine("glmnet")

ridge_Poverty_fit <- fit(ridge_Poverty, x2019 ~ . -Elements, data= Poverty_train)
```

```
ridge_Poverty_fit %>%
  extract_fit_engine() %>%
  plot(xvar= "lambda")
```




This implies that the magnitude of the coefficients is regularized to zero with an increasing penalty.

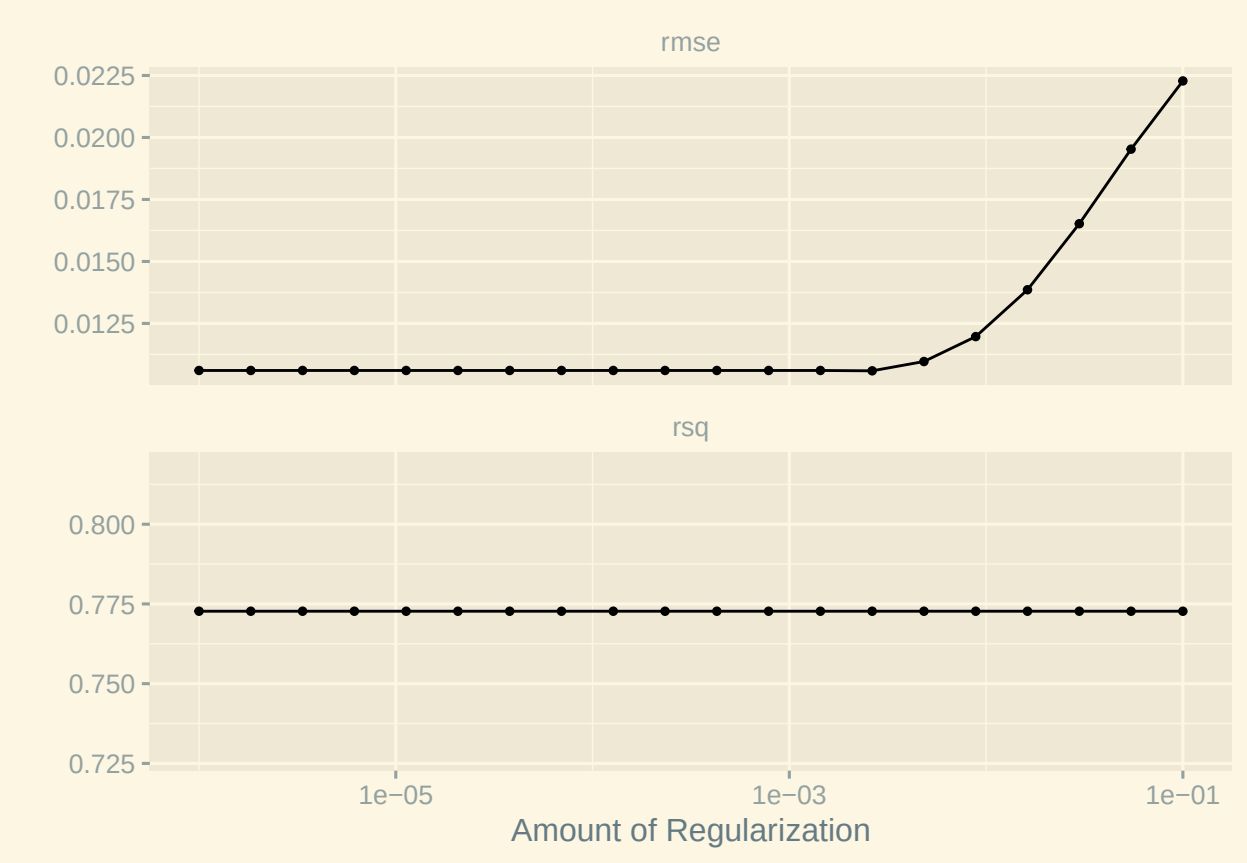
Next, let's tune the penalty parameter.

```
ridge_Poverty <- linear_reg(penalty= tune(), mixture= 0) %>%
  set_mode("regression") %>%
  set_engine("glmnet")

ridge_Poverty_wf <- workflow() %>%
```

```
  add_recipe(ridge_Poverty_recipe) %>%
  add_model(ridge_Poverty)

penalty_Poverty_Grid <- grid_regular(penalty(range= c(-6, -1)), levels= 20)
```

Now, create a grid of evenly spaced parameter values to obtain the penalty values.

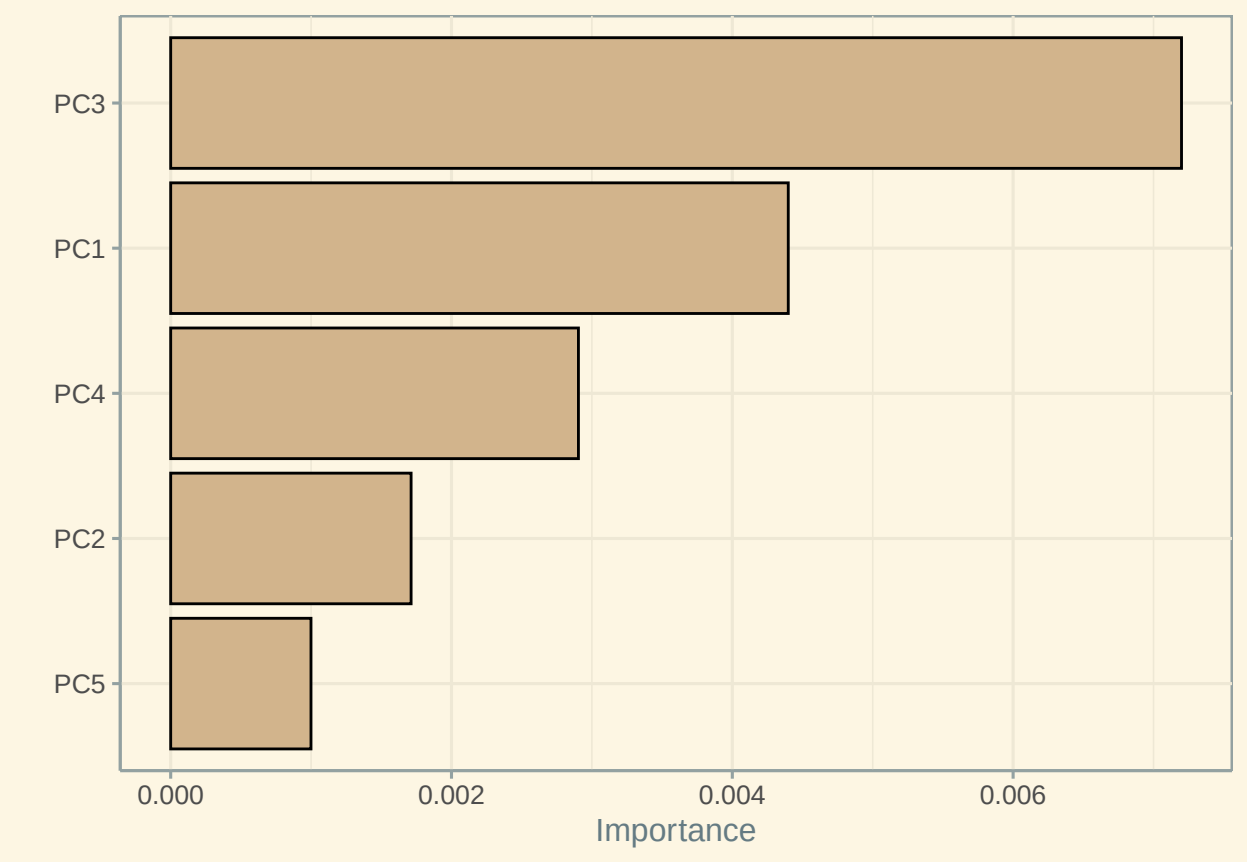
```
ridge_Poverty_tune <- tune_grid(ridge_Poverty_wf, resamples= Poverty_fold, grid= penalty_Poverty_Grid)

