19603 - Data Science for Technology, Innovation and Policy
Final Group Project
Multi-Factor Income Prediction Model

Group 1

Abdulqader Koshak

Aditya Sai Sandeep Reddy Ankinapalli

Hariharan Sivaramakrishnan

Jiayue Yang

Kebing Bi

Table of Content

Abdulqader Koshak, Aditya Sai Sandeep Reddy Ankinapalli, Hariharan Sivaramakrishnan, Jiayue Yang, Kebing Bi

Introduction

In this report, we will present the findings of our project aimed at predicting the individual salary with the IPUMS CPS dataset. The project explores the interplay between industry dynamics and regional factors, and individual characteristics to understand patterns in job seeker suitability and income prediction. Our analysis of regional industry performance and individual salary prediction shows the complex interplay between economic, demographic, and geographic factors. By leveraging different algorithms, we not only provide actionable insights for stakeholders navigating the dynamic landscape of the labor market, but also all job seekers to evaluate the suitability of certain regions for their professional development. For individual users, a predictive income model can provide several benefits:

1. For people who are searching for their first job, the model could provide valuable insights into various industries and geographical features, offering knowledge about the entry-level incomes for individuals with diverse backgrounds. It also could outline the role expectations to help them set their professional goals.

2. For people wanting to switch their current job, this model demonstrates potential income increases or decreases associated with moving into a new field or position. This information is crucial for individuals trying to maximize their earnings or seeking a better work-life balance.

3. Our model also can help assess the financial return on investing in further education like a college degree, moving to another location, or changing the occupation in terms of increased earnings potential.

Research Questions

Our research aims to build a comprehensive model that predicts individual salary levels by integrating demographic, geographic, and labor force data. We analyze factors such as age, sex, race/ethnicity, education level, state, urban/rural location, industry, occupation, and work experience to understand the intricate influences on economic outcomes.

This study's insights have significant implications for workforce development, regional economic planning, and individual career strategies. By quantifying the predictive power of these factors, our research provides policymakers and job seekers with valuable insights to navigate the complexities of today's global economy.

Literature review

Income prediction methods have steadily improved, offering valuable insights. Early research, like Lazar's (2004) using SVMs, focused on accuracy and efficiency. Kibekbaev and Duman (2016) explored simpler models like linear regression, finding them surprisingly effective.

Later studies tackled more complex scenarios. Matbouli and Alghamdi (2022) used advanced algorithms like neural networks for the Saudi Arabian market, achieving high accuracy. Matkowski (2021) incorporated a wider range of data (economic, demographic, labor) with random forests and gradient boosting, revealing the intricate interplay of factors affecting income.

These advancements showcase a shift from basic prediction to understanding the complex dynamics that influence income. This progression informs our project's methodology, driving us to explore a multifaceted approach that offers both accurate prediction and insightful explanations for income variations.

Data Processing

About the Dataset

IPUMS CPS is a comprehensive dataset derived from the Current Population Survey (CPS), offering harmonized monthly data from 1962 to the present. It covers a wide range of topics related to the social, economic, and health aspects of the United States. This dataset is valuable for researchers studying long-term trends and changes in variables such as employment, family structures, and income. We have created an extract with an extensive list of variables (outlined in appendix) from which we select key variables for exploratory analysis, feature selection, and modeling listed below:

Feature Name	Description	Type of Variable
STATEFIP	State (FIPS code)	Categorical
METFIPS	Metropolitan area (FIPS code)	Categorical
AGE	Age	Continuous
SEX	Sex	Categorical
RACE	Race	Categorical
CITIZEN	Citizenship status	Categorical
OCC	Occupation	Categorical
IND	Industry	Categorical

UHRSWORKT	Hours usually worked per week at all jobs	Continuous
EDUC	Educational attainment	Continuous
INCTOT	Total personal income **(Target Variable)**	**Continuous**
Log (INCTOT)	**Log transformed Target Variable**	**Continuous**

Data Cleaning

To clean the data, we have applied various rules based on the variable documentation to make sure the records are inline with the data we require for the prediction model. Further, we have used mapping files to map the encoded State, Metropolitan Area, Occupation, Industry values to their Text values so that we may be able to interpret the trends and gain meaningful insights from exploratory data analysis. We have detailed the data cleaning part in **APPENDIX I.**

Exploratory Data Analysis

Based on the EDA, we have the following key insights:

- The transformation of INCTOT to Log(INCTOT) gives a more normalized income distribution, reducing skewness and making statistical analysis more robust.
- Scatter plot of “Age vs INCTOT” indicates a typical lifecycle income pattern: incomes rise with age as experience increases, peak in middle age, and decline as individuals approach retirement.
- There is a positive correlation between “Educational Attainment and INCTOT", as shown in the scatter plot, highlighting that higher education levels generally lead to higher incomes.
- Violin plot for “SEX vs INCTOT” shows only slight differences in income distribution between genders, with males slightly more likely to be in higher income brackets.
- Map visualization of “average income by state” reveals regional disparities, with higher incomes concentrated in states like New York and California, reflecting the economic opportunities tied to specific geographic locations.

We have documented the complete EDA process in **APPENDIX II**

Modeling

Feature Selection

Our feature selection process involved Lasso regression, a method well-suited for datasets with a large number of features. Lasso helps in feature reduction by penalizing

the absolute size of the regression coefficients. Through this technique, all variables with significant contribution to predicting individual income (INCTOT). This outcome indicates a robust interrelation between demographic, geographic, and occupational attributes with income levels.

The dataset was divided into two subsets based on the target variables: INCTOT and Log(INCTOT). This approach allowed us to construct models tailored to different transformations of income, addressing skewness in the income distribution by using its logarithmic transformation.

Model Selection and Fitting

We have adopted a two-stage modeling process designed to leverage the strengths of both Lasso and Ridge regression techniques. This strategic combination allows us to leverage the feature selection capabilities of Lasso while utilizing the predictive power and stability of Ridge regression.

With the optimal set of predictors identified by Lasso, we proceeded to fit a Ridge regression model. This method was chosen due to its ability to handle multicollinearity between predictors and its robustness in situations where prediction stability is crucial. Ridge regression, unlike Lasso, does not force coefficients to zero but instead shrinks them towards zero, which allows us to retain all the shortlisted features in the model while also controlling for overfitting.

We have documented the complete Feature Selection and Modeling process in **APPENDIX III**

Interpretation of Model Results

Based on the analysis above, our model has been carefully crafted to forecast individual income levels by considering three kinds of features - demographic, geographic and labor force factors. Using two Ridge regression models, one with raw income data and one with a logarithmic transformation to income data, we found that the performance of the model using just raw income was suboptimal and nonsensical while the Log Transformation on the Income variable improved the performance quite a bit.

Model	Train_MSE	Validation_MSE	R_squared	Test_MSE
Log(INCTOT)	0.71	0.59	0.27	0.67
INCTOT	8.40E+09	7.52E+09	0.19	6.75E+09

Further, as the output of the fitting the model, we also get coefficients which can be interpreted as follows:

In the case of the Log(INCTOT) model, the Intercept sets 9.28 as the baseline with all other coefficients modifying the Income of the individual by a percentage which is determined by the coefficient. For example, if we consider the coefficient of AGE (assume 25 for an Individual) which is ~0.012 and EDUCATION (Consider 123, which corresponding to Masters Degree) which is ~0.01, assuming the person is living in STATE coded as 11(Columbia; Coeff = 0.27), with the OCCUPATION_CODE as 3(IT Services; Coeff = 0.04) and INDUSTRY_CODE as 27(Broadcasting; Coeff = 0.21), we can interpret it as each increment in AGE and Level of EDUCATION in this STATE, OCCUPATION, and INDUSTRY, would increase the person's wage by 2x (Final Coeff = 2.05 or 205%).

Limitations of Current Approach

Using the Ridge model, our approach is based on the assumption that there are linear relationships between predictors and the target variable. However, real-world economic behavior often demonstrates non-linear, threshold, or even discontinuous relationships. Therefore, as a next step we could add interaction items in our model to improve the predictive performance.

