Dimensionality Reduction and Feature Importance/Selection Applied to NYCgov Poverty Measure Data 2018

Le' Sean Roberts (NYC Office of Management and Budget College Aide)

Abstract

Dimensionality reduction and feature importance/selection techniques play a critical role in analyzing complex datasets, particularly when dealing with socioeconomic indicators such as poverty rates. In this study, we explore the application of dimensionality reduction methods and feature selection techniques to the NYCgov Poverty Measure dataset. By leveraging these techniques, we aim to uncover the most influential factors contributing to poverty in New York City while reducing the dimensionality of the dataset. This research sheds light on the key socioeconomic determinants of poverty in one of the world's largest and most diverse urban areas, providing valuable insights for policymakers, researchers, and practitioners striving to address poverty-related challenges.

Introduction

Understanding and addressing poverty are central concerns for policymakers and researchers worldwide, particularly in large urban areas like New York City (NYC). The NYCgov Poverty Measure provides a comprehensive framework for assessing poverty that accounts for the city's unique socioeconomic landscape, including its high cost of living and diverse population. However, analyzing the NYCgov Poverty Measure dataset poses significant challenges due to its high dimensionality and complex interrelationships among variables.

Dimensionality reduction techniques offer a powerful approach to address these challenges by transforming high-dimensional data into a lower-dimensional representation while preserving essential information. Similarly, feature importance/selection methods help identify the most relevant variables contributing to the phenomenon under study, facilitating a more focused analysis. Feature selection is a critical step in the machine learning pipeline, where the goal is to identify the most relevant features (or variables) that contribute the most to the predictive performance of a model while reducing overfitting and computational complexity.

In this study, we investigate the application of dimensionality reduction and feature importance/selection techniques to the NYCgov Poverty Measure dataset. **Note: the particular data set is NYCgov Poverty Measure data 2018. There is no assumption of such data characterising the current standing of New York City residents.** Such data however serves as a good resource to practice dimensionality reduction and feature importance/selection techniques. By reducing the dimensionality of the dataset and identifying the most influential features, we aim to gain deeper insights into the socioeconomic factors driving poverty in NYC. Specifically, we explore techniques such as Principal Component Analysis (PCA), and feature selection algorithms like Boruta and entropy-based feature selection algorithms with a sparse matrix support (FSelectorRccp).

Through this analysis, we seek to uncover the underlying structure of the NYCgov Poverty Measure data, identify key determinants of poverty in NYC, and provide actionable insights for policymakers, researchers, and stakeholders. By leveraging advanced data analysis techniques, we aim to contribute to the ongoing efforts to address poverty and promote socioeconomic equity in one of the world's most dynamic urban environments.

Following the dimensionality reduction or feature importance/selection process, predictive modelling or machine learning methods can be applied; whether it be classification, regression or ensemble methods.

Motivation

The data set consist of 61 columns (to be treated as variables) and general 68, 273 instances (or observations) for each column. However, for some columns there may cases of unattainable observes in particular rows. Hence, data set is likely to be reduced in size after data cleaning. Furthermore, treating 61 variables with correlation measures and correlation heatmaps is not a practical task because analysis of the measure visually is tedious with such large quantity of variables. Namely 61 by 61 is not practical. Of consequence, more abstract techniques are required involving data reduction with much loss of character. As well, the same issue with descriptive statistics, skewness, kurtosis (and the mode when dealing with categorical or ordinal measures) for 61 variables.

Preliminary Data Cleaning

```r
library(readr)
NYCgov_Poverty_Measure_Data_2018_20240326 <- read_csv("NYCgov_Poverty_Measure_Data__2018__202
```

```r
library(tidyverse)
glimpse(NYCgov_Poverty_Measure_Data_2018_20240326)
```

```
Rows: 68,273
Columns: 61
$ SERIALNO          <dbl> 1, 16, 16, 47, 47, 55, 55, 55, 194, 218, 218, 398, 39~
$ SPORDER           <dbl> 1, 1, 2, 1, 2, 1, 2, 3, 1, 1, 2, 1, 2, 1, 2, 3, 1, 2,~
$ PWGTP             <dbl> 95, 181, 210, 62, 71, 430, 466, 453, 201, 55, 60, 119~
$ WGTP              <dbl> 95, 181, 181, 62, 62, 430, 430, 430, 201, 56, 56, 119~
$ AGEP              <dbl> 31, 63, 65, 68, 63, 29, 25, 32, 76, 46, 49, 61, 66, 4~
$ CIT               <dbl> 1, 1, 1, 4, 4, 5, 5, 1, 4, 5, 5, 4, 4, 1, 4, 1, 5, 5,~
$ REL               <dbl> 0, 0, 1, 0, 1, 0, 1, 15, 0, 0, 1, 0, 1, 0, 1, 2, 0, 1~
$ SCH               <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1,~
$ SCHG              <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 15, 0~
$ SCHL              <dbl> 22, 22, 19, 1, 1, 21, 21, 16, 21, 17, 20, 13, 17, 19,~
$ SEX               <dbl> 2, 2, 1, 1, 2, 2, 1, 1, 1, 2, 1, 1, 2, 2, 1, 1, 1, 2,~
$ ESR               <dbl> 1, 1, 6, 1, 1, 6, 1, 1, 6, 1, 1, 1, 6, 1, 1, NA, 1, 6~
$ LANX              <dbl> 2, 2, 2, 1, 1, 1, 1, 1, 2, 1, 1, 2, 2, 2, 2, 2, 1, 1,~
$ ENG               <dbl> NA, NA, NA, 2, 1, 3, 1, 1, NA, 4, 2, NA, NA, NA, NA, ~
$ MSP               <dbl> 6, 1, 1, 1, 1, 1, 1, 6, 3, 1, 1, 1, 1, 1, 1, NA, 1, 1~
$ MAR               <dbl> 5, 1, 1, 1, 1, 1, 1, 5, 2, 1, 1, 1, 1, 1, 1, 5, 1, 1,~
$ WKW               <dbl> 1, 1, NA, 1, 1, 1, 1, 1, NA, 1, 4, 1, NA, 1, 1, NA, 1~
$ WKHP              <dbl> 60, 60, 0, 40, 40, 40, 40, 40, 0, 35, 40, 40, 0, 40, ~
$ DIS               <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2,~
$ JWTR              <dbl> 12, 1, NA, 10, 10, NA, 1, 1, NA, 4, 4, 4, NA, 4, 1, N~
$ NP                <dbl> 1, 2, 2, 2, 2, 3, 3, 3, 1, 2, 2, 2, 2, 3, 3, 3, 4, 4,~
$ TEN               <dbl> 2, 1, 1, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 1, 1, 1, 3, 3,~
$ HHT               <dbl> 6, 1, 1, 1, 1, 1, 1, 1, 4, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
$ AgeCateg          <dbl> 2, 2, 3, 3, 2, 2, 2, 2, 3, 2, 2, 2, 3, 2, 2, 1, 2, 2,~
$ Boro              <dbl> 2, 3, 3, 2, 2, 4, 4, 4, 2, 2, 2, 3, 3, 3, 3, 3, 2, 2,~
$ CitizenStatus     <dbl> 1, 1, 1, 2, 2, 3, 3, 1, 2, 3, 3, 2, 2, 1, 2, 1, 3, 3,~
$ EducAttain        <dbl> 4, 4, 3, 1, 1, 4, 4, 2, 4, 2, 3, 1, 2, 3, 1, 1, 2, 3,~
$ EST_Childcare     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ EST_Commuting     <dbl> 1722.000, 3840.842, 3840.842, 0.000, 0.000, 4954.167,~
$ EST_EITC          <dbl> 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.00~
$ EST_FICAtax       <dbl> 9765.242, 13950.346, 13950.346, 9765.242, 9765.242, 2~
$ EST_HEAP          <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ EST_Housing       <dbl> 1478.328, 0.000, 0.000, 2612.802, 2612.802, 0.000, 0.~
$ EST_IncomeTax     <dbl> 33580.543, 95503.234, 95503.234, 27574.365, 27574.365~
$ EST_MOOP          <dbl> 2642.80, 4766.80, 4766.80, 3218.80, 3218.80, 0.00, 0.~
$ EST_Nutrition     <dbl> 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0~
$ EST_PovGap        <dbl> 0.000, 0.000, 0.000, 0.000, 0.000, 3214.779, 3214.779~
$ EST_PovGapIndex   <dbl> 0.0000000, 0.0000000, 0.0000000, 0.0000000, 0.0000000~
$ Ethnicity         <dbl> 4, 1, 1, 1, 1, 4, 4, 4, 2, 3, 3, 5, 2, 2, 2, 2, 1, 1,~
$ FamType_PU        <dbl> 8, 2, 2, 2, 2, 2, 2, 7, 8, 2, 2, 2, 2, 1, 1, 1, 1, 1,~
$ FTPTWork          <dbl> 1, 1, 3, 1, 1, 1, 1, 1, 3, 1, 2, 1, 3, 1, 1, 3, 1, 3,~
$ INTP_adj          <dbl> 0.0000, 101309.7000, 50654.8520, 0.0000, 0.0000, 0.00~
$ MRGP_adj          <dbl> 0.0000, 2532.7424, 2532.7424, 0.0000, 0.0000, 0.0000,~
$ NYCgov_Income     <dbl> 81417.96, 228417.95, 228417.95, 117058.23, 117058.23,~
$ NYCgov_Pov_Stat   <dbl> 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,~
$ NYCgov_REL        <dbl> 0, 0, 1, 0, 1, 0, 1, 12, 0, 0, 1, 0, 1, 0, 1, 2, 0, 1~
$ NYCgov_Threshold <dbl> 16241.60, 22900.65, 22900.65, 22900.65, 22900.65, 229~
$ Off_Pov_Stat      <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,~
$ Off_Threshold     <dbl> 13064, 16815, 16815, 15178, 15178, 16815, 16815, 1306~
$ OI_adj            <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ PA_adj            <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ Povunit_ID        <dbl> 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
$ Povunit_Rel       <dbl> 1, 1, 2, 1, 2, 1, 2, 1, 1, 1, 2, 1, 2, 1, 2, 3, 1, 2,~
$ PreTaxIncome_PU   <dbl> 127650.22, 346479.19, 346479.19, 155003.84, 155003.84~
$ RETP_adj          <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 23402~
$ RNTP_adj          <dbl> 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 2026.1940, 20~
$ SEMP_adj          <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 11144.07, 0.00, 0~
$ SSIP_adj          <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ SSP_adj           <dbl> 0.00, 0.00, 12157.16, 27353.62, 0.00, 0.00, 0.00, 0.0~
$ TotalWorkHrs_PU   <dbl> 2, 2, 2, 1, 1, 1, 1, 3, 5, 2, 2, 3, 3, 1, 1, 1, 3, 3,~
$ WAGP_adj          <dbl> 127650.220, 182357.450, 0.000, 91178.727, 36471.492, ~
```

```r
# Removal of NAs
nyc_poverty_data_2018 <- NYCgov_Poverty_Measure_Data_2018_20240326 |>
  na.omit()

# Verification
sum(is.na(nyc_poverty_data_2018))
```

```
[1] 0
```