autoplot(ridge_Poverty_tune) + theme(axis.text.y= element_text(size= 8)) + theme_solarized_2()
```




The plots indicate that the amount of regularization affect the metric in different ways. For rmse, more regularization is more significant, while for rsq it stays constant.

Let's select the penalty that performed best and use it to finalize the workflow and fit it to predict the training set values.

```
Best_ridge_penalty <- select_best(ridge_Poverty_tune, metric= "rsq")

final_Poverty_ridge <- finalize_workflow(ridge_Poverty_wf, Best_ridge_penalty)

Class_ridge_fit <- fit(final_Poverty_ridge, data= Poverty_train)

Class_ridge_fit %>%
  extract_fit_engine() %>%
  vip(aesthetics = list(fill= "tan", color= "black")) + theme_solarized()
```




Principal Component 3 had the greatest importance when determined the prediction values.

```
ridge_Regression <- augment(Class_ridge_fit, new_data= Poverty_train) %>%
  ggplot(aes(x= .pred, y= x2019)) +
  geom_point(alpha= 0.6)+
  geom_abline(lty= 4) +
  coord_obs_pred() + labs(title= "Ridge Regression", x= "Prediction", y= "2019" ) +
  theme(plot.title= element_text(size= 10),
        panel.background= element_rect(fill= "lightgoldenrodyellow", color= "black"))

Best_ridge <- augment(Class_ridge_fit, new_data= Poverty_test) %>%
  rsq(truth= x2019, estimate= .pred)

Best_ridge
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.959
```

Ridge regression performed well with a 95% rsq estimate.

**Third Model: Polynomial Regression** Let's now choose one variable to predict the outcome using polynomial expansion. The variable `x1999` is in the middle of the data set, and I considered it will be useful since it contains the median information in the year range I'm evaluating. The variable will derive five polynomial degrees as `x1999^1`, `x1999^2`, `x1999^3`, `x1999^4`, and `x1999^5` that will attempt to predict `x2019`.

```
poly_Poverty_recipe <- recipe(x2019 ~ x1999, data= Poverty_train) %>%
  step_normalize(all_predictors()) %>%
  step_poly(x1999, degree= 5) %>%
  step_pca(all_predictors())
```

```
estimate_pca <- prep(poly_Poverty_recipe, training= Poverty_train)
tidy(estimate_pca, number= 3)
```

```
## # A tibble: 25 x 4
##    terms          value component id
##    <chr>          <dbl> <chr>     <chr>
##  1 x1999_poly_1  0      PC1       pca_2Jspy
##  2 x1999_poly_2 -0.398  PC1       pca_2Jspy
##  3 x1999_poly_3 -0.109  PC1       pca_2Jspy
##  4 x1999_poly_4  0.0572 PC1       pca_2Jspy
##  5 x1999_poly_5  0.909  PC1       pca_2Jspy
##  6 x1999_poly_1 -1      PC2       pca_2Jspy
##  7 x1999_poly_2  0      PC2       pca_2Jspy
##  8 x1999_poly_3  0      PC2       pca_2Jspy
##  9 x1999_poly_4  0      PC2       pca_2Jspy
## 10 x1999_poly_5  0      PC2       pca_2Jspy
## # ... with 15 more rows
```

There will be five polynomial values for each of the five principal components.

Next, create the model with a linear regression engine, set the workflow, and fit it to predict the outcome.

```
poly_Poverty <- linear_reg() %>%
  set_mode("regression") %>%
  set_engine("lm")

poly_Poverty_wf <- workflow() %>%
  add_model(poly_Poverty) %>%
  add_recipe(poly_Poverty_recipe)

poly_Poverty_fit <- fit(poly_Poverty_wf, data= Poverty)
```

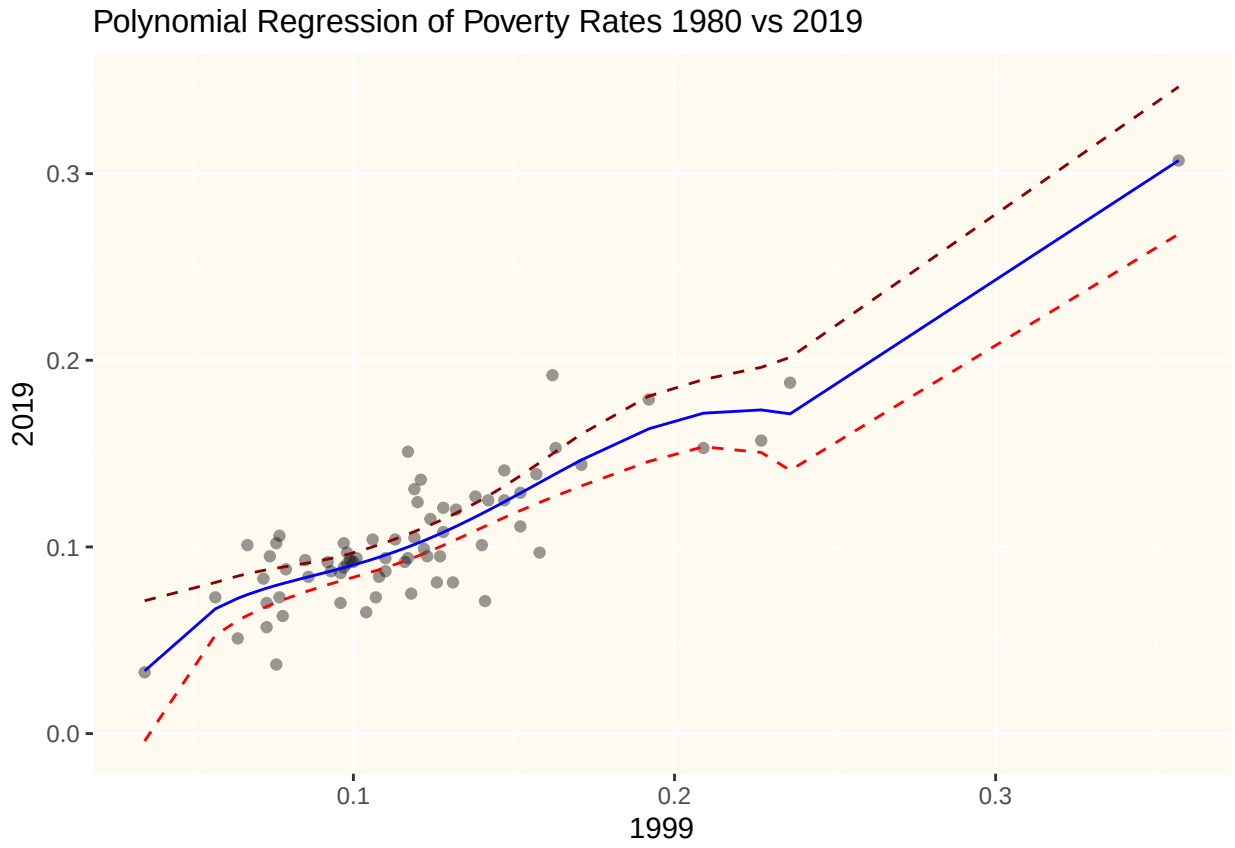
Let's take a look at the predictions of each `x2019` element and the bounds.