Additionally, the removal of few features in the final model, as determined by Lasso regression, could remove potentially significant factors from consideration. This exclusion could limit the model's ability to reflect the full spectrum of socio-economic diversity and its impact on income.

Finally, our model is only as good as the data it learns from. If there are biases or systemic issues within the dataset, our model may inadvertently perpetuate or fail to identify these. Hence, the model's predictions and insights should be contextualized within the broader socio-economic framework.

Conclusion

Our project aimed to create a model that predicts what people earn using a mix of information about their jobs, where they live, and their personal background. We used a special two-step method to pick the most important pieces of information that affect how much money people make.

First, we used a technique called Lasso regression to figure out which pieces of information (like age, education, and job type) were important. Then, we used another

method called Ridge regression to predict income levels. This approach was great for dealing with complex data and making sure our predictions were reliable.

But our method isn't perfect. For example, we assumed that the relationship between the factors we studied and income is a straight line, which might not always be the case. We also had to leave out some features which could be important. Plus, if the data we started with has any mistakes or biases, those could affect our predictions too.

In the end, our model gives us a good look at what affects how much money people might earn and could help job seekers and those making policies. It's a step forward in using data to understand the job market, but there's still more to be done to make our predictions even better.

References

1. Kibekbaev, Azamat, and Ekrem Duman. "Benchmarking regression algorithms for income prediction modeling." Information Systems 61 (2016): 40-52.
2. Lazar, Alina. "Income prediction via support vector machine." In ICMLA, pp. 143-149. 2004.
3. Matbouli, Yasser T., and Suliman M. Alghamdi. "Statistical machine learning regression models for salary prediction featuring economy wide activities and occupations." Information 13.10 (2022): 495.
4. Matkowski, Michael. "Prediction of individual income: A machine learning approach." (2021).

Appendix

1. **Complete List of Features in the Data Extract=**
2. **Appendix I (Data Processing)**
3. **Appendix II (Exploratory Data Analysis)**
4. **Appendix III (Modeling)**

Complete List of Features in the Data Extract

Feature	Description
YEAR	Survey year
SERIAL	Household serial number
CPSID	CPSID, household record
REGION	Region and division
STATEFIP	State (FIPS code)
COUNTY	County (FIPS code)
METFIPS	Metropolitan area (FIPS code)
PERNUM	Person number in sample unit
AGE	Age
SEX	Sex
RACE	Race
YRIMMIG	Year of immigration
CITIZEN	Citizenship status
NATIVITY	Foreign birthplace or parentage
EMPSTAT	Employment status
LABFORCE	Labor force status
OCC	Occupation
IND	Industry
CLASSWKR	Class of worker
UHRSWORKT	Hours usually worked per week at all jobs
DURUNEMP	Continuous weeks unemployed
WKSTAT	Full or part time status
EDUC	Educational attainment recode
FTOTVAL	Total family income
INCTOT	Total personal income
INCRENT	Income from rent
EITCRED	Earned income tax credit
MARGTAX	Federal income marginal tax rate
STATETAX	State income tax liability, before credits
STATAXAC	State income tax liability, after all credits
TAXINC	Taxable income amount

Data Pre-processing

Data Loading

To load the extracts created on the https://cps.ipums.org/cps/index.shtml platform in RStudio, we require the ".XML" file and ".DAT" file corresponding to the extract as well as the library "ipumsr" which will enable RStudio to read the xml and load the data accordingly. We also use the "openxlsx" library to store the cleaned data in an XLSX format for ease of exploration in Excel.

```
list.of.packages <- c("ipumsr", "openxlsx", "tidyverse", "tidycensus", "tigris")
new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
if(length(new.packages)) install.packages(new.packages)

library('ipumsr')
library('tidyverse')
library('openxlsx')

set.seed(123) # Setting a seed for reproducibility

ddi <- read_ipums_ddi("cps_00004.xml")
data <- read_ipums_micro(ddi)
```

Data Cleaning and Filtering

In this pre-processing step, we apply various rules to clean and filter our extract. Below we describe the rules applied as well as the rationale behind the rule:

1. `YEAR == 2023`: Selects data only for the year 2023, focusing on a specific year for analysis
2. `INCTOT != 999999999`: Excludes cases where total income is coded as 999999999, which represents missing or invalid values.
3. `AGE != 0`: Filters out cases where age is coded as 0 (invalid/missing value)
4. `!is.na(CITIZEN)`: Removes cases where citizenship status is missing, ensuring all selected cases have a valid citizenship status.

   1- Born in U.S, 2- Born in U.S. outlying, 3- Born abroad of American parents, 4- Naturalized citizen, 5- Not a citizen
5. `!is.na(IND)`: Eliminates cases where industry information is missing, ensuring all selected cases have a valid industry code.
6. `IND != 0`: Excludes cases where industry code is 0 (invalid/missing value)
7. `METFIPS < 99998`: Excludes cases where the metropolitan area code is 99999 and 99998 (invalid/missing value)

8. `RACE != 999`: Excludes cases where race is coded as 999 (invalid/missing value)
9. `!is.na(RACE)`: Removes cases where race information is missing, ensuring all selected cases have a valid race code.
10. `EDUC != 999`: Excludes cases where education level is coded as 999, assuming it's an invalid or missing value.
11. `INCTOT > 0`: Filters out cases where total income is less than or equal to 0
12. `EMPSTAT == 10`: Selects cases where employment status is coded as 10, indicating being employed.
13. `CLASSWKR %in% c(21,22,25,27,28)`: Selects cases where class of worker is in the specified values (21, 25, 27, 28), which represent specific categories of worker status.

    21- Wage/salary, private, 22- Private, for profit, 24- Wage/salary, government, 25- Federal government employee, 27- State government employee, 28- Local government employee
14. `WKSTAT %in% c(11,12,14,15)`: Selects cases where work status is in the specified values (11, 12, 14, 15), representing specific categories of work status.

    11 Full-time hours (35+), usually full-time 12 Part-time for non-economic reasons, usually full-time 14 Full-time hours, usually part-time for economic reasons 15 Full-time hours, usually part-time for non-economic reasons
15. `UHRSWORKT < 997`: Excludes cases where usual hours worked per week are greater than or equal to 997 (invalid/missing value)

Further, post cleaning of the data extract, we drop the columns which would not be useful in modelling/cannot be used as features such as YEAR, ID, and certain variables used for filtering

```r
# Data Cleaning Process
data = data %>% filter(
  YEAR == 2023,
  INCTOT != 999999999,
  AGE != 0,
  !is.na(CITIZEN),
  !is.na(IND),
  IND != 0,
  METFIPS < 99998,
  RACE != 999,
  !is.na(RACE),
  EDUC != 999,
  INCTOT > 0,
  EMPSTAT == 10,
  CLASSWKR %in% c(21,22,24,25,27,28),
  WKSTAT %in% c(11,12,14,15),
  UHRSWORKT < 997
)

data <- data %>%
  select(-YEAR, -SERIAL, -CPSID,
         -PERNUM, -COUNTY, -YRIMMIG,
         -NATIVITY, -EMPSTAT, -LABFORCE,
         -CLASSWKR, -DURUNEMP, -WKSTAT,
         -FTOTVAL, -INCRENT, -EITCRED,
         -MARGTAX, -STATETAX, -STATAXAC,
         -TAXINC, -REGION)
```

Feature Engineering

Based on our literature review, we also implement a log transform of our target variable which is "INCTOT" which denotes the Total personal income of the individual. This is done primarily due to a few reasons:

1. **Reduce Skewness:** Income data often exhibits skewness, ie, there are a few very high earners and a larger number of people clustered near the median income. Log transformation compresses the scale, bringing those high earners closer to the rest of the data and creating a more symmetrical distribution. This is important because many statistical methods assume normality in the data
2. **Linearize Relationship:** In our model, we are also interested in observing the **percentage change** in income rather than the absolute difference. Logarithms have a convenient property where a change in the log of income reflects a percentage change in the original income. This can make it easier to interpret the relationship between income and other economic factors in the model
3. **Improved Performance:** Log-transformed income may improve the performance of our models by making the relationship between income and other variables more linear

```
# Performing Log transformation on target variable for future analysis

data$log_INCTOT <- log(data$INCTOT)
```

Further, we also explore a few categorical columns which have an exceptionally large number of categories. The columns are namely: 1. OCC (Occupation Code): ~**520 Categories** 2. IND (Industry Code): ~**260 Categories** 3. METFIPS (Metropolitan Area FIPS Code): ~**300 Categories**

For OCC and IND, we were able to locate Mapping files `Mapping.xlsx` which enable us to group various OCC codes and IND codes into their respective classes, vastly reducing the number of categories to <50. In `Mapping.xlsx`, we have used separate sheets for each important feature.