Principal Component Analysis (PCA) - An Unsupervised Learning Method

Unsupervised learning is a type of machine learning where the model is trained on input data without explicit output labels. The goal of unsupervised learning is to uncover hidden patterns, structures, or relationships within the data. This means that the algorithm must learn to identify patterns or structure in the input data without explicit guidance.
As well, with PCA there is no need of specifying a target (or response variable), which is a huge upside involving data reduction; PCA is often quite attractive for this reason. PCA is just one of many unsupervised learning methods, however, visually its interpretation is not tedious concerning identification of the strong variables and the possible high association between them.

The main objective of PCA is dimensionality reduction, where it seeks to transform the original high-dimensional dataset into a lower-dimensional space while preserving as much variance as possible. PCA achieves this by finding orthogonal axes (principal components) that capture the maximum variance in the data. These principal components are linear combinations of the original features.

While PCA is often used as a preprocessing step for supervised learning tasks, such as classification or regression, PCA itself does not utilize any target labels during its execution. Instead, it focuses solely on the input features to identify patterns and reduce the dimensionality of the data.

Principal Component Analysis (PCA) is a widely used technique for dimensionality reduction and data visualization. While PCA is a powerful tool, it relies on certain assumptions for its validity. Here are the main assumptions for PCA:

1. **Linearity**: PCA assumes that the relationships between variables are linear. This means that the variables are related to each other through straight-line relationships. If the relationships are highly nonlinear, PCA may not provide meaningful results.

2. **Multivariate Normality**: PCA assumes that the variables follow a multivariate normal distribution. This means that the joint distribution of the variables is Gaussian. If the data deviates significantly from this assumption, PCA results may not be reliable.

3. **Independence or Low Correlation**: PCA works best when variables are either independent or have low correlation with each other. The technique aims to find orthogonal (uncorrelated) axes of maximum variance. If variables are highly correlated, PCA may not effectively capture the underlying structure of the data.

4. **Homoscedasticity**: PCA assumes that variables have equal variance, also known as homoscedasticity. If the variance of variables differs substantially, PCA may be biased towards variables with higher variance.

5. **Large Variances Represent Important Information**: PCA assumes that large variances in the data represent important information and should be retained in the reduced-dimensional space. Variables with small variances are assumed to contain less relevant information and are often discarded in dimensionality reduction.

6. **Scale Consistency**: PCA is sensitive to the scale of the variables. Therefore, it's essential to scale or normalize the variables before performing PCA to ensure that variables with larger scales do not dominate the analysis.

7. **Complete and Non-Sparse Data**: PCA assumes that the dataset is complete, meaning there are no missing values. Additionally, PCA works best with dense (non-sparse) data, where most of the values are observed.

While PCA is robust and widely applicable, violating these assumptions can lead to unreliable results. It's essential to assess whether the data meets these assumptions before applying PCA and consider alternative techniques if the assumptions are not met. Additionally, exploratory data analysis (EDA) can help identify potential issues with the data that may affect the validity of PCA results.

Our data is comprised of categorical variables, ordinal variables and continuous variables. Principal Component Analysis (PCA) is primarily designed to handle continuous variables. However, there are techniques to extend PCA to datasets with a mixture of categorical, ordinal, and continuous variables.