```
Range1999 <- Poverty[,21]

reg_lines <- bind_cols(augment(poly_Poverty_fit, new_data= Range1999),
                       predict(poly_Poverty_fit, new_data= Range1999, type= "conf_int"))
reg_lines
```

```
## # A tibble: 69 x 4
##     x1999  .pred .pred_lower .pred_upper
##     <dbl>  <dbl>       <dbl>       <dbl>
##  1 0.119  0.101      0.0944       0.108
##  2 0.171  0.147      0.133        0.161
##  3 0.101  0.0908     0.0843       0.0974
##  4 0.0970 0.0889     0.0823       0.0955
##  5 0.035  0.0336    -0.00398      0.0712
##  6 0.064  0.0725     0.0606       0.0844
##  7 0.092  0.0866     0.0798       0.0935
##  8 0.163  0.139      0.127        0.151
```

```
##  9 0.357  0.307      0.268        0.347
## 10 0.098  0.0894     0.0828       0.0959
## # ... with 59 more rows
```

```
Poverty %>%
  ggplot(aes(x1999, x2019)) +
  geom_point(alpha= 0.4) +
  geom_line(aes(y=.pred), color= "blue",
            data= reg_lines) +
  geom_line(aes(y= .pred_upper), data= reg_lines,
            linetype= "dashed", color= "darkred") +
  geom_line(aes(y= .pred_lower), data= reg_lines,
            linetype= "dashed", color= "red") +
  labs(title= "Polynomial Regression of Poverty Rates 1980 vs 2019", x= "1999", y= "2019" ) +
  theme(plot.title= element_text(size= 12),
        panel.background= element_rect(fill= "floralwhite"))
```




We can see that the predicted ranges converge to the cluster of poverty rates while the confidence bands separate when there is limited data points. This analysis implies that having more data can state a smaller confidence interval, and thus a more accurate approximation.

Finally, select the best performing prediction to assess performance.

```
Class_poly <- predict(poly_Poverty_fit, Poverty_train) %>%
  bind_cols(Poverty_train %>% select(x2019))

Best_poly <- rsq(Class_poly, truth= x2019, estimate= .pred)
```

```
Best_poly
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.705
```

Polynomial regression did not perform as well as the previous models. This means that only using one predictor, x1999 in this case, is not as optimal as evaluating most at the same time. Only 70% of the dependent variables are explained by the outcome, not efficient.

```
poly_Regression <- Class_poly %>%
  ggplot(aes(x= .pred, y= x2019)) +
  geom_point(alpha= 0.4) +
  geom_abline(lty= 4) +
  coord_obs_pred() +
  labs(title= "Polynomial Regression", x= "Prediction", y= "2019" ) +
  theme(plot.title= element_text(size= 10),
        panel.background= element_rect(fill= "lightgoldenrodyellow", color= "black"))
```

**Fourth Model: Regression Spline** We can also use the polynomial regression to develop a spline approximation. I will also use x1999 to compare results with the polynomial regression model.

```
spline_Poverty_recipe <- recipe(x2019 ~ x1999, data= Poverty) %>%
  step_bs(x1999, options= list(knots= 5)) %>%
  step_pca(all_predictors())
```

Take the same model from polynomial regression to create the workflow and fit.

```
spline_Poverty_wf <- workflow() %>%
  add_model(poly_Poverty) %>%
  add_recipe(spline_Poverty_recipe)

spline_Poverty_fit <- fit(spline_Poverty_wf, data= Poverty)
```

Let's see the spline predicted and the bounds for the confidence interval.

```
Range1999 <- Poverty[,21]

reg_lines <- bind_cols(augment(spline_Poverty_fit, new_data= Range1999),
                       predict(spline_Poverty_fit, new_data= Range1999, type= "conf_int"))
reg_lines
```

```
## # A tibble: 69 x 4
##     x1999  .pred .pred_lower .pred_upper
##     <dbl>  <dbl>       <dbl>       <dbl>
##  1 0.119  0.104       0.0984      0.110
##  2 0.171  0.138       0.129       0.148
##  3 0.101  0.0926      0.0868      0.0984
##  4 0.0970 0.0900      0.0842      0.0958
##  5 0.035  0.0468      0.0224      0.0713
##  6 0.064  0.0677      0.0571      0.0783
##  7 0.092  0.0867      0.0807      0.0926
##  8 0.163  0.133       0.125       0.141
##  9 0.357  0.306       0.266       0.346
## 10 0.098  0.0906      0.0848      0.0964
## # ... with 59 more rows
```

```
Poverty %>%
  ggplot(aes(x1999, x2019)) +
  geom_point(alpha= 0.4) +
  geom_line(aes(y=.pred), color= "blue",
            data= reg_lines) +
  geom_line(aes(y= .pred_upper), data= reg_lines,
            linetype= "dashed", color= "darkred") +
  geom_line(aes(y= .pred_lower), data= reg_lines,
            linetype= "dashed", color= "red") +
  labs(title= "Regression Spline of Poverty Rates", x= "1999", y= "2019" ) +
  theme(plot.title= element_text(size= 12),
        panel.background= element_rect(fill= "floralwhite"))
```


Similarly to polynomial regression, the best approximations occur where there is more data. The intervals become broad as the number of observations decreases, reducing optimal predictions for those elements.

Let's select the best spline to use later for the testing set.

```
Class_spline <- predict(spline_Poverty_fit, Poverty_train) %>%
  bind_cols(Poverty_train %>% select(x2019))

Best_spline <- rsq(Class_spline, truth= x2019, estimate= .pred)