For METFIPS, we used `tidycensus` and `tigris` packages, using which we mapped the Metropolitan areas to their respective populations, and ranked them according to their sizes to create a numerical feature. The underlying assumption for this is, not considering cost-of-living and taxes, larger cities tend to be more lucrative in terms of opportunity, and consequently would have higher wages and would attract more people.

Further, we use additional mapping files to map codes OCC, IND, STATEFIPS, and METFIPS with their respective Text Values so that we can interpret the Exploratory Data Analysis results

```
wb <- loadWorkbook("Mapping.xlsx")
occ <- read.xlsx(wb, sheet = "OCC")
ind <- read.xlsx(wb, sheet = "IND")
state <- read.xlsx(wb, sheet = "STATEFIPS")
met <- read.xlsx(wb, sheet = "METFIPS")

library(tidycensus)
library(tigris)
options(tigris_use_cache = TRUE)

pop <- get_estimates(geography = "metropolitan statistical area/micropolitan statistical area", variable
pop <- pop %>% filter(variable == "POPESTIMATE")
pop <- pop %>% select(c(GEOID, value))

ctc <- c("STATEFIP", "METFIPS", "AGE", "SEX", "RACE", "CITIZEN", "OCC", "IND", "UHRSWORKT", "EDUC", "IN
data <- data %>% mutate_at(vars(ctc), as.integer)
```

```r
data <- left_join(data, occ, by = "OCC")
data <- left_join(data, ind, by = "IND")
data <- left_join(data, state, by = "STATEFIP")
data <- left_join(data, met, by = "METFIPS")

data <- merge(data, pop, by.x = "METFIPS", by.y = "GEOID", all.x = TRUE)

# Checking for unique Metropolitan areas where population is missing
unique(subset(data, is.na(value))$MET)

# Unique Met Areas are :"California-Lexington Park, MD", "Cleveland-Elyria, OH", "Dayton, OH", "Prescot
# We manually find the population of these 4 areas and add to the data

data$value[which(data$MET == "California-Lexington Park, MD")] = 113000
data$value[which(data$MET == "Cleveland-Elyria, OH")] = 2063132
data$value[which(data$MET == "Dayton, OH")] = 135944
data$value[which(data$MET == "Prescott, AZ")] = 47603

# Create a Rank using "value" (population) column
uni_pop <- unique(data[, c("MET", "value")])
uni_pop$MET_CODE <- rank(uni_pop$value, ties.method = "min")

data <- merge(data, uni_pop, by.x = c("MET", "value"), by.y = c("MET", "value"), all.x = TRUE)

# Drop population value since we obtained RANK for each Metro Area
data <- data %>% select(-value)

# Ordering Columns
data <- data[, c("METFIPS", "MET", "MET_CODE", "STATEFIP", "STATE", "AGE", "SEX", "RACE", "CITIZEN", "E
```

Saving the Dataset

Finally, we save the data. We use this cleaned, processed data for exploratory Data Analysis

```r
write.csv(data, "eda_data_clean.csv", row.names = FALSE)
```

Exploratory Data Analysis

Here, we explore the trends in wages across various features and summarize our findings.

Data Loading

```
list.of.packages <- c("tidyverse", "maps", "ggplot2", "dplyr")
new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
if(length(new.packages)) install.packages(new.packages)

library('tidyverse')
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(ggplot2)
library(dplyr)
library(maps)
```

```
##
## Attaching package: 'maps'
##
## The following object is masked from 'package:purrr':
##
##     map
```

```
library(viridis) # For color scales in maps
```

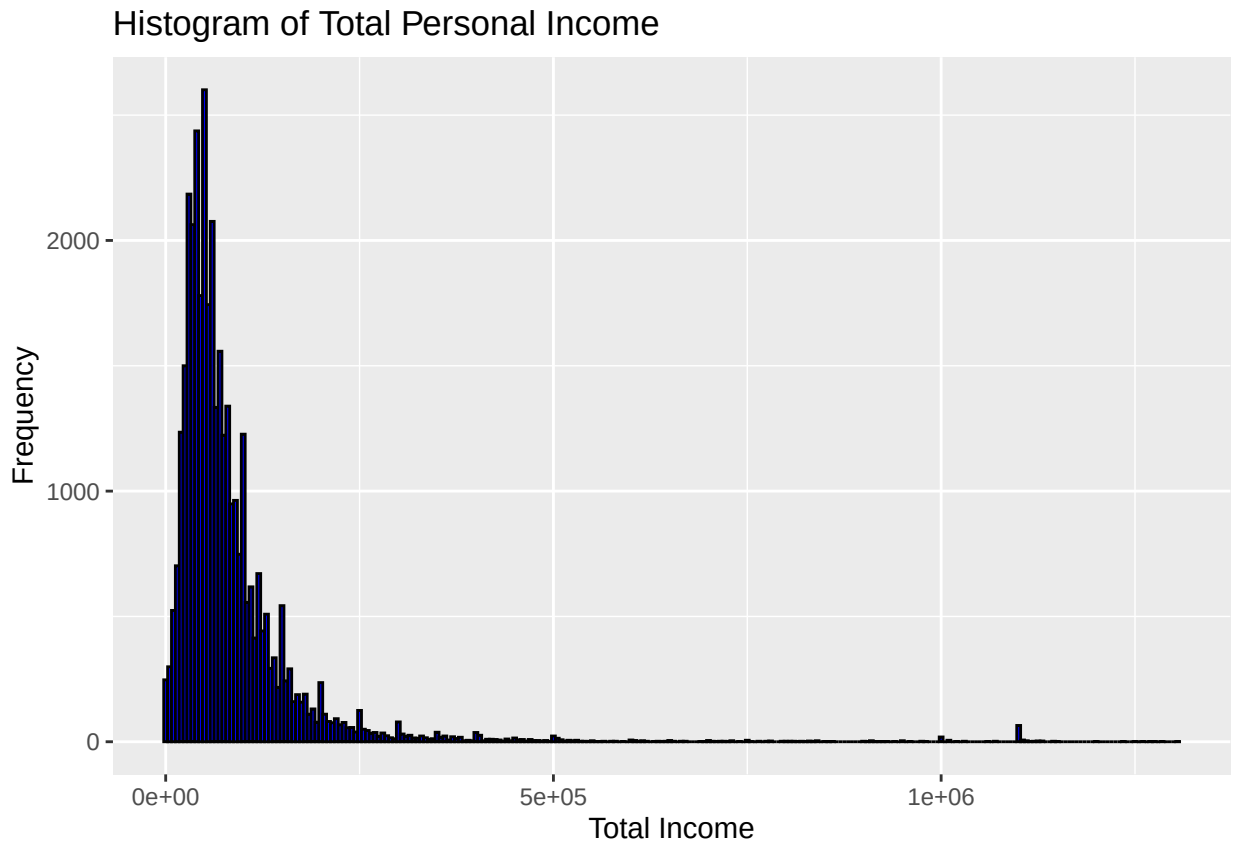
```
## Loading required package: viridisLite
##
## Attaching package: 'viridis'
##
## The following object is masked from 'package:maps':
##
##     unemp
```

```r
set.seed(123) # Setting a seed for reproducibility

# Loading the processed data
data <- read.csv("eda_data_clean.csv")
```