One common approach is to preprocess the data to convert categorical and ordinal variables into a format suitable for PCA. Here are some techniques:

1. **One-Hot Encoding**: For categorical variables with a small number of unique categories, you can use one-hot encoding to convert them into binary variables. Each category becomes a binary feature, indicating its presence or absence in the observation. For the case of having a categorical variable with three or more categories (1, 2, 3+), you would create three binary variables, each representing one category. For each observation, one of these binary variables would be 1, indicating the presence of that category, while the others would be 0.

2. **Ordinal Encoding**: For ordinal variables, you can assign integer values based on their order. This preserves the ordinal relationship between categories while making them suitable for PCA.

3. **Scaling**: It's important to scale the continuous variables to ensure that they have comparable magnitudes. Common scaling techniques include standardization (subtracting the mean and dividing by the standard deviation) or min-max scaling (scaling the values to a range between 0 and 1).

The issue major draw backs with our data set of interest:
-There's 61 variables to consider. Generally, one can make note of the different types of variables involves, where appropriately apply the transformation through streamlined variable calls in coding. Yet, for our purposes the assumption is "we don't have time for that".
-The prior assumptions for PCA makes it overall difficult in compatibility with general data. At this time there's no interest in trying to make data look normal (to be done before applying the prior three techniques).

Alternatively, there are specialized techniques for dimensionality reduction and feature selection that are better suited for mixed types of variables, such as Multiple Correspondence Analysis (MCA) or Factor Analysis of Mixed Data (FAMD). These methods can handle categorical, ordinal, and continuous variables simultaneously and provide interpretable results tailored to the nature of the variables.

At this time, the goal is just to implement dimensionality reduction and feature importance/selection techniques at a basic level. Of consequence, to abstain from implementing PCA and the prior mentioned alternatives.

The Boruta Algorithm

The Boruta algorithm is a feature selection technique that aims to identify the most relevant features in a dataset by comparing their importance against randomly generated shadow features. The Boruta algorithm can work with data consisting of categorical variables, ordinal variables, and continuous variables. The Boruta algorithm is primarily used for feature selection, particularly in machine learning tasks. It works by comparing the importance of each feature against randomized versions of itself to determine whether it significantly contributes to the prediction task.

The Boruta algorithm is a feature selection technique that aims to identify the most relevant features in a dataset by comparing their importance against randomly generated shadow features. While Boruta is relatively robust and flexible, it operates under certain assumptions and considerations:

1. **Random Forest as Base Estimator**: Boruta relies on a Random Forest classifier (or regressor) as its base estimator. Therefore, it assumes that the Random Forest model is suitable for the given dataset and task. Random Forest is known for its robustness and ability to handle noisy data, but it may not perform optimally in all scenarios.
2. **Independence of Features**: Boruta assumes that the features in the dataset are independent of each other. This assumption is important because Boruta evaluates the importance of features based on their contribution to the model's performance. If features are highly correlated, Boruta may not accurately assess their importance.
3. **Sufficient Sample Size**: Boruta requires a sufficiently large sample size to ensure robustness and reliability in feature selection. Small sample sizes may lead to overfitting or unstable results.
4. **Representativeness of Data**: Boruta assumes that the dataset is representative of the underlying population. Biased or unrepresentative datasets may lead to biased feature selection results.
5. **Noisy Features**: Boruta assumes that the dataset may contain noisy features, and it is designed to handle them by comparing feature importance against randomly generated shadow features. However, excessive noise or irrelevant features may affect the performance of Boruta.

6. **Appropriate Parameters**: Boruta has parameters that need to be appropriately set, such as the number of iterations and the significance level for determining feature importance. Setting these parameters incorrectly may impact the effectiveness of the algorithm.
7. **Binary Outcome**: Boruta is primarily designed for classification tasks with a binary outcome variable. While it can be adapted for regression tasks, its performance may vary depending on the nature of the outcome variable.
8. **Interpretability**: Boruta provides a ranking of feature importance but does not provide direct information about the nature or direction of the relationships between features and the outcome. Interpretation of Boruta results should be done in conjunction with domain knowledge and further analysis.

The Boruta algorithm is geared towards supervised learning. Namely, a target must be established to apply feature selection.

```r
library(Boruta)
# Ensure reproducibility and consistency in your analysis;
# Too reproduce your results exactly, provided that the code and data remain unchanged. Henc
set.seed(111)
boruta_features_results <- Boruta(NYCgov_Pov_Stat ~ ., data = nyc_poverty_data_2018, doTrace
```

```
 1. run of importance source...

 2. run of importance source...

 3. run of importance source...

 4. run of importance source...

 5. run of importance source...

 6. run of importance source...

 7. run of importance source...

 8. run of importance source...

 9. run of importance source...
 10. run of importance source...

 11. run of importance source...

 12. run of importance source...

 13. run of importance source...

After 13 iterations, +17 secs:

 confirmed 31 attributes: EST_Childcare, EST_Commuting, EST_EITC, EST_FICAtax, EST_Housing a

 rejected 7 attributes: ESR, LANX, OI_adj, PA_adj, SCH and 2 more;

 still have 22 attributes left.

 14. run of importance source...

 15. run of importance source...

 16. run of importance source...

 17. run of importance source...

After 17 iterations, +22 secs:

 rejected 3 attributes: CIT, DIS, SCHG;

 still have 19 attributes left.

 18. run of importance source...

 19. run of importance source...

 20. run of importance source...

 21. run of importance source...
```

```
After 21 iterations, +26 secs:

 rejected 1 attribute: AgeCateg;

 still have 18 attributes left.

 22. run of importance source...

 23. run of importance source...

 24. run of importance source...

After 24 iterations, +29 secs:

 confirmed 2 attributes: Povunit_Rel, SEMP_adj;

 rejected 1 attribute: RETP_adj;

 still have 15 attributes left.

 25. run of importance source...

 26. run of importance source...

 27. run of importance source...

After 27 iterations, +32 secs:

 confirmed 2 attributes: MAR, PWGTP;

 rejected 2 attributes: EST_HEAP, SEX;

 still have 11 attributes left.

 28. run of importance source...

 29. run of importance source...
 30. run of importance source...

 31. run of importance source...

 32. run of importance source...

 33. run of importance source...

After 33 iterations, +38 secs:

 confirmed 1 attribute: SPORDER;

 rejected 1 attribute: CitizenStatus;

 still have 9 attributes left.

 34. run of importance source...

 35. run of importance source...

 36. run of importance source...

 37. run of importance source...

 38. run of importance source...

 39. run of importance source...

After 39 iterations, +44 secs:

 confirmed 1 attribute: SERIALNO;

 still have 8 attributes left.

 40. run of importance source...

 41. run of importance source...
 42. run of importance source...

 43. run of importance source...

 44. run of importance source...

 45. run of importance source...

 46. run of importance source...

 47. run of importance source...

 48. run of importance source...

 49. run of importance source...

 50. run of importance source...

 51. run of importance source...

 52. run of importance source...

 53. run of importance source...

After 53 iterations, +57 secs:

 rejected 1 attribute: WKHP;

 still have 7 attributes left.

 54. run of importance source...

 55. run of importance source...

 56. run of importance source...

 57. run of importance source...
 58. run of importance source...

 59. run of importance source...

 60. run of importance source...

 61. run of importance source...

 62. run of importance source...

 63. run of importance source...

 64. run of importance source...

 65. run of importance source...

 66. run of importance source...

 67. run of importance source...

 68. run of importance source...

 69. run of importance source...

 70. run of importance source...

 71. run of importance source...

 72. run of importance source...

 73. run of importance source...

 74. run of importance source...

After 74 iterations, +1.3 mins:

 confirmed 1 attribute: SCHL;
 still have 6 attributes left.

 75. run of importance source...

 76. run of importance source...

After 76 iterations, +1.3 mins:

 confirmed 1 attribute: AGEP;

 still have 5 attributes left.

 77. run of importance source...

 78. run of importance source...

 79. run of importance source...

 80. run of importance source...

 81. run of importance source...

 82. run of importance source...

 83. run of importance source...

 84. run of importance source...

 85. run of importance source...

 86. run of importance source...

 87. run of importance source...

 88. run of importance source...

 89. run of importance source...
 90. run of importance source...

 91. run of importance source...

 92. run of importance source...

 93. run of importance source...

 94. run of importance source...

 95. run of importance source...

 96. run of importance source...

 97. run of importance source...

 98. run of importance source...

 99. run of importance source...
```

```
print(boruta_features_results)
```

```
Boruta performed 99 iterations in 1.64885 mins.
 39 attributes confirmed important: AGEP, EST_Childcare, EST_Commuting,
EST_EITC, EST_FICAtax and 34 more;
 16 attributes confirmed unimportant: AgeCateg, CIT, CitizenStatus,
DIS, ESR and 11 more;
 5 tentative attributes left: Boro, EducAttain, ENG, FTPTWork, JWTR;
```

```
# Be are that the Boruta implementation will run 99 iterations.
# Computation time: 12 to 16.06 minutes concerning the current data applied.
```