Best_spline
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
```

```
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.703
```

Again, performance is similar to polynomial regression. With a 70% rsq estimate not all variation can be explained by `x2019`. Thus, potentially not an optimal model for the testing set.

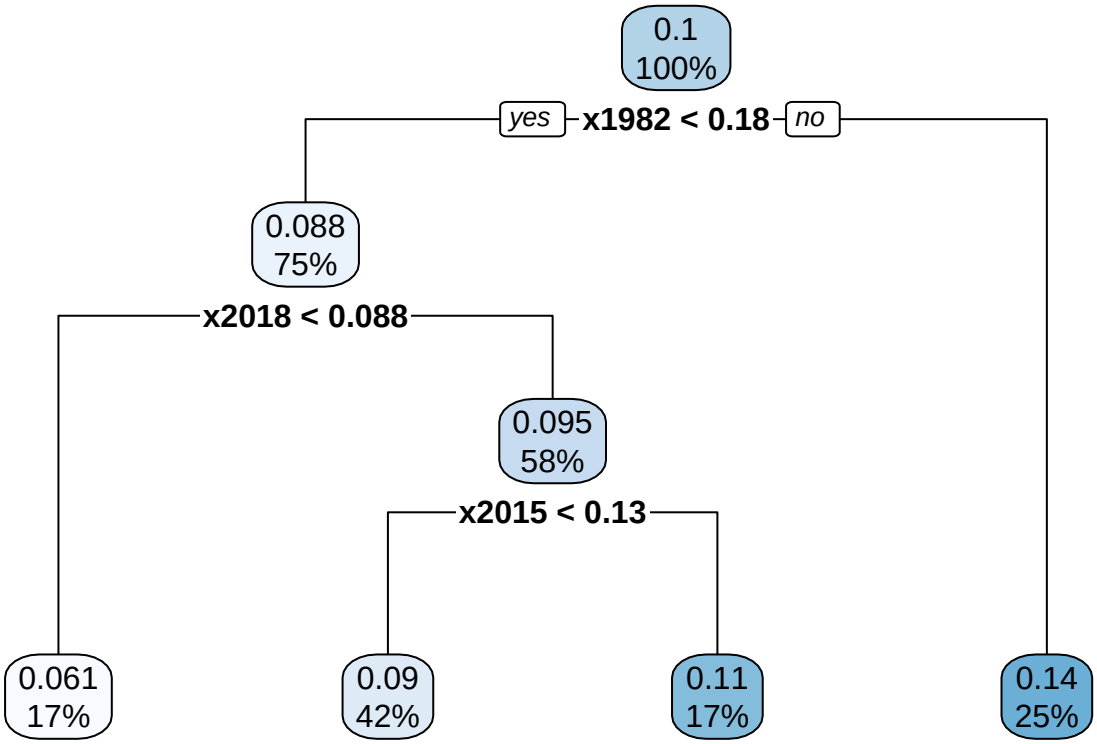
```
spline_Regression <- Class_spline %>%
  ggplot(aes(x= .pred, y= x2019)) +
  geom_point(alpha= 0.4) +
  geom_abline(lty= 4) +
  coord_obs_pred() + labs(title= "Regression Spline", x= "Prediction", y= "2019" ) +
  theme(plot.title= element_text(size= 10),
        panel.background= element_rect(fill= "lightgoldenrodyellow", color= "black"))
```

**Fifth Model: Regression Tree** The response for this project is continuous, thus performing a regression tree is an appropriate model for prediction.

```
tree_Poverty_recipe <- recipe(x2019 ~ x1980 + x1981 + x1982 + x1983 + x1984
                              + x1985 + x1986 + x1987 + x1988 + x1989 + x1990
                              + x1991 + x1992 + x1993 + x1994 + x1995 + x1996
                              + x1997 + x1998 + x1999 + x2000 + x2001 + x2002
                              + x2003 + x2004 + x2005 + x2006 + x2007 + x2008
                              + x2009 + x2010 + x2011 + x2012 + x2013 + x2014
                              + x2015 + x2016 + x2017 + x2018,
                              data= Poverty_train) %>%
  step_normalize(all_predictors()) %>%
  step_pca(all_predictors())
```

Set the model with the rpart engine and fit with the continuous variables.

```
tree_Poverty <- decision_tree() %>%
  set_mode("regression") %>%
  set_engine("rpart")

tree_Poverty_fit <- fit(tree_Poverty, x2019 ~ . -Elements, Poverty_train)

tree_Poverty_fit %>%
  extract_fit_engine() %>%
  rpart.plot()
```

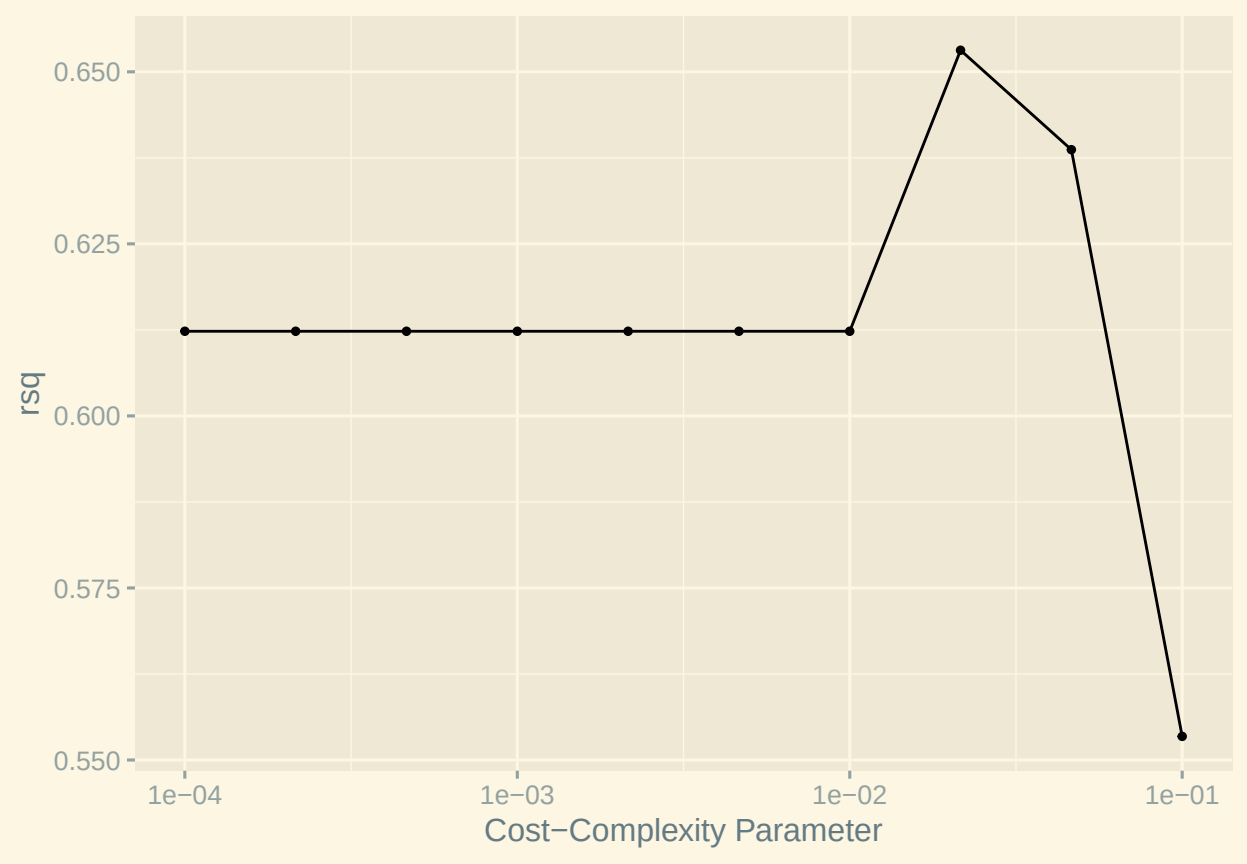



The decision tree shows the poverty rates of each percentage of the selected populations. We can see that `Elements` have the lowest poverty rates in recent years.

Let's tune cost_complexity and perform a grid to find the best regression tree.