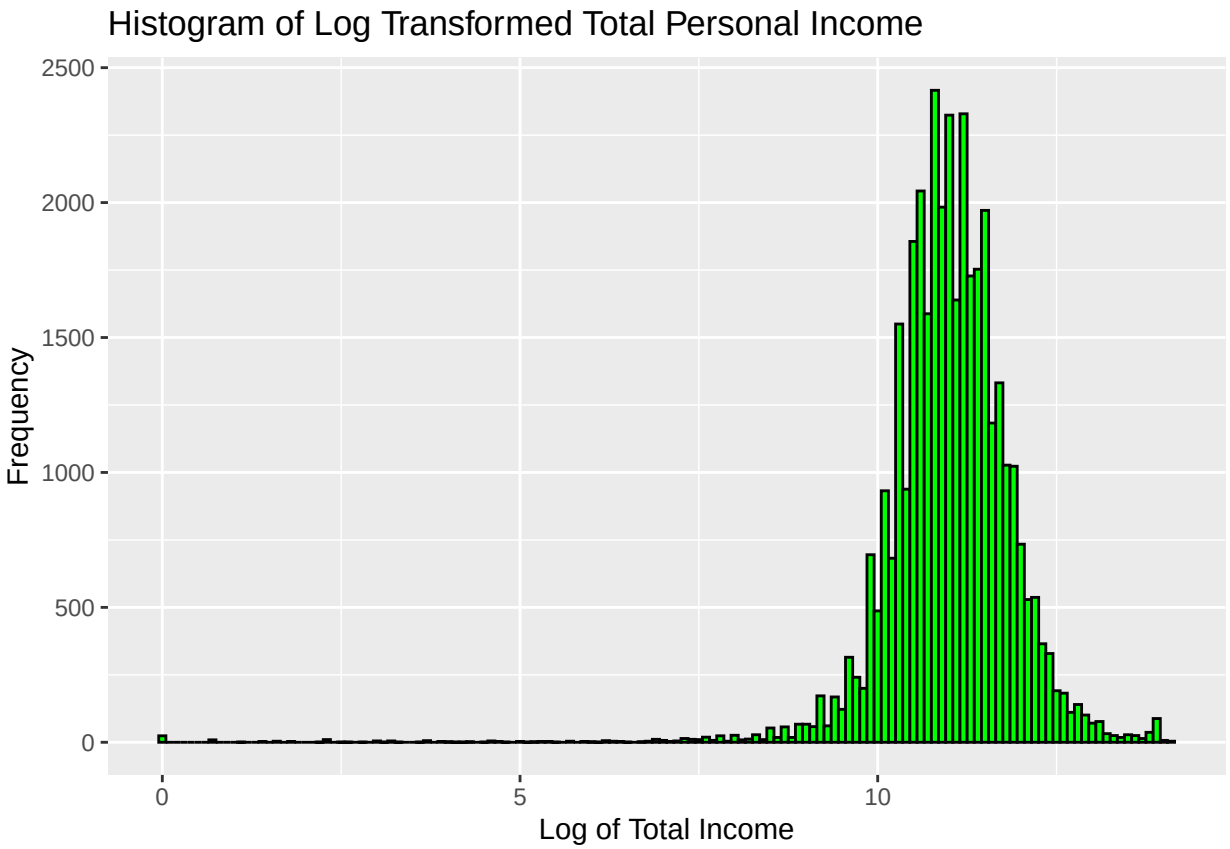
Target Variable Histograms

Here, we look at histograms for INCTOT and Log(INCTOT)

```r
# Histogram for INCTOT
ggplot(data, aes(x = INCTOT)) +
geom_histogram(binwidth = 5000, fill = "blue", color = "black") +
labs(title = "Histogram of Total Personal Income", x = "Total Income", y = "Frequency")
```




```r
# Histogram for Log(INCTOT)
ggplot(data, aes(x = log_INCTOT)) + # Assuming the log column is named as Log.INCTOT
geom_histogram(binwidth = 0.1, fill = "green", color = "black") +
labs(title = "Histogram of Log Transformed Total Personal Income", x = "Log of Total Income", y
= "Frequency")
```

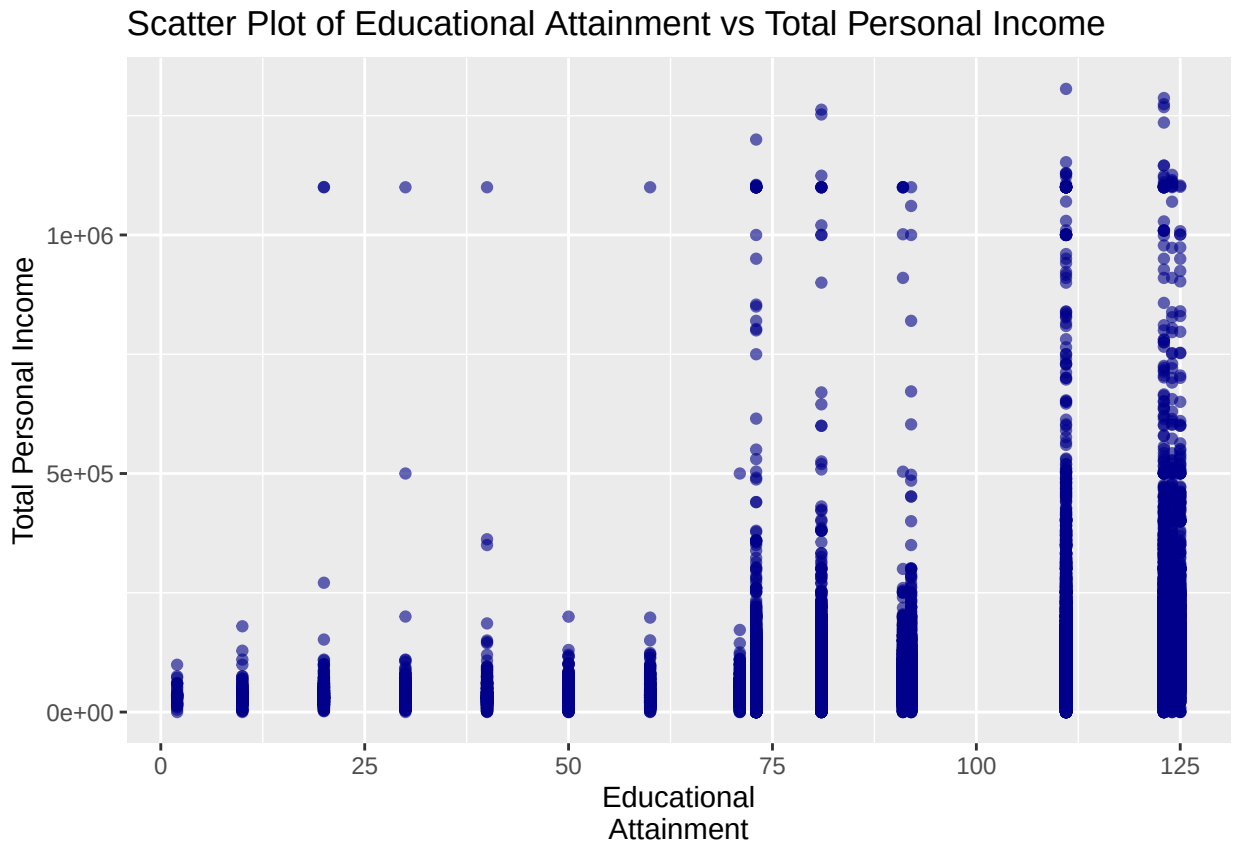


We observe that by log transformation, we are able to shift the concentration from the left side of the original plot to a distribution which is more central and suitable for our analysis.