Tentative features have an importance that is so close to their best shadow features that Boruta is not able to make a decision with the desired confidence in the default number of Random Forest runs.

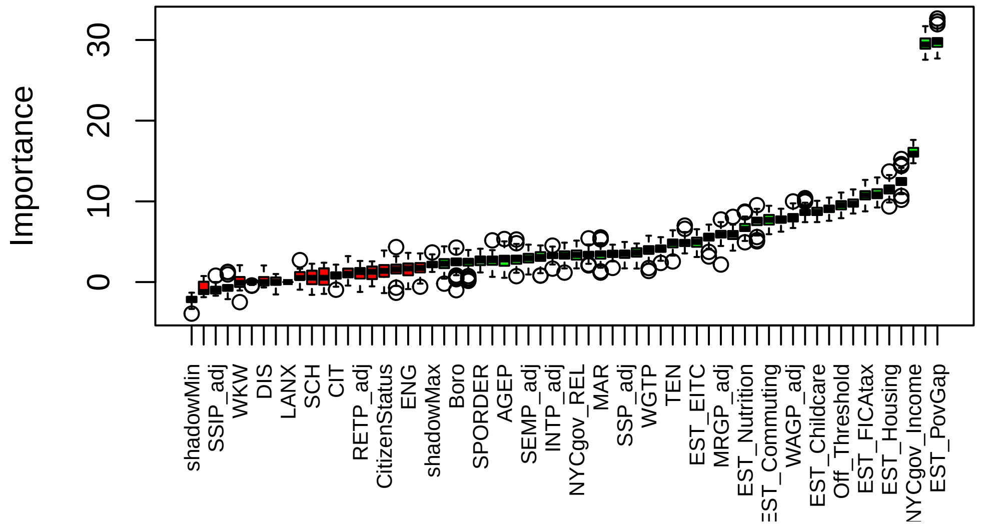
The `boruta` package also contains a `TentativeRoughFix()` function, which can be used to fill missing decisions by simple comparison of the median feature Z-score with the median Z-score of the most important shadow feature:

```r
# Take a call on tentative features
boruta_features <-
  TentativeRoughFix(boruta_features_results)
print(boruta_features)
```

```
Boruta performed 99 iterations in 1.64885 mins.
Tentatives roughfixed over the last 99 iterations.
 42 attributes confirmed important: AGEP, Boro, EducAttain,
EST_Childcare, EST_Commuting and 37 more;
 18 attributes confirmed unimportant: AgeCateg, CIT, CitizenStatus,
DIS, ENG and 13 more;
```

To now plot the boruta variable importance chart by calling `plot(boruta_features)`. However, the `x` axis labels will be horizontal. This won't be really neat. That's why to add the feature labels to the `x` axis vertically, just like in the following code chunk:

```r
plot(boruta_features, xlab = "", xaxt = "n")
lz<-lapply(1:ncol(boruta_features$ImpHistory),function(i)
boruta_features$ImpHistory[is.finite(boruta_features$ImpHistory[,i]),i])
names(lz) <- colnames(boruta_features$ImpHistory)
Labels <- sort(sapply(lz,median))
axis(side = 1, las = 2,labels = names(Labels),
at = 1:ncol(boruta_features$ImpHistory), cex.axis = 0.7)
```



The `y` axis label `Importance` identifies the `Z` score of every feature in the shuffled dataset. Any observed blue boxplots correspond to minimal, average and maximum `Z` score of a shadow feature; the red and green boxplots represent Z scores of rejected and confirmed features, respectively; red boxplots have lower `Z` score than that of maximum `Z` score of shadow feature which is precisely the reason they were put in the unimportant category.

One can confirm the importance of the features by the following:

```
get_selected_attratibutes <- getSelectedAttributes(boruta_features, withTentative = F)
print(get_selected_attratibutes)
```

```
 [1] "SERIALNO"         "SPORDER"          "PWGTP"            "WGTP"
 [5] "AGEP"             "REL"              "SCHL"             "MSP"
 [9] "MAR"              "JWTR"             "NP"               "TEN"
[13] "HHT"              "Boro"             "EducAttain"       "EST_Childcare"
[17] "EST_Commuting"    "EST_EITC"         "EST_FICAtax"      "EST_Housing"
[21] "EST_IncomeTax"    "EST_MOOP"         "EST_Nutrition"    "EST_PovGap"
[25] "EST_PovGapIndex"  "Ethnicity"        "FamType_PU"       "INTP_adj"
[29] "MRGP_adj"         "NYCgov_Income"    "NYCgov_REL"       "NYCgov_Threshold"
[33] "Off_Pov_Stat"     "Off_Threshold"    "Povunit_ID"       "Povunit_Rel"
[37] "PreTaxIncome_PU"  "RNTP_adj"         "SEMP_adj"         "SSP_adj"
[41] "TotalWorkHrs_PU"  "WAGP_adj"
```

```
glimpse(get_selected_attratibutes)
```

```
 chr [1:42] "SERIALNO" "SPORDER" "PWGTP" "WGTP" "AGEP" "REL" "SCHL" "MSP" ...
```

The above selected conveys more of a list than ranking when observing the prior Boruta based geometric exhibition.

Using the "FSelectorRccp" Package for Feature Selection

The "FSelectorRccp" package is a feature selection library in R that provides various methods for feature selection. Use of entropy-based feature selection algorithms with a sparse matrix support. It is also equipped with a parallel backend.

Here's an overview of the "Selector" package:

1. **Features**: The package offers different feature selection algorithms, including filter methods. These methods assess the importance or relevance of features based on statistical or model-driven criteria.

2. **Filter Methods**: Filter methods, such as standard information gain (also called mutual information), gain ratio, symmetric uncertainty, etc. can handle a mixture of categorical, ordinal, and continuous variables. These methods evaluate the relevance of features based on statistical measures and do not require specific assumptions about variable types.