```
tree_Poverty_wf <- workflow() %>%
  add_model(tree_Poverty %>% set_args(cost_complexity= tune())) %>%
  add_recipe(tree_Poverty_recipe)

tree_Poverty_Grid <- grid_regular(cost_complexity(range= c(-4, -1)), levels= 10)
```

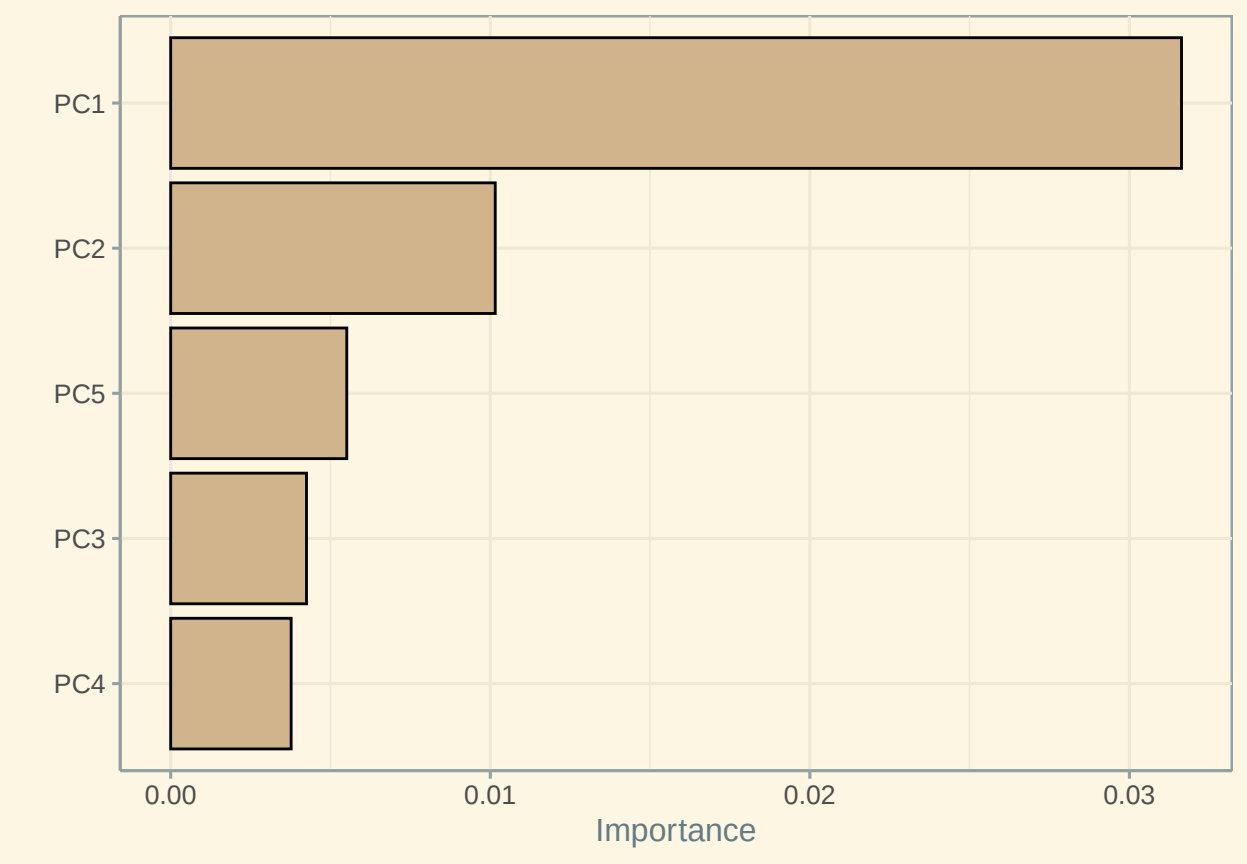
```
tree_Poverty_tune <- tune_grid(tree_Poverty_wf, resamples= Poverty_fold,
                               grid= tree_Poverty_Grid, metrics= metric_set(rsq))

autoplot(tree_Poverty_tune) + theme(axis.text.y= element_text(size= 8)) + theme_solarized_2()
```




rsq decreases to 0.550 when cost-complexity is higher.

Finalize the workflow with the best tree and collect results.

```
best_tree_penalty <- select_best(tree_Poverty_tune, metric= "rsq")

final_Poverty_tree <- finalize_workflow(tree_Poverty_wf, best_tree_penalty)

Class_tree_fit <- fit(final_Poverty_tree, data= Poverty_train)

Class_tree_fit %>%
  extract_fit_engine() %>%
  vip(aesthetics = list(fill= "tan", color= "black")) + theme_solarized()
```




The variance importance plot shows that Principal Component 1 had the greatest importance when performing the best tree.

```
tree_Regression <- augment(Class_tree_fit, new_data= Poverty_train) %>%
  ggplot(aes(x= .pred, y= x2019)) +
  geom_point(alpha= 0.6) +
  geom_abline(lty= 4) +
  coord_obs_pred() + geom_jitter() +
  labs(title= "Regression Tree", x= "Prediction", y= "2019" ) +
  theme(plot.title= element_text(size= 10),
        panel.background= element_rect(fill= "lightgoldenrodyellow", color= "black"))


Best_tree <- augment(Class_tree_fit, new_data= Poverty_train) %>%
  rsq(truth= x2019, estimate= .pred)

Best_tree
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.740
```

The regression tree performed sufficiently, but it is not the most optimal so far. A rsq of 73% may show an important error range when predicting the outcome values.

**Sixth Model: Random Forest** The final model will be a random forest, using the common recipe for most models.

```
forest_Poverty_recipe <- recipe(x2019 ~ x1980 + x1981 + x1982 + x1983 + x1984
                                + x1985 + x1986 + x1987 + x1988 + x1989
                                + x1990+ x1991 + x1992 + x1993 + x1994
                                + x1995 + x1996 + x1997 + x1998 + x1999
                                + x2000 + x2001 + x2002 + x2003 + x2004
                                + x2005 + x2006 + x2007 + x2008 + x2009
                                + x2010 + x2011 + x2012 + x2013 + x2014
                                + x2015 + x2016 + x2017 + x2018,
                                data= Poverty_train) %>%
  step_normalize(all_predictors()) %>%
  step_pca(all_predictors())
```

The model represents bagging, since all predictors will be evaluated. Fit to continuous variables.