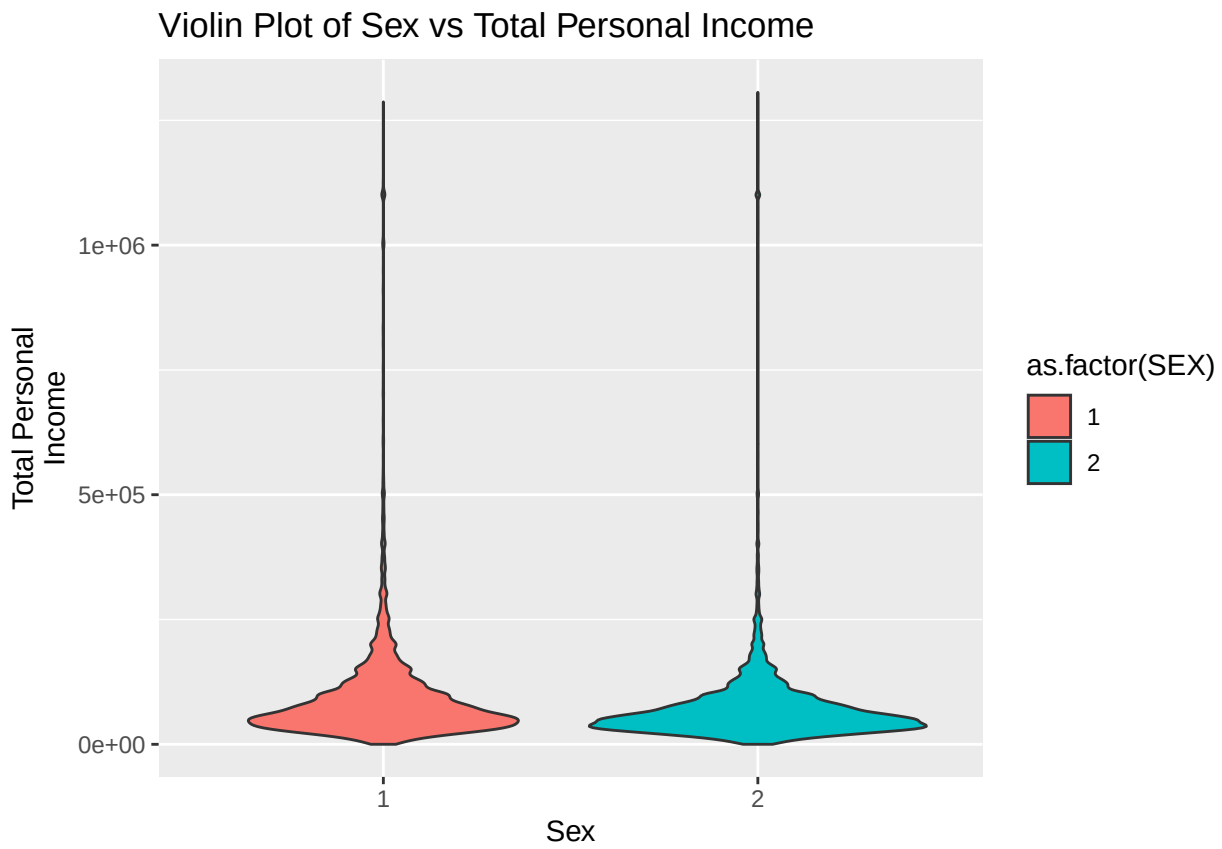
Plotting AGE and Education against Total Income

```
# Scatter Plot for Age vs INCTOT
ggplot(data, aes(x = AGE, y = INCTOT)) +
geom_point(alpha = 0.6, color = "red") +
labs(title = "Scatter Plot of Age vs Total Personal Income", x = "Age", y = "Total Personal
Income")
```


```
# Scatter Plot for EDUC vs INCTOT
ggplot(data, aes(x = EDUC, y = INCTOT)) +
geom_point(alpha = 0.6, color = "darkblue") +
labs(title = "Scatter Plot of Educational Attainment vs Total Personal Income", x = "Educational
Attainment", y = "Total Personal Income")
```




```
# Violin Plot for SEX vs INCTOT
ggplot(data, aes(x = as.factor(SEX), y = INCTOT, fill = as.factor(SEX))) +
geom_violin() +
labs(title = "Violin Plot of Sex vs Total Personal Income", x = "Sex", y = "Total Personal
Income")
```



We are able to observe trends in Age and Education, where in we see a rise in income as Age increases and then after a certain point (retirment), the income goes down. In case of Education, we see as the degree attained is better, the income is also generally better.

There are no stark differences for the gender plot. However, we can see that Males(1) have a slightly higher concentration on higher personal income as compared to Female(2).

State-wise Trends

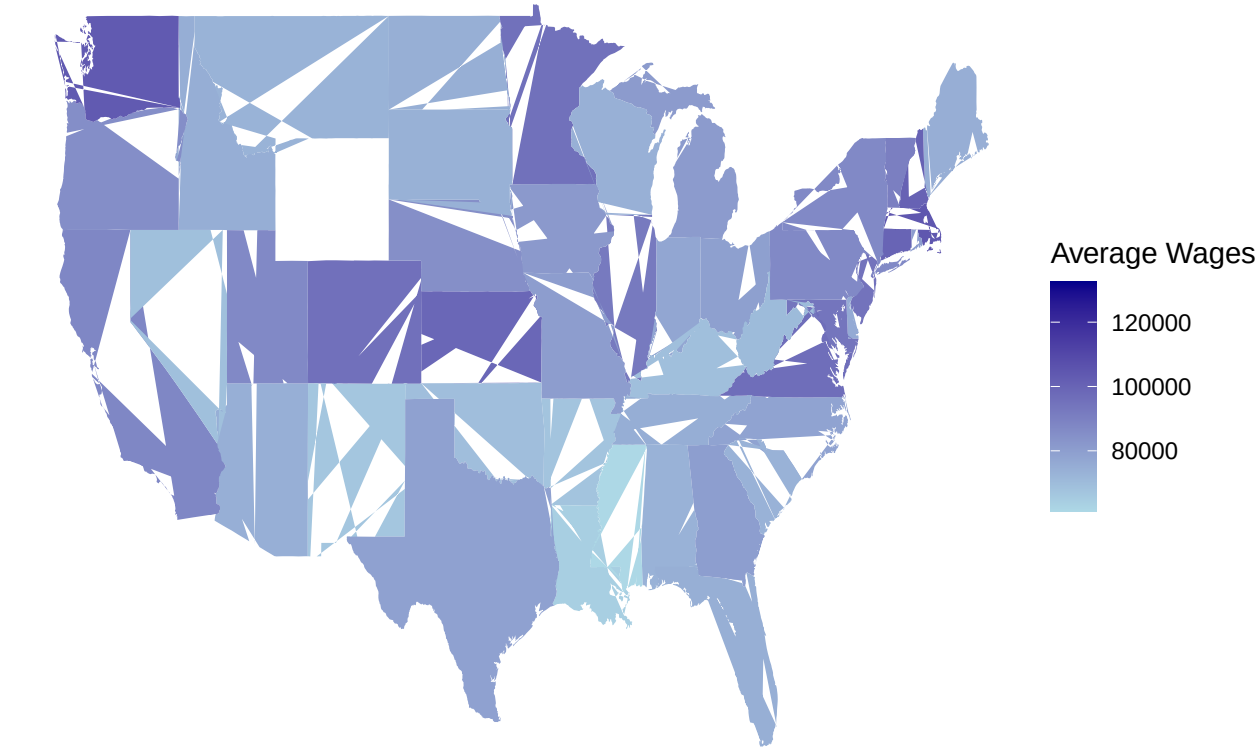
We plot the average wages across various states on the US map.

```
us_map <- map_data("state")

# Aggregate data to get average INCTOT by STATEFIP
state_income <- data %>%
group_by(STATEFIP, STATE) %>%
summarise(Avg_INCTOT = mean(INCTOT, na.rm = TRUE))
```

```
## `summarise()` has grouped output by 'STATEFIP'. You can override using the
## `.groups` argument.
```

```
state_income$STATE <- toupper(state_income$STATE)
us_map$region <- toupper(us_map$region)
```

```r
map_data_merged <- merge(us_map, state_income, by.x = "region", by.y = "STATE", all.x = TRUE)
map_data_merged <- map_data_merged %>% select(-subregion)
map_data_merged_clean <- na.omit(map_data_merged)


# Plotting the map
ggplot(data = map_data_merged_clean) +
  geom_polygon(aes(x = long, y = lat, group = group, fill = Avg_INCTOT)) +
  scale_fill_gradient(low = "lightblue", high = "darkblue", name = "Average Wages") +
  labs(title = "Average Wages per State") +
  theme_void()
```



We observe the typical states where we would see large incomes, on eastern coast, New York and on the Western Coast, Seattle and California