Specifically for standard information gain (to be the choice in this project), information gain is based on Shannon Entropy, introduced by Claude Shannon in 1948, is a measure of uncertainty or randomness in a set of data. It quantifies the average amount of information (or surprise) received when observing a random variable. In the context of feature selection, Shannon entropy is used to measure the impurity or disorder of a dataset.

Information gain is a metric used in decision trees and feature selection to measure the effectiveness of a feature in separating different classes. It represents the reduction in entropy or uncertainty achieved by splitting the dataset based on a particular feature. The information gain for a feature A is calculated as the difference between the entropy of the original dataset $H(S)$ and the weighted average of the entropies of the subsets created by splitting the dataset based on feature A :

$$IG(A) = H(S) - \sum_{i=1}^{n} \frac{|S_i|}{|S|} H(S_i)$$

where S_i represents the i-th subset created by splitting the data set based on feature A, $|S|$ is the total number of instances in the data set, $|S_i|$ is the number of instances in subset S_i.

The above forumla represents the reduction in entropy achieved by splitting the data set based on feature A , and it's customarily applied in decision tree algorithms and feature selection methods to determine the most informative features.

In summary, Shannon entropy provides a measure of uncertainty in a dataset, while information gain quantifies the reduction in uncertainty achieved by splitting the dataset based on a particular feature. Information gain is calculated using Shannon entropy and is used to select the most informative features for classification or decision making.

For Shannon Entropy, $Attribute = Feature$ and $Class = Target$

Infogain:

$$H(Class) + H(Attribute) - H(Class, Attribute)$$

$H(X)$ is the Shannon Entropy for a variable X , and $H(X, y)$ is a joint Shannon Entropy for a variable X with a condition Y.

The Shannon Entropy $H(X)$ of a random variable X of a random variable X with probability distribution $P(X)$ is defined as:

$$H(X) = -\sum_i P(x_i)\log_2(P(x_i))$$

where: $P(x_i)$ is the probability of the i-th event (i.e., the probability mass function).

The sum is taken over all possible events x_i in the distribution.

The base of the logarithm is typically 2 , which leads to entropy measured in bits, representing the minimum number of bits needed to encode each event.

When $P(x_i) = 0$ the term $P(x_i)log_2(P(x_i))$ is defined as 0.

The synopsis:

- If all events in the distribution are equally probable, the entropy is maximized, indicating maximum uncertainty or randomness.
- If one event is certain to occur (i.e., $P(x_i) = 1$ for one event and 0 for all others), the entropy is minimized (0 bits), indicating no uncertainty.
- The entropy value is always non-negative and bounded by 0 (minimum entropy) and $log_2(N)$ bits (maximum entropy), where N is the number of possible events in the distribution.

3. **Compatibility**: The "FSelectorRccp" package is designed to work with various machine learning algorithms and frameworks in R. It can be integrated seamlessly into the machine learning pipeline for tasks such as classification, regression, and clustering.

```r
features_identify <- nyc_poverty_data_2018 |>
  select(-NYCgov_Pov_Stat)
      # Making a features data frame
library(FSelectorRcpp)
FS_features_prospects <-
  information_gain(features_identify,
                   nyc_poverty_data_2018$NYCgov_Pov_Stat)
      # Aquiring the importance of features
FS_features_sorted <- FS_features_prospects |>
  arrange(desc(importance)) |>
  filter(importance >= 0.012)
      # Sorting the features in descending order
      # Specify a "tolerance" based on personal preference
FS_features_sorted
```

```
        attributes importance
1       EST_PovGap 0.37287569
2  EST_PovGapIndex 0.37287569
3    NYCgov_Income 0.19640553
4  PreTaxIncome_PU 0.18500460
5    EST_IncomeTax 0.16099976
6      EST_FICAtax 0.15702549
7     Off_Pov_Stat 0.09525817
8         WAGP_adj 0.09050722
9  TotalWorkHrs_PU 0.05032167
10        EST_MOOP 0.03478146
11        EST_EITC 0.03095719
12   EST_Commuting 0.02921115
13   EST_Nutrition 0.02710769
14        FTPTWork 0.01995883
15            WKHP 0.01678655
16            SCHL 0.01340257
17             WKW 0.01277917
18      EducAttain 0.01210320
```

Enhancing Feature Selection in Data Science: A Comparative Analysis of Boruta and FSelectorRccp

Using multiple feature importance/selection methods, such as Boruta and FSelectorRccp, in a comparative manner can provide a more robust and reliable approach to choosing the best features for a given task or dataset.

Boruta is a powerful feature selection algorithm that identifies relevant features by comparing their importance against randomized shadow features. It considers all features in the dataset and determines their importance based on how well they distinguish between the target variable and random noise. Boruta is known for its ability to handle complex, nonlinear relationships and interactions in the data.

On the other hand, FSelectorRccp offers a range of feature selection methods, including filter, wrapper, and embedded methods, which assess feature importance using different criteria such as correlation, mutual information, or model performance. FSelectorRccp provides flexibility in selecting the most appropriate feature selection method based on the characteristics of the dataset and the goals of the analysis.

By using both Boruta and FSelectorRccp comparatively, we can leverage the strengths of each approach and mitigate their weaknesses. Boruta's ability to handle complex relationships and interactions can complement FSelectorRccp's diverse feature selection methods, providing a comprehensive evaluation of feature importance from different perspectives.

Comparing the results obtained from Boruta and FSelectorRccp allows us to identify consensus features that are consistently deemed important across multiple methods, as well as features that are uniquely highlighted by each approach. This comparative analysis helps in validating the robustness of feature selection results and gaining deeper insights into the dataset's characteristics.

Moreover, by integrating multiple feature selection methods, we can enhance the interpretability and generalizability of the selected features, leading to more effective and reliable predictive models. Ultimately, the combined use of Boruta and FSelectorRccp contributes to the development of data-driven solutions that are both accurate and interpretable, paving the way for better decision-making in various domains.

```
FS_features_sorted_list <- FS_features_sorted |>
  select(attributes) |>
  as.list()
FS_features_sorted_list
```

```
$attributes
 [1] "EST_PovGap"       "EST_PovGapIndex" "NYCgov_Income"   "PreTaxIncome_PU"
 [5] "EST_IncomeTax"    "EST_FICAtax"     "Off_Pov_Stat"    "WAGP_adj"
 [9] "TotalWorkHrs_PU" "EST_MOOP"         "EST_EITC"        "EST_Commuting"
[13] "EST_Nutrition"    "FTPTWork"        "WKHP"            "SCHL"
[17] "WKW"              "EducAttain"
```

Identifying common attributes or feature importance/selection between the Boruta method and theFSelectorRccp method.

```
# Identify common elements
common_elements <- intersect(unlist(get_selected_attratibutes),
                             unlist(FS_features_sorted_list))
print(common_elements)
```

```
 [1] "SCHL"             "EducAttain"      "EST_Commuting"   "EST_EITC"
 [5] "EST_FICAtax"      "EST_IncomeTax"   "EST_MOOP"        "EST_Nutrition"
 [9] "EST_PovGap"       "EST_PovGapIndex" "NYCgov_Income"   "Off_Pov_Stat"
[13] "PreTaxIncome_PU" "TotalWorkHrs_PU" "WAGP_adj"
```