```
forest_Poverty <- rand_forest(mtry= .cols()) %>%
  set_mode("regression") %>%
  set_engine("ranger", importance= "impurity")

forest_Poverty_fit <- fit(forest_Poverty, x2019 ~ . -Elements, data= Poverty_train)
```

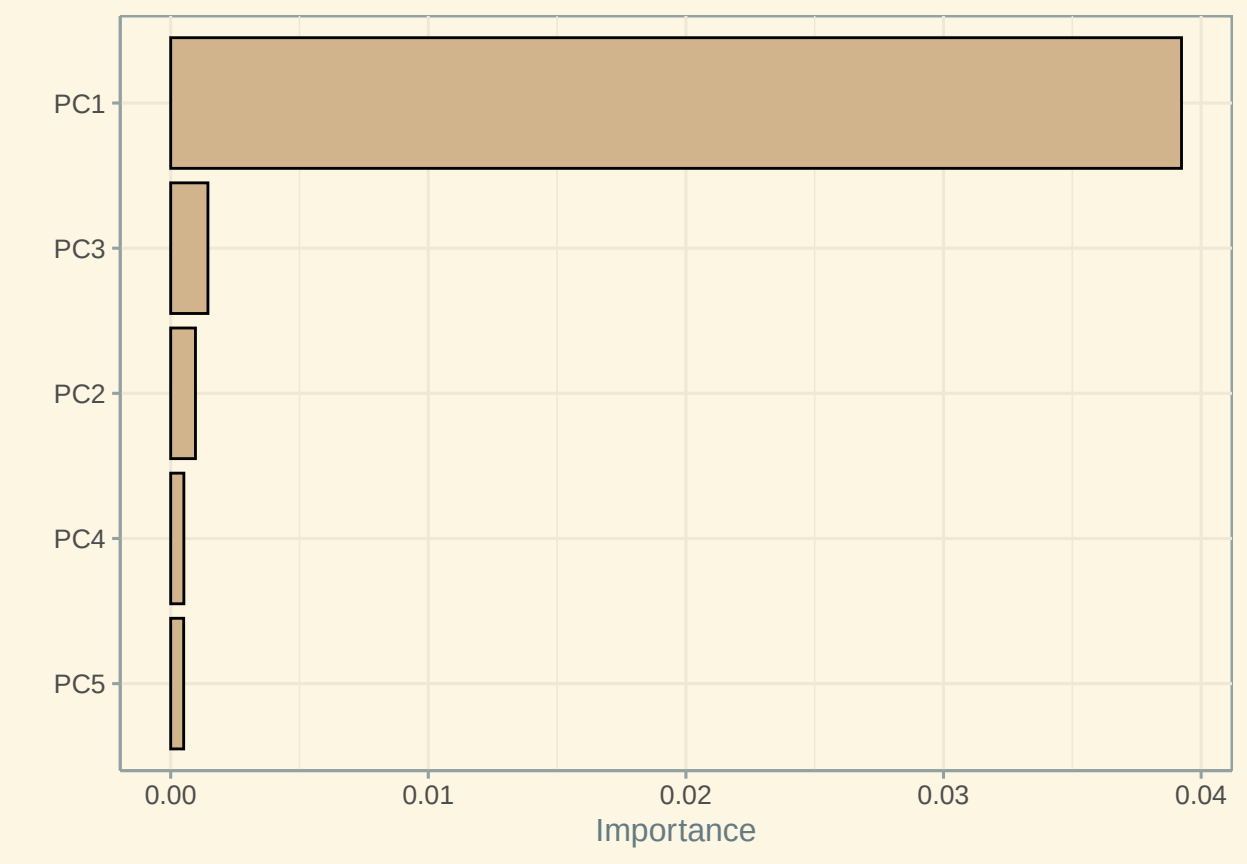
Set the model to the grid with the corresponding ranges.

```
forest_Poverty_wf <- workflow() %>%
  add_model(forest_Poverty) %>%
  add_recipe(forest_Poverty_recipe)

forest_Poverty_Grid <- grid_regular(mtry(range= c(1,38)), trees(range= c(200, 900)),
                                    min_n(range= c(4, 20)), levels= 10)
```

```
forest_Poverty_tune <- tune_grid(forest_Poverty_wf, resamples= Poverty_fold,
                                 grid= forest_Poverty_Grid, metrics= metric_set(rsq))
```

Select the best random forest and finalize the workflow. Collect results.

```
best_forest_penalty <- select_best(forest_Poverty_tune, metric= "rsq")

final_Poverty_forest <- finalize_workflow(forest_Poverty_wf, best_forest_penalty)

Class_forest_fit <- fit(final_Poverty_forest, data= Poverty_train)

Class_forest_fit %>%
  extract_fit_engine() %>%
  vip(aesthetics = list(fill= "tan", color= "black")) + theme_solarized()
```




Principal Component 1 has a significantly greater importance compared to the other components.

```
forest_Regression <- augment(forest_Poverty_fit, new_data= Poverty_train) %>%
  ggplot(aes(x= .pred, y= x2019)) +
  geom_point(alpha= 0.6) +
  geom_abline(lty= 4) +
  coord_obs_pred() +
  labs(title= "Random Forest", x= "Prediction", y= "2019" ) +
  theme(plot.title= element_text(size= 10),
        panel.background= element_rect(fill= "lightgoldenrodyellow", color= "black"))

Best_forest <- augment(Class_forest_fit, new_data = Poverty_train) %>%
  rsq(truth= x2019, estimate= .pred)

Best_forest
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.967
```

It seems that we have a winner! The random forest model has an rsq of 96% in the training set. This is a great result since it means that most of the variation in the variables can be determined by `x2019`.

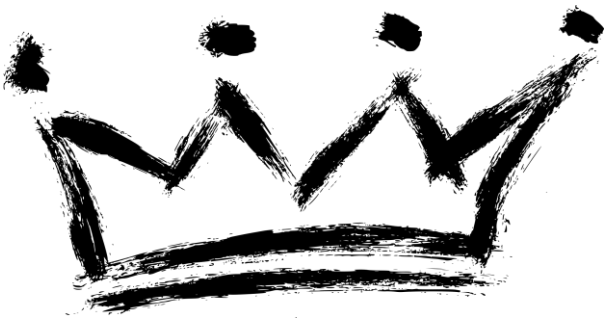
### Accuracy of Models Comparison

Now that we have performed all models on the training set, let's recapitulate and compare results.

```
Accuracy <- c(Best_linear$.estimate, Best_ridge$.estimate, Best_poly$.estimate,
              Best_spline$.estimate,  Best_tree$.estimate, Best_forest$.estimate)

Model <- c("Linear Regression", "Ridge Regression", "Polynomial Regression",
           "Regression Spline", "Regression Trees", "Random Forest")

TotalResults <- tibble(Model= Model, Accuracy= Accuracy)