Top Metropolitans, Occupations, and Industry

Here, we aggregate the data to get the top 10 areas, occupations and indsutry based on average wages.

```r
# Top 10 Metropolitan Areas by Average Salary
data %>%
group_by(MET) %>%
summarise(Avg_Salary = mean(INCTOT, na.rm = TRUE)) %>%
```

```
arrange(desc(Avg_Salary)) %>%
head(10)
```

```
## # A tibble: 10 x 2
##    MET                                          Avg_Salary
##    <chr>                                             <dbl>
##  1 Boulder, CO                                     146960.
##  2 San Jose-Sunnyvale-Santa Clara, CA              139548.
##  3 San Francisco-Oakland-Fremont, CA               128529.
##  4 Santa Cruz-Watsonville, CA                      127287.
##  5 Bridgeport-Stamford-Norwalk, CT                 126619.
##  6 South Bend-Mishawaka, IN-MI                     125892
##  7 Washington-Arlington-Alexandria, DC-VA-MD-WV    121899.
##  8 Medford, OR                                     115758.
##  9 Sherman-Dennison, TX                            115681.
## 10 Durham-Chapel Hill, NC                          115178.
```

```
data %>%
group_by(OCC_VAL) %>%
summarise(Avg_Salary = mean(INCTOT, na.rm = TRUE)) %>%
arrange(desc(Avg_Salary)) %>%
head(10)
```

```
## # A tibble: 10 x 2
##    OCC_VAL                                   Avg_Salary
##    <chr>                                          <dbl>
##  1 Fire inspectors                               361253
##  2 Podiatrists                                   352205
##  3 Surgeons                                      344075.
##  4 Prepress technicians and workers              268707.
##  5 Other physicians                              255147.
##  6 Chief executives                              225412.
##  7 Lawyers                                       220460.
##  8 Dentists                                      194531.
##  9 Broadcast announcers and radio disc jockeys   194410.
## 10 Petroleum engineers                           191392.
```

```
data %>%
group_by(OCC_CATEG) %>%
summarise(Avg_Salary = mean(INCTOT, na.rm = TRUE)) %>%
arrange(desc(Avg_Salary)) %>%
head(10)
```

```
## # A tibble: 10 x 2
##    OCC_CATEG                                                Avg_Salary
##    <chr>                                                         <dbl>
##  1 Legal occupations                                          171316.
##  2 Computer and mathematical science occupations              127758.
##  3 Management occupations                                     124286.
##  4 Architecture and engineering occupations                   122267.
##  5 Healthcare practitioner and technical occupations          109344.
##  6 Business and financial operations occupations              104068.
```

```
##  7 Life, physical, and social science occupations                99804.
##  8 Arts, design, entertainment, sports, and media occupations    90943.
##  9 Protective service occupations                                83385.
## 10 Sales and related occupations                                 83073.
```

```
data %>%
group_by(IND_VAL) %>%
summarise(Avg_Salary = mean(INCTOT, na.rm = TRUE)) %>%
arrange(desc(Avg_Salary)) %>%
head(10)
```

```
## # A tibble: 10 x 2
##    IND_VAL                                                      Avg_Salary
##    <chr>                                                             <dbl>
##  1 Internet publishing and broadcasting and web search portals      219765.
##  2 Oil and gas extraction                                           203224.
##  3 Wholesale electronics markets, agents and brokers                179979.
##  4 Securities, commodities, funds, trusts, and other financial inves~ 167255.
##  5 Legal services                                                   157467.
##  6 Software publishers                                              151709.
##  7 Cutlery and hand tool manufacturing                              150291.
##  8 Computer systems design and related services                     144991.
##  9 Aerospace product and parts manufacturing                        138477.
## 10 Textile and fabric finishing and coating mills                   135777
```

```
data %>%
group_by(IND_CATEG) %>%
summarise(Avg_Salary = mean(INCTOT, na.rm = TRUE)) %>%
arrange(desc(Avg_Salary)) %>%
head(10)
```

```
## # A tibble: 10 x 2
##    IND_CATEG                                            Avg_Salary
##    <chr>                                                     <dbl>
##  1 Broadcasting (except internet)                          157811.
##  2 Publishing industries (except internet)                 131513.
##  3 Management of companies and enterprises                 130441.
##  4 Professional and technical services                     129379.
##  5 Finance                                                 127787.
##  6 Internet service providers and data processing services 125266.
##  7 Computer and electronic products                        110703.
##  8 Chemical manufacturing                                  110112.
##  9 Mining                                                  109368.
## 10 Telecommunications                                      102464
```

Saving Data for modelling

Finally, once we have utilized the required columns for EDA, we save a data with only the required features for modelling

```r
data1 <- data %>% select(-METFIPS, -MET, -STATE,
                         -OCC, -OCC_VAL, -OCC_CATEG,
                         -IND, -IND_VAL, -IND_CATEG)

write.csv(data1, "model_data_clean.csv", row.names = FALSE)
```

Modeling

In this section, we fit the model. Before doing that, however, it is essential to select the appropriate features and the ideal model to fit based on the problem that we are trying to solve for.

Data Loading

We install and load the required packages, followed by reading the model-ready csv file. Further, we convert certain categorical columns so that R considers them as a factor instead of an integer

```
list.of.packages <- c("caret", "glmnet", "tidyverse", "randomForest")
new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
if(length(new.packages)) install.packages(new.packages)

library('tidyverse')
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library('caret')
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library('glmnet')
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
```

```
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8
```

```
library('randomForest')
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(123) # Setting a seed for reproducibility

data <- read.csv("model_data_clean.csv")

data$STATEFIP <- as.factor(data$STATEFIP)
data$SEX <- as.factor(data$SEX)
data$RACE <- as.factor(data$RACE)
data$OCC_CODE <- as.factor(data$OCC_CODE)
data$IND_CODE <- as.factor(data$IND_CODE)
data$CITIZEN <- as.factor(data$CITIZEN)
```