Now to "ratify" such consensus features into a data frame for future interests:

```
acknowledged_features <- nyc_poverty_data_2018 |>
  select(SCHL, EducAttain, EST_Commuting, EST_EITC, EST_FICAtax, EST_IncomeTax, EST_MOOP, ES

# A reminder that our target can be called by the following: nyc_poverty_data_2018$NYCgov_Po
```

```r
glimpse(acknowledged_features)
```

```
Rows: 15,625
Columns: 15
$ SCHL             <dbl> 1, 1, 21, 16, 17, 20, 17, 16, 16, 20, 24, 21, 16, 16, ~
$ EducAttain       <dbl> 1, 1, 4, 2, 2, 3, 2, 2, 2, 3, 4, 4, 2, 2, 4, 3, 1, 3, ~
$ EST_Commuting    <dbl> 0.000, 0.000, 4954.167, 6703.500, 1795.800, 1795.800, ~
$ EST_EITC         <dbl> 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 3051.696, 54~
$ EST_FICAtax      <dbl> 9765.242, 9765.242, 2915.390, 3875.096, 6122.652, 6122~
$ EST_IncomeTax    <dbl> 27574.3650, 27574.3650, 1115.2131, 8213.8594, 11775.87~
$ EST_MOOP         <dbl> 3218.80, 3218.80, 0.00, 4176.96, 9036.96, 9036.96, 201~
$ EST_Nutrition    <dbl> 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 612.5, 0.0, 0.0, 0.0, 0.~
$ EST_PovGap       <dbl> 0.000, 0.000, 3214.779, 0.000, 0.000, 0.000, 0.000, 13~
$ EST_PovGapIndex <dbl> 0.00000000, 0.00000000, 0.14037937, 0.00000000, 0.0000~
$ NYCgov_Income    <dbl> 117058.23, 117058.23, 19685.88, 27685.43, 51303.38, 51~
$ Off_Pov_Stat     <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, ~
$ PreTaxIncome_PU <dbl> 155003.84, 155003.84, 28670.65, 50654.85, 80034.66, 80~
$ TotalWorkHrs_PU <dbl> 1, 1, 1, 3, 2, 2, 3, 1, 1, 1, 3, 3, 3, 2, 3, 1, 1, 1, ~
$ WAGP_adj         <dbl> 91178.727, 36471.492, 0.000, 50654.852, 70916.789, 911~
```

Features Applying to Predictive Modelling and Machine Learning Methods

Our initial dilemma of treating 61 variables is now reduced to 15. a total of 15 variables may still be considered extensive to some practicioners of data analysis and data science. However, a reduction in features (predictors) to just under 25% is an accomplishment. It must be made known that the feature selection process greatly mitigated any occurrence of cognitive bias. Cognitive bias refers to systematic patterns of deviation from rationality or objective judgment in decision-making, often influenced by psychological factors or heuristics. These biases can lead individuals to make subjective, irrational, or inaccurate judgments or decisions, despite possessing relevant information.

There are numerous cognitive biases documented in psychology and behavioral economics, and they can manifest in various contexts, including social interactions, problem-solving, and risk assessment. Some common cognitive biases include:

1. Confirmation Bias: The tendency to favor information that confirms one's preexisting beliefs or hypotheses while disregarding contradictory evidence.
2. Anchoring Bias: The tendency to rely too heavily on the first piece of information encountered (the "anchor") when making decisions, even if it is irrelevant or misleading.
3. Availability Heuristic: The tendency to overestimate the importance or likelihood of events based on their availability in memory, often influenced by recent or vivid examples.

4. Overconfidence Bias: The tendency to overestimate one's abilities, knowledge, or the accuracy of one's judgments, leading to unwarranted confidence in decision-making.
5. Loss Aversion: The tendency to prefer avoiding losses over acquiring equivalent gains, leading to risk aversion and potentially irrational decision-making in situations involving potential losses.
6. Framing Effect: The tendency to be influenced by the way information is presented or framed, leading to different decisions based on the same information presented in different ways.
7. Hindsight Bias: The tendency to perceive events as having been more predictable or foreseeable after they have occurred, leading to an overestimation of one's ability to predict outcomes.

These are just a few examples of cognitive biases, and there are many others that can influence human judgment and decision-making. Recognizing and understanding these biases is essential for improving decision-making processes, reducing errors, and promoting more rational and effective choices in various domains, including business, healthcare, and public policy.

The only possible recognized trace of cognitive biases may be the personal "tolerance" preference applied with the FSelectorRccp implementation, namely, the cut-off value being 0.012. All development from the beginning to current is based on credible algorithms, without influence of any blatant agendas.

Now comes the intimate part, namely, becoming more direct with the features and target. The accompanying data dictionary gives details for the features and the target. Recalling the features of interest and the target:

The Target

Concerning the variable, quantitative investigation can be achieved by the following two sets of scripts:

```
summary(nyc_poverty_data_2018$variable)
unique_values_variable <- unique(nyc_poverty_data_2018$variable)
print(unique_values_educAttain)
```

"NYCgov_Pov_Stat", being the NYCgov Poverty Status, with categories *"1= In Poverty" and "2= Not in Poverty".* The target can be identified as a categorical variable. Categorical variables represent groups or categories that have no inherent order or ranking. In this case, individuals are categorized into two distinct groups: those who are in poverty and those who are not in poverty. The categories do not have a natural ordering or ranking; they are simply different groups of individuals based on their poverty status.

The Features

```r
# Dropping "SCHL" column because "EducAttain" is derived from the former...
      # According to the accompanying databook;
      # "EducAttain" is easier to interpret.
      # Importance measure for both a re clse to each other.
acknowledged_features <- acknowledged_features |>
  select(-SCHL)
# Features (predictors) of interest. Showing the first 5 or 6 rows/observations.
head(acknowledged_features)
```

```
# A tibble: 6 x 14
  EducAttain EST_Commuting EST_EITC EST_FICAtax EST_IncomeTax EST_MOOP
       <dbl>         <dbl>    <dbl>       <dbl>         <dbl>    <dbl>
1          1             0        0       9765.        27574.    3219.
2          1             0        0       9765.        27574.    3219.
3          4          4954.       0       2915.         1115.       0
4          2          6704.       0       3875.         8214.    4177.
5          2          1796.       0       6123.        11776.    9037.
6          3          1796.       0       6123.        11776.    9037.
# i 8 more variables: EST_Nutrition <dbl>, EST_PovGap <dbl>,
#   EST_PovGapIndex <dbl>, NYCgov_Income <dbl>, Off_Pov_Stat <dbl>,
#   PreTaxIncome_PU <dbl>, TotalWorkHrs_PU <dbl>, WAGP_adj <dbl>
```

Concerning the variables, quantitative investigation can be achieved by the following two sets of scripts:

summary(acknowledged_features$variable)

unique_values_variable <- unique(acknowledged_features$variable)
print(unique_values_educAttain)

"EducAttain". According to the data dictionary,
1 = "less than high school", 2 = "high school degree", 3 = "some college", 4 = "Bachelors degree or higher".
This variable is ordinal. Ordinal variables represent categories with a natural order or ranking, but the differences between categories may not be uniform or measurable. In this case, the categories "less than high school", "high school degree", "some college", and "Bachelors degree or higher" have a clear order or ranking from lower to higher levels of education.
Since the categories have a meaningful order but the difference between each category may not be uniform (e.g., the difference in educational attainment between "less than high school" and

“high school degree” may not be the same as between “some college” and “Bachelors degree or higher”), “EducAttain” to be considered an ordinal variable.