TotalResults %>%
  arrange(-Accuracy)
```

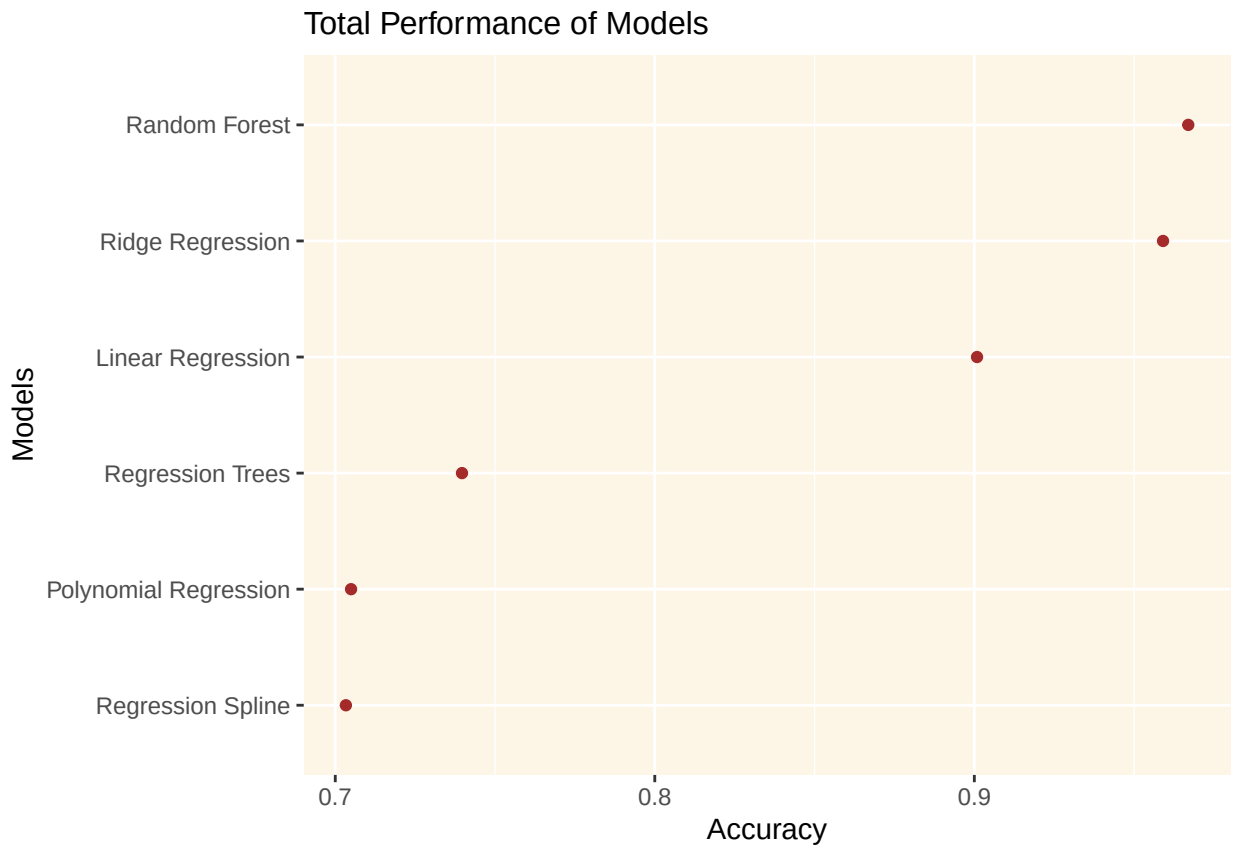
```
## # A tibble: 6 x 2
##   Model                 Accuracy
##   <chr>                    <dbl>
## 1 Random Forest            0.967
## 2 Ridge Regression         0.959
## 3 Linear Regression        0.901
## 4 Regression Trees         0.740
## 5 Polynomial Regression    0.705
## 6 Regression Spline        0.703
```

Random Forest and Ridge had a really close prediction, but Random Forest took the crown.



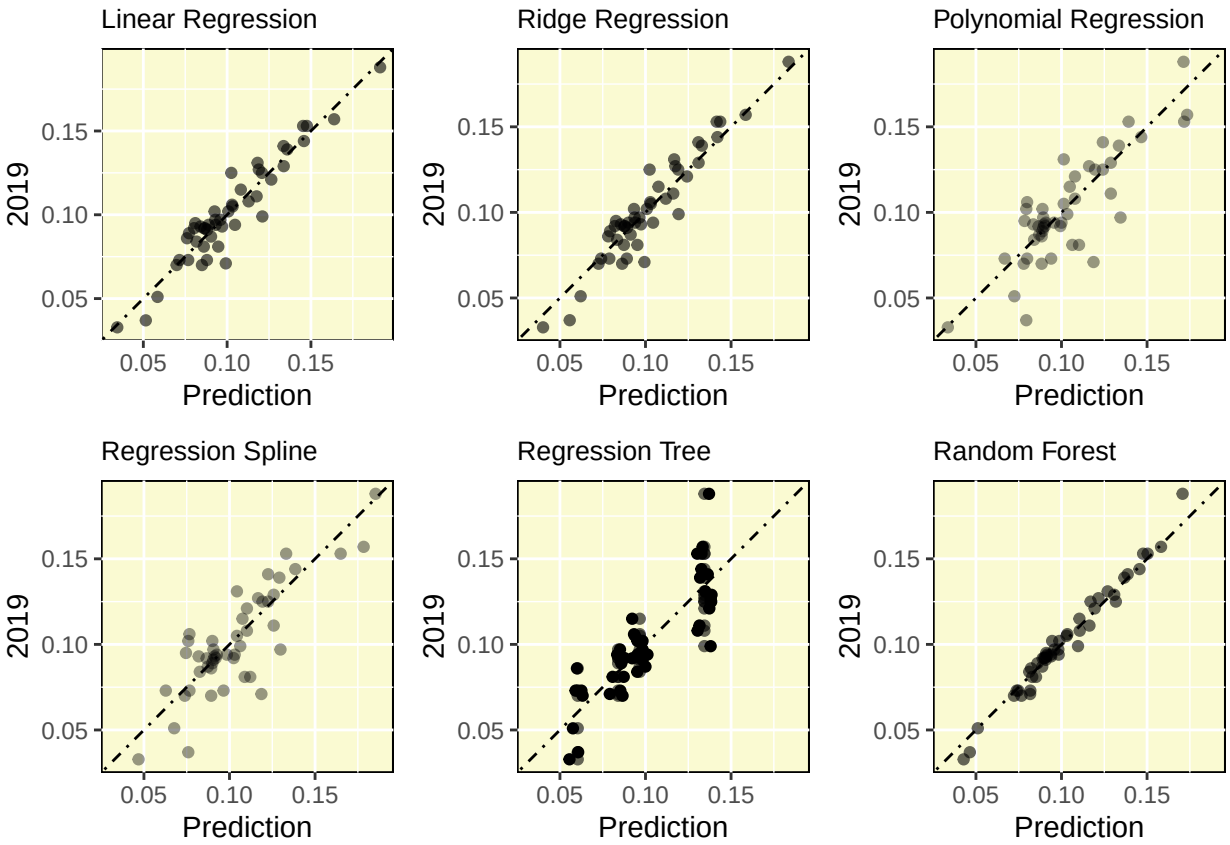
Let's visualize how each model performed.

```
Models <- factor(TotalResults$Model, levels= TotalResults$Model[order(TotalResults$Accuracy)])

Total_plot <- ggplot(TotalResults, aes(x= Models, y= Accuracy)) + geom_point(colour= "brown") +
  labs(title= "Total Performance of Models", x= "Models", y= "Accuracy" ) +
  theme(plot.title= element_text(size= 12),
        panel.background= element_rect(fill= "oldlace")) + coord_flip()

Total_plot
```




Most models performed very well in the regression of the training data. However, Random Forest appears to have the greatest accuracy when predicting the model. A 0.9669481 is a great approximation to predict the outcome in the testing data.