Feature Selection

Once we have the data loaded in, we perform feature selection. In our case, we use the Lasso Regression method to understand which features are not important. Lasso Regression automatically performs feature selection by shrinking some coefficients of features to zero. We ensure that Dummy Variables are created for each categorical feature before running lasso regression and

```
# Split input data to two datasets, one for Log transformed income and other for raw income values
data1 <- data %>% select(-INCTOT)
data2 <- data %>% select(-log_INCTOT)

# Create Train and Test data for both these datasets
splitIndex1 <- createDataPartition(data$log_INCTOT, p = 0.8, list = FALSE)
train1 <- data1[ splitIndex1,]
test1 <- data1[-splitIndex1,]

splitIndex2 <- createDataPartition(data$INCTOT, p = 0.8, list = FALSE)
train2 <- data2[ splitIndex2,]
test2 <- data2[-splitIndex2,]
```

```r
# Define X and Y for all Cases as well as create Dummy Variables for Categorical Features
x1 <- train1[, c("MET_CODE", "STATEFIP", "AGE", "SEX", "RACE", "CITIZEN", "OCC_CODE", "IND_CODE", "EDUC
x1 <- model.matrix(~., data = x1)
y1 <- train1$log_INCTOT

x2 <- train2[, c("MET_CODE", "STATEFIP", "AGE", "SEX", "RACE", "CITIZEN", "OCC_CODE", "IND_CODE", "EDUC
x2 <- model.matrix(~., data = x2)
y2 <- train2$INCTOT

# Fit LASSO Model to perform Feature Selection
fit1 <- cv.glmnet(x1, y1, family="gaussian")
coef_selected1 <- coef(fit1, s="lambda.min")
print(coef_selected1)
```

```
## 150 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept)  9.2807169424
## (Intercept)  .
## MET_CODE     0.0002106139
## STATEFIP4    0.0216144081
## STATEFIP5   -0.0748942153
## STATEFIP6    0.0709584246
## STATEFIP8    0.0522905384
## STATEFIP9    0.1198153903
## STATEFIP10  -0.0125434876
## STATEFIP11   0.2041834733
## STATEFIP12  -0.0521119321
## STATEFIP13   .
## STATEFIP15   .
## STATEFIP16   .
## STATEFIP17   0.0546076128
## STATEFIP18  -0.0137392152
## STATEFIP19   .
## STATEFIP20   .
## STATEFIP21   .
## STATEFIP22  -0.0708479894
## STATEFIP23   .
## STATEFIP24   0.1135998593
## STATEFIP25   0.0794252899
## STATEFIP26   0.0325740766
## STATEFIP27   0.1220598901
## STATEFIP28  -0.2235868097
## STATEFIP29   0.0335774926
## STATEFIP30   0.0334916688
## STATEFIP31   0.0546510618
## STATEFIP32  -0.0528894415
## STATEFIP33   0.0983096685
## STATEFIP34   0.0756824216
## STATEFIP35  -0.0808104030
## STATEFIP36   0.0191823776
## STATEFIP37  -0.0608422978
## STATEFIP38   0.0170593128
## STATEFIP39  -0.0087523331
```

```
## STATEFIP40  -0.1395860391
## STATEFIP41   0.0472878282
## STATEFIP42   0.0230028410
## STATEFIP44   0.0901328362
## STATEFIP45  -0.0679741416
## STATEFIP46  -0.0078002832
## STATEFIP47  -0.0286410382
## STATEFIP48  -0.0298732158
## STATEFIP49   0.0064683061
## STATEFIP50   0.0570714998
## STATEFIP51   0.0119824237
## STATEFIP53   0.1495437072
## STATEFIP54  -0.0354807478
## STATEFIP55  -0.0464689074
## AGE          0.0125996277
## SEX2        -0.2021993291
## RACE200     -0.0993347789
## RACE300     -0.0433241449
## RACE651      0.0105318883
## RACE652     -0.0202073839
## RACE801     -0.1649758062
## RACE802     -0.0439386953
## RACE803      0.0871572906
## RACE804     -0.1438827856
## RACE805     -0.0578753491
## RACE806      .
## RACE807      0.0463344508
## RACE808      0.2953449839
## RACE809      .
## RACE810      .
## RACE811     -0.4123873847
## RACE812      .
## RACE813      0.1028154325
## RACE816      .
## RACE817      .
## RACE819      .
## RACE820     -0.4220689865
## RACE830      0.4049830609
## CITIZEN2    -0.0455321162
## CITIZEN3    -0.0528105584
## CITIZEN4    -0.0676636027
## CITIZEN5    -0.1181049689
## OCC_CODE2   -0.1099284828
## OCC_CODE3    0.0064373151
## OCC_CODE4    .
## OCC_CODE5   -0.2929857386
## OCC_CODE6   -0.3432532355
## OCC_CODE7    0.0808994485
## OCC_CODE8   -0.3453067802
## OCC_CODE9   -0.2562148565
## OCC_CODE10  -0.0330667331
## OCC_CODE11  -0.4721650465
## OCC_CODE12  -0.2309211254
## OCC_CODE13  -0.5307815869
```

```
## OCC_CODE14  -0.5133977186
## OCC_CODE15  -0.5315947433
## OCC_CODE16  -0.2240631495
## OCC_CODE17  -0.3828792459
## OCC_CODE18  -0.5984824745
## OCC_CODE19  -0.3422765678
## OCC_CODE20  -0.2530499926
## OCC_CODE21  -0.4345813200
## OCC_CODE22  -0.4648495646
## IND_CODE2   -0.0262041453
## IND_CODE3    0.2415106961
## IND_CODE4    .
## IND_CODE5    0.0073255674
## IND_CODE6   -0.0222780101
## IND_CODE7    0.1127708032
## IND_CODE8    0.1193878366
## IND_CODE9    0.0740564236
## IND_CODE10   0.1008904694
## IND_CODE11   0.0724777560
## IND_CODE12  -0.0750241164
## IND_CODE13   .
## IND_CODE14  -0.0084171698
## IND_CODE15   0.0195359418
## IND_CODE16  -0.0215698409
## IND_CODE17   0.0605987354
## IND_CODE18   0.1978994495
## IND_CODE19   0.1568444089
## IND_CODE20  -0.1500394527
## IND_CODE21   0.0418165903
## IND_CODE22  -0.1383033009
## IND_CODE23   0.0339773784
## IND_CODE24   0.1658700038
## IND_CODE25   0.2326363404
## IND_CODE26   0.1157002295
## IND_CODE27   0.2473913601
## IND_CODE29   0.1180650986
## IND_CODE30   0.1591036055
## IND_CODE31   .
## IND_CODE32   0.1449572940
## IND_CODE33   0.0182512544
## IND_CODE35   .
## IND_CODE36   0.1173787436
## IND_CODE37   0.0615662743
## IND_CODE38  -0.1146523215
## IND_CODE39   0.0453147356
## IND_CODE40  -0.1382562182
## IND_CODE41  -0.0247423349
## IND_CODE42  -0.0127539997
## IND_CODE43  -0.2179079674
## IND_CODE44  -0.1376543066
## IND_CODE45  -0.1722616666
## IND_CODE46  -0.2016343020
## IND_CODE47  -0.1929582806
## IND_CODE48  -0.0595795954
```

```
## IND_CODE49  -0.1597056513
## IND_CODE50  -0.4007666283
## IND_CODE51   0.0612021354
## EDUC         0.0102533112
## UHRSWORKT    0.0127088746
```

```
selected_features1 <- which(abs(coef_selected1[-1]) != 0)
x1 <- x1[, c(selected_features1)]

fit2 <- cv.glmnet(x2, y2, family="gaussian")
coef_selected2 <- coef(fit2, s="lambda.min")
print(coef_selected2)
```

```
## 150 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept) -72335.56862
## (Intercept)      .
## MET_CODE        13.14425
## STATEFIP4    -3149.25163
## STATEFIP5    -7833.29925
## STATEFIP6     7558.07778
## STATEFIP8     4840.22909
## STATEFIP9    15565.73566
## STATEFIP10   -4055.27650
## STATEFIP11   24807.76377
## STATEFIP12   -3840.68002
## STATEFIP13   -1750.69126
## STATEFIP15   -7530.04360
## STATEFIP16   -3089.21432
## STATEFIP17    6264.59053
## STATEFIP18     -19.27852
## STATEFIP19    3257.22509
## STATEFIP20   14266.47022
## STATEFIP21   -8293.07978
## STATEFIP22  -11476.56451
## STATEFIP23   -4613.52814
## STATEFIP24    5011.13591
## STATEFIP25   10845.28359
## STATEFIP26   -1622.72011
## STATEFIP27    9289.69418
## STATEFIP28  -10861.97211
## STATEFIP29    -604.05765
## STATEFIP30   -1325.95319
## STATEFIP31      68.75162
## STATEFIP32   -1101.60360
## STATEFIP33    6199.61171
## STATEFIP34    5377.30318
## STATEFIP35  -11283.89953
## STATEFIP36    3837.99909
## STATEFIP37       .
## STATEFIP38   -2498.56761
## STATEFIP39   -5702.47764
## STATEFIP40   -8410.62941
## STATEFIP41    1822.22046
```

```
## STATEFIP42     986.88247
## STATEFIP44    -149.81111
## STATEFIP45   -1245.82129
## STATEFIP46  -10612.00376
## STATEFIP47   -4493.44880
## STATEFIP48       .
## STATEFIP49    6025.84581
## STATEFIP50       .
## STATEFIP51    7477.23693
## STATEFIP53   14970.12287
## STATEFIP54   -3771.85411
## STATEFIP55  -10411.09815
## AGE            962.66543
## SEX2        -20213.23861
## RACE200     -10417.36106
## RACE300      -7646.52559
## RACE651       2492.45361
## RACE652       -299.77396
## RACE801     -13080.50503
## RACE802      -6779.57646
## RACE803       7090.01295
## RACE804      -6007.08736
## RACE805     -16577.92414
## RACE806          .
## RACE807          .
## RACE808      38813.19558
## RACE809          .
## RACE810     -14383.95198
## RACE811      -2225.21152
## RACE812     -22096.46447
## RACE813      18442.67728
## RACE816      51183.31993
## RACE817          .
## RACE819      -9368.36183
## RACE820          .
## RACE830      21677.22388
## CITIZEN2     -2965.84639
## CITIZEN3     -5120.66085
## CITIZEN4     -3333.93102
## CITIZEN5     -1953.37690
## OCC_CODE2   -18640.04632
## OCC_CODE3    -6597.11377
## OCC_CODE4   -10242.76033
## OCC_CODE5   -29200.33324
## OCC_CODE6   -38005.57111
## OCC_CODE7    29604.21219
## OCC_CODE8   -27300.59931
## OCC_CODE9   -28945.91911
## OCC_CODE10       .
## OCC_CODE11  -38686.84748
## OCC_CODE12  -23246.34229
## OCC_CODE13  -28493.12788
## OCC_CODE14  -35288.16424
## OCC_CODE15  -29673.53172
```

```
## OCC_CODE16  -17577.41805
## OCC_CODE17  -36705.57867
## OCC_CODE18  -35749.23419
## OCC_CODE19  -34103.18469
## OCC_CODE20  -31028.16491
## OCC_CODE21  -40811.70681
## OCC_CODE22  -40139.23704
## IND_CODE2         .
## IND_CODE3    19951.44600
## IND_CODE4         .
## IND_CODE5    -3802.70671
## IND_CODE6     -985.06232
## IND_CODE7     2463.10926
## IND_CODE8    10194.90853
## IND_CODE9    11354.08121
## IND_CODE10    7171.98742
## IND_CODE11        .
## IND_CODE12   -3958.03045
## IND_CODE13        .
## IND_CODE14   -1169.13523
## IND_CODE15        .
## IND_CODE16    -306.01295
## IND_CODE17   -3109.07517
## IND_CODE18   10854.92275
## IND_CODE19   16296.64865
## IND_CODE20    -228.31144
## IND_CODE21    2312.08853
## IND_CODE22   -5443.31725
## IND_CODE23        .
## IND_CODE24   10816.27804
## IND_CODE25   21280.76995
## IND_CODE26    7172.37658
## IND_CODE27   36266.10622
## IND_CODE29    7768.93033
## IND_CODE30   13045.86588
## IND_CODE31        .
## IND_CODE32   32192.97425
## IND_CODE33    1585.75712
## IND_CODE35   -8937.67855
## IND_CODE36   18535.19710
## IND_CODE37    2227.88321
## IND_CODE38   -5770.59769
## IND_CODE39        .
## IND_CODE40  -19374.16274
## IND_CODE41    -609.95811
## IND_CODE42        .
## IND_CODE43  -14525.36936
## IND_CODE44  -11749.00279
## IND_CODE45  -11157.42876
## IND_CODE46  -18050.25001
## IND_CODE47  -10013.98928
## IND_CODE48   -4546.46899
## IND_CODE49  -16577.46122
## IND_CODE50  -15640.73855
```

```
## IND_CODE51       .
## EDUC           862.96870
## UHRSWORKT     1477.65553
```

```
selected_features2 <- which(abs(coef_selected2[-1]) != 0)
x2 <- x2[, c(selected_features2)]
```