“EST_Commuting”. NYCgov estimated Commuting Costs. Units is dollar amount so this is a continuous variable, namely, “what you see is what you get”.

“EST_EITC”. NYCgov estimated Earned Income Tax Credit. Units is dollar amount so this is a continuous variable, namely, “what you see is what you get”.

“EST_FICAtax”. NYCgov estimated FICA (payroll) tax. Units is dollar amount so this is a continuous variable, namely, “what you see is what you get”.

“EST_IncomeTax”. NYCgov estimated Net Income Taxes (tax paid, net of tax credits). Units is dollar amount so this is a continuous variable, namely, “what you see is what you get”.

“EST_MOOP”. NYCgov estimated Total Medical Spending. Units is dollar amount so this is a continuous variable, namely, “what you see is what you get”.

“EST_Nutrition”. NYCgov estimated combined Nutrition-related income adjustments (SNAP, WIC, School Meals). Units is dollar amount so this is a continuous variable, namely, “what you see is what you get”.

“EST_PovGap”. The dollar difference between NYCgov family income and the NYCgov threshold. Units is dollar amount so this is a continuous variable, namely, “what you see is what you get”.

“EST_PovGapIndex”. The poverty gap expressed as a proportion of the poverty threshold. Ratio of family resource to their threshold. Continuous variables can also take on any numeric value within a certain range, and they typically represent measurements or quantities that can be expressed as real numbers.

“NYCgov_Income”. NYCgov estimated Total Income to be compared to NYC gov Threshold for determining poverty status. Units is dollar amount so this is a continuous variable, namely, “what you see is what you get”.

“Off_Pov_Stat”. Official/Federal Poverty Status, with categories *“1= In Poverty” and “2= Not in Poverty”.* The target can be identified as a categorical variable. Categorical variables represent groups or categories that have no inherent order or ranking. In this case, individuals are categorized into two distinct groups: those who are in poverty and those who are not in poverty. The categories do not have a natural ordering or ranking; they are simply different groups of individuals based on their poverty status.

“PreTaxIncome_PU”. Pre-Tax Cash Income. Units is dollar amount so this is a continuous variable, namely, “what you see is what you get”.

“TotalWorkHrs”. Total Annual Hours worked by Poverty Unit members.
1: 3500+ Hrs (Two Full-Time, Year-Round Workers)
2: > 2340 & < 3500 Hrs (One Full-Time, Year-Round, One Part-Time Worker)
3: >= 1750 & <= 2340 Hrs (One Full-Time, Year-Round Worker)
4: < 1750 Hrs (Less than One Full-Time, Year-Round Worker)
5: No Hrs Worked
This variable is ordinal. Ordinal variables represent categories with a natural order or ranking, but the differences between categories may not be uniform or measurable. Since the categories have a meaningful order but the difference between each category may not be uniform this variable is to be considered an ordinal variable.

Spearman Correlation

Spearman’s rank correlation coefficient can be appropriate for analyzing relationships between variables in a correlation matrix when you have a mixture of continuous, categorical, and ordinal variables. Here’s why Spearman’s rank correlation is often a good choice in such scenarios:

1. **Non-parametric Measure**: Spearman’s rank correlation does not assume that the variables are normally distributed or have a linear relationship. This makes it suitable for analyzing relationships between variables of different types, including ordinal and categorical variables.

2. **Monotonic Relationship**: Spearman’s rank correlation assesses the strength and direction of the monotonic relationship between variables. It measures whether variables tend to change together in the same direction, regardless of the specific form of the relationship. This is particularly useful when dealing with ordinal variables or when the relationship between variables is non-linear.

3. **Robustness**: Spearman’s rank correlation is less sensitive to outliers compared to Pearson correlation coefficient. This can be advantageous when dealing with variables that may have extreme values or when the assumptions of parametric correlation measures are violated.

4. **Ordinal Variables**: Spearman’s rank correlation is well-suited for analyzing relationships involving ordinal variables because it ranks the observations rather than considering their exact values. This allows it to capture the underlying order of the variables without making assumptions about the interval between categories.

When constructing a correlation matrix involving a mixture of continuous, categorical, and ordinal variables, you can compute Spearman's rank correlation coefficient for all pairs of variables. This approach provides a comprehensive overview of the relationships between variables while accommodating their different types and characteristics. However, it's important to interpret the results in the context of the specific data and research questions, considering the strengths and limitations of Spearman's rank correlation and the variables involved.

```
# All variable for potential modeling.
variables_of_interest <- nyc_poverty_data_2018 |>
  select(NYCgov_Pov_Stat, EducAttain, EST_Commuting, EST_EITC, EST_FICAtax, EST_IncomeTax, ES
```

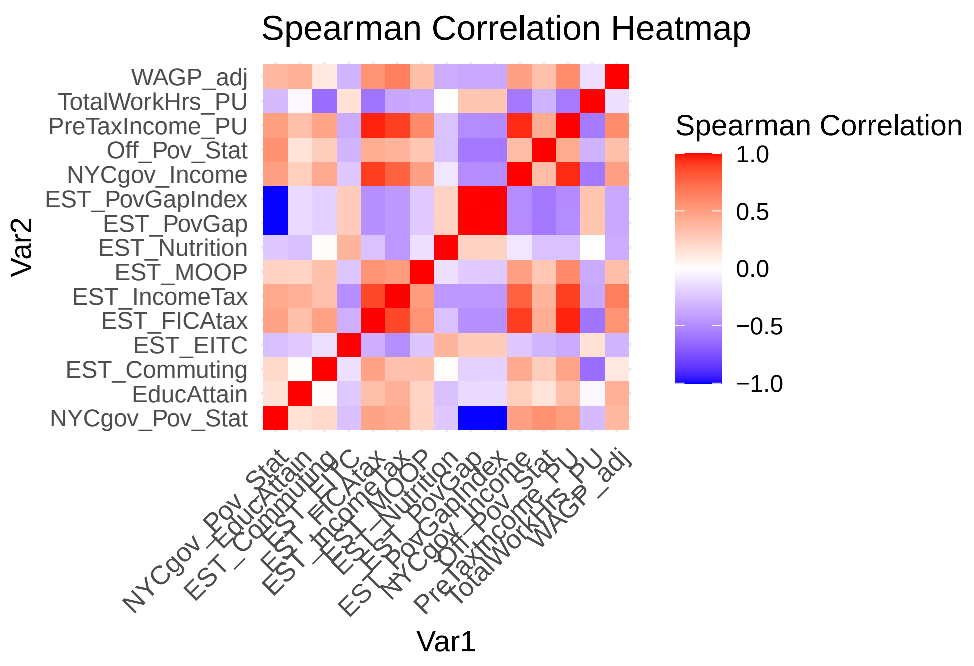
Pursuit of Spearman correlation heatmap.

```
library(Hmisc)  # For rcorr function
```

```


Attaching package: 'Hmisc'


The following objects are masked from 'package:dplyr':

    src, summarize


The following objects are masked from 'package:base':

    format.pval, units
```

```
# Assuming you have your data loaded as "data"
# Replace "data" with the name of your data frame

# Step 2: Compute Spearman correlation matrix
cor_matrix <- rcorr(as.matrix(variables_of_interest), type = "spearman")

# Step 3: Visualize the correlation matrix as a heatmap
# Convert the correlation matrix to a data frame
cor_df <- as.data.frame(cor_matrix$r)

# Get row and column names of the correlation matrix
rownames <- rownames(cor_df)
```

```r
colnames <- colnames(cor_df)

# Convert row and column names to a data frame
cor_df <- expand.grid(Var1 = rownames, Var2 = colnames)
cor_df$corr <- as.vector(cor_matrix$r)

# Create a heatmap using ggplot2
ggplot(cor_df, aes(x = Var1, y = Var2, fill = corr)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Spearman Correlation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                   size = 10, hjust = 1)) +
  coord_fixed() +
  labs(title = "Spearman Correlation Heatmap")
```



The above Spearman correlation heat map suggests multicollinearity or redundancy for variables in pairs resulting intense red blocks. Then again, underfitting may be an issue. Underfitting occurs when a machine learning model is too simple to capture the underlying structure of the data. It often results in poor performance on both the training data and unseen data.