```
plot_grid(linear_Regression, ridge_Regression, poly_Regression,
          spline_Regression, tree_Regression, forest_Regression)
```




We can see that Random Forest predicted each observation more accurately. Most points form a straight line!

Now, let's choose random forest to predict the outcome on the testing data.

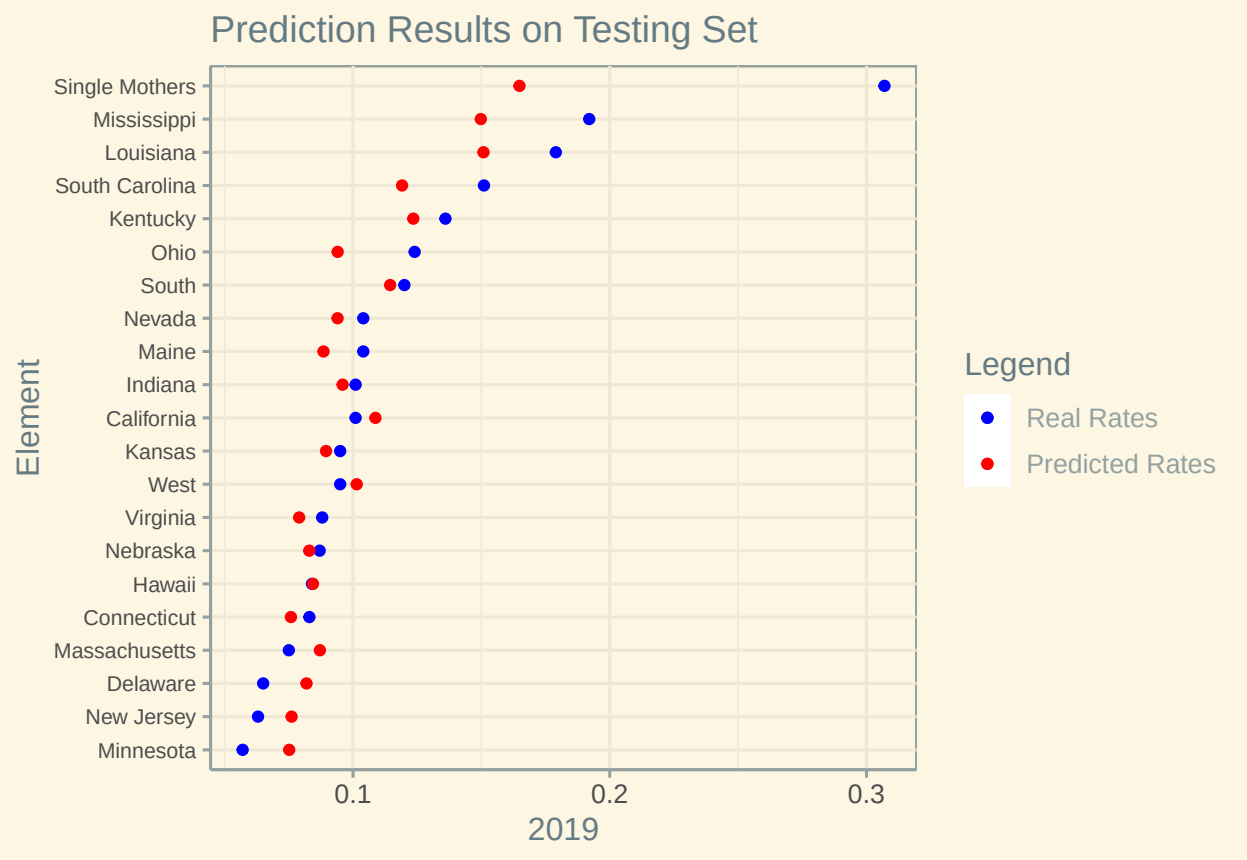
```
predict_Poverty <- predict(Class_forest_fit, new_data= Poverty_test)

predict_Compare <- predict_Poverty %>%
  bind_cols(Poverty_test[, c("Elements", "x2019")])

predict_Compare
```

```
## # A tibble: 21 x 3
##     .pred Elements       x2019
##     <dbl> <chr>          <dbl>
##  1 0.165  Single Mothers 0.307
##  2 0.101  West           0.095
##  3 0.114  South          0.12
##  4 0.109  California     0.101
##  5 0.0758 Connecticut    0.083
##  6 0.0819 Delaware       0.065
##  7 0.0844 Hawaii         0.084
##  8 0.0959 Indiana        0.101
##  9 0.0895 Kansas         0.095
## 10 0.123  Kentucky       0.136
## # ... with 11 more rows
```

It appears that most observations were closely predicted! However, `Single Mothers` is the exception since that element has had significantly higher poverty rates during the period in this assessment.

Let's visualize how close the predictions were to the actual poverty rates in 2019.

```
CompareOrder <- factor(predict_Compare$Elements,
                       levels= predict_Compare$Elements[order(predict_Compare$x2019)])

Predict2019 <- ggplot(predict_Compare, aes(x= CompareOrder, y= x2019)) +
  labs(title= "Prediction Results on Testing Set", x= "Element", y= "2019", color= "Legend") +
  geom_point(data= predict_Compare, mapping= aes(x= CompareOrder, y= x2019, color= "blue")) +
  geom_point(data= predict_Compare, mapping= aes(x= CompareOrder, y= .pred, color= "red")) +
  theme_solarized()+ theme(axis.text.y= element_text(size= 8)) +
  coord_flip() + scale_color_manual(values = c("blue", "red"),
                                    labels = c("Real Rates", "Predicted Rates")) +
  scale_shape_identity()

Predict2019
```




It seems that most predictions were really close, `Hawaii`'s rates were almost identical whereas `Single Mothers` was the farthest prediction.

Finally, let's see the overall performance of the random forest on the testing set.

```
rsq_Poverty <- augment(Class_forest_fit, new_data= Poverty_test) %>%
  rsq(x2019, estimate= .pred) %>%
  select(.estimate)

rsq_Poverty
```

```
## # A tibble: 1 x 1
```

```
##   .estimate
##       <dbl>
## 1     0.845
```

A 0.8449812 is a good estimate to evaluate a successful prediction. Most of the error was given to only one observation as seen previously, and that might be the reason of the drop from the training accuracy.

### Results Interpretation

Throughout the project we have seen how multiple regression models have attempted to predict poverty rates in 2019 in the United States. These results support the EDA presented in the initial steps of this report since the relationships between variables and outcome hold. All models predicted the outcome fairly well, but two of them were really close to finding a complete dependence of the variables and the outcome. These models were Random Forest and Ridge Regression with 0.9669481 and 0.9590533, respectively. Utilizing hyper parameter tuning could have improved the results since most of those models performed better. Thus, I utilized Random Forest to predict the outcome on the testing data.

Performing this analysis showed interesting results through visualization and quantitative results from the models. In the EDA, for example, I found poverty rate trends in the United States. Even though there was a decrease in poverty for all communities in the forty-year period, most communities have similar positions today relative to 1980. This is the case of single mothers, black communities, the south, and Mississippi, elements that continue to have the lowest rates in their respective category. These were the facts I found interesting in this project and hope it can be used in future research assessments.

### Bibliographic References

"Labor + Economics." *Recruiting News Network*, https://www.recruitingnewsnetwork.com/category/labor-economics.

"Percent of People in Poverty." *USAFACTS*, https://usafacts.org/data/topics/people-society/poverty/poverty-measures/poverty-rate-of-all-persons/.

Rezaian, Lachin. "America Struggles Over Poverty." *Mehr News Agency*, 13 May 2015, https://en.mehrnews.com/photo/107198/America-struggles-over-poverty.