In underfitting, the model fails to learn the relationships between the features and the target variable, leading to high bias.
Some common reasons for underfitting include:

1. **Model Complexity:** The model may be too simple to capture the complexity of the data. For example, using a linear model to fit a nonlinear relationship in the data.
2. **Insufficient Features:** The model may lack important features that are necessary to accurately predict the target variable.
3. **Too Much Regularization:** Regularization techniques, such as L1 or L2 regularization, can help prevent overfitting by penalizing large coefficients. However, applying too much regularization can lead to underfitting by overly constraining the model.
4. **Small Training Dataset:** If the training dataset is too small, the model may not have enough examples to learn the underlying patterns in the data.

By observation of the correlation heatmap matrix case 2 prior is the readily observed possible issue. Will for now abstain from reduction in features and pursue a predictive or machine learning model.

Binary Logistic Regression

Recalling the target. "NYCgov_Pov_Stat", being the NYCgov Poverty Status, with categories *"1= In Poverty" and "2= Not in Poverty"*. The target can be identified as a categorical variable. Categorical variables represent groups or categories that have no inherent order or ranking. In this case, individuals are categorized into two distinct groups: those who are in poverty and those who are not in poverty. The categories do not have a natural ordering or ranking; they are simply different groups of individuals based on their poverty status.

Following we aware that our features are continuous, categorical and ordinal.

Case of target variable having only two categories and the categories are unordered (i.e., there is no inherent order or ranking among the categories), while the features are continuous, categorical and ordinal.

```
# Convert 1 to 0 and 2 to 1
# Using recode_factor() from the 'dplyr' package
variables_of_interest$NYCgov_Pov_Stat <- recode_factor(variables_of_interest$NYCgov_Pov_Stat

# Split data into training and test sets
set.seed(123)  # Set seed for reproducibility
# Generate random indices for training and test sets
n <- nrow(variables_of_interest)
```

```r
train_indices <- sample(1:n, floor(0.7 * n))  # 70% for training
test_indices <- setdiff(1:n, train_indices)  # Remaining for testing

# Create training and test sets
train_data <- variables_of_interest[train_indices, ]
test_data <- variables_of_interest[test_indices, ]

# Fit logistic regression model using training data
logistic_model <- glm(NYCgov_Pov_Stat ~.,
                      data = train_data,
                      family = binomial(link = "logit"),
                      maxit = 1000) # Increase max iterations
```

```
Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
# Summarize the model
summary(logistic_model)
```

```

Call:
glm(formula = NYCgov_Pov_Stat ~ ., family = binomial(link = "logit"),
    data = train_data, maxit = 1000)

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)      7.971e+01  2.186e+05   0.000    1.000
EducAttain      -9.156e+00  9.308e+03  -0.001    0.999
EST_Commuting    5.633e-04  5.707e+00   0.000    1.000
EST_EITC        -2.137e-02  1.163e+01  -0.002    0.999
EST_FICAtax     -1.447e-02  4.172e+01   0.000    1.000
EST_IncomeTax    7.773e-03  3.903e+00   0.002    0.998
EST_MOOP         6.112e-03  7.800e+00   0.001    0.999
EST_Nutrition   -2.186e-03  9.738e+00   0.000    1.000
EST_PovGap      -2.978e+00  4.204e+02  -0.007    0.994
EST_PovGapIndex -6.259e+04  1.290e+07  -0.005    0.996
NYCgov_Income    9.313e-03  3.955e+00   0.002    0.998
Off_Pov_Stat    -3.143e+01  1.008e+05   0.000    1.000
PreTaxIncome_PU -1.943e-03  4.848e+00   0.000    1.000
TotalWorkHrs_PU  1.365e+00  3.023e+04   0.000    1.000
WAGP_adj        -1.726e-03  2.423e+00  -0.001    0.999
```

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 8.0800e+03  on 10936  degrees of freedom
Residual deviance: 6.4574e-08  on 10922  degrees of freedom
AIC: 30

Number of Fisher Scoring iterations: 99
```

```r
# Make predictions on test data
predicted_probs <- predict(logistic_model, newdata = test_data, type = "response")

# Convert probabilities to binary predictions
predicted_classes <- ifelse(predicted_probs > 0.5, 1, 0)

# Calculate accuracy
accuracy <- mean(predicted_classes == test_data$NYCgov_Pov_Stat)
print(paste("Accuracy:", accuracy))
```

```
[1] "Accuracy: 0.999786689419795"
```

The warning message "glm.fit: algorithm did not converge" indicates that the logistic regression algorithm failed to converge to a solution. This can happen for several reasons, including issues with the data or the model specification.

Check for Separation: Separation occurs when one or more predictors perfectly predict the outcome variable. Check if any of your predictors are perfectly associated with the target variable. If separation exists, you may need to address it by removing or transforming predictors.

References

American Community Survey and Puerto Rico Community Survey 2018 Subject Definitions. (n.d.). https://www2.census.gov/programs-surveys/acs/tech_docs/subject_definitions/2018_ACSSubjectDefin

Kosinski , M., & Zawadzki, Z. (n.d.). *Motivation, Installation and Quick Workflow.* Motivation, installation and quick workflow. Comprehensive R Archive Network. https://cran.r-project.org/web/packages/FSelectorRcpp/vignettes/get_started.html

Kursa, M. B., & Rudnicki, W. R. (2010). Feature Selection with the Boruta Package. *Journal of Statistical Software*, *36*(11), 1–13. https://doi.org/10.18637/jss.v036.i11

Opportunity, M. O. for E. (2021, July 27). *Nycgov Poverty Measure Data (2018): NYC open data.* NYCgov Poverty Measure Data (2018) | NYC Open Data. https://data.cityofnewyork.us/City-Government/NYCgov-Poverty-Measure-Data-2018-/cts7-vksw/about_data

Shannon, C. E. (1948). "A Mathematical Theory of Communication," in *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379-423, July 1948, doi: 10.1002/j.1538-7305.1948.tb01338.x.

Wilke, A., & Mata, R. (2012). Cognitive Bias. In V. S. Ramachandran (Ed.), Encyclopedia of Human Behavior (Second Edition) (pp. 531-535). *Academic Press.*

Team, D. (2018, March 7). *Boruta feature selection in R.* https://www.datacamp.com/tutorial/feature-selection-R-boruta