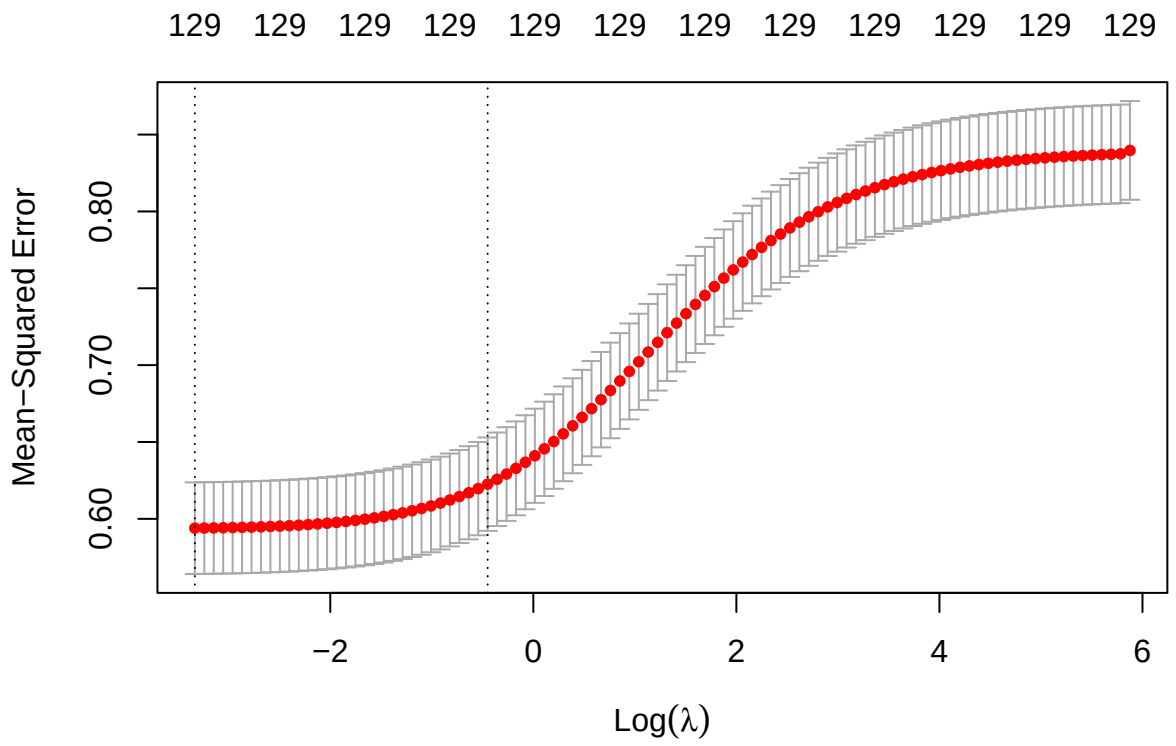
Model Building

For income prediction, we are using RIDGE Regression for prediction. Further, we are considering 2 models, one with the RAW Income variable INCTOT and the second with the LOG transformed Income variable log_INCTOT. By using Ridge regression on both, we try to compare the performance and try to understand which is a better model for our usecase,

Model 1:

Here, we model for the Log Transformed model

```
cv_ridge <- cv.glmnet(as.matrix(x1), y1, alpha = 0, type.measure = "mse", nfolds = 10)

# Plot the CV results to find the optimal lambda
plot(cv_ridge)
```



```r
cv_error <- cv_ridge$cvm

train_error <- mean(cv_error)
validation_error <- min(cv_error)  # This is the minimum error, which corresponds to the validation err

# Best lambda from CV
best_lambda <- cv_ridge$lambda.min
print(paste("Optimal lambda:", best_lambda))
```

```
## [1] "Optimal lambda: 0.0356564821437246"
```

```r
# Fit the model using the best lambda
final_ridge_model <- glmnet(x1, y1, alpha = 0, lambda = best_lambda, nfolds = 10, standardize = TRUE)

# Predict on Test Data
x_test1 <- test1[, c("MET_CODE", "STATEFIP", "AGE", "SEX", "RACE", "CITIZEN", "OCC_CODE", "IND_CODE", "
x_test1 <- model.matrix(~., data = x_test1)
x_test1 <- x_test1[, c(selected_features1)]
y_test1 <- test1$log_INCTOT

predictions <- predict(final_ridge_model, newx = x_test1)
rsq <- 1 - sum((y_test1 - predictions)^2) / sum((y_test1 - mean(y_test1))^2)
mse_test <- mean((y_test1 - predictions)^2)
rmse_test <- sqrt(mse_test)
mae_test <- mean(abs(y_test1 - predictions))

# Print the results
print(paste("Train MSE:", train_error))
```

```
## [1] "Train MSE: 0.715235630941903"
```

```r
print(paste("Validation MSE:", validation_error))
```

```
## [1] "Validation MSE: 0.593914882875985"
```

```r
print(paste("R-squared: ", rsq))
```

```
## [1] "R-squared:  0.267658312012786"
```

```r
print(paste("Test MSE: ", mse_test))
```

```
## [1] "Test MSE:  0.669002228799646"
```

```r
print(paste("Test RMSE: ", rmse_test))
```

```
## [1] "Test RMSE:  0.817925564339229"
```

```r
print(paste("Test MAE: ", mae_test))
```

```
## [1] "Test MAE:  0.469130391910932"
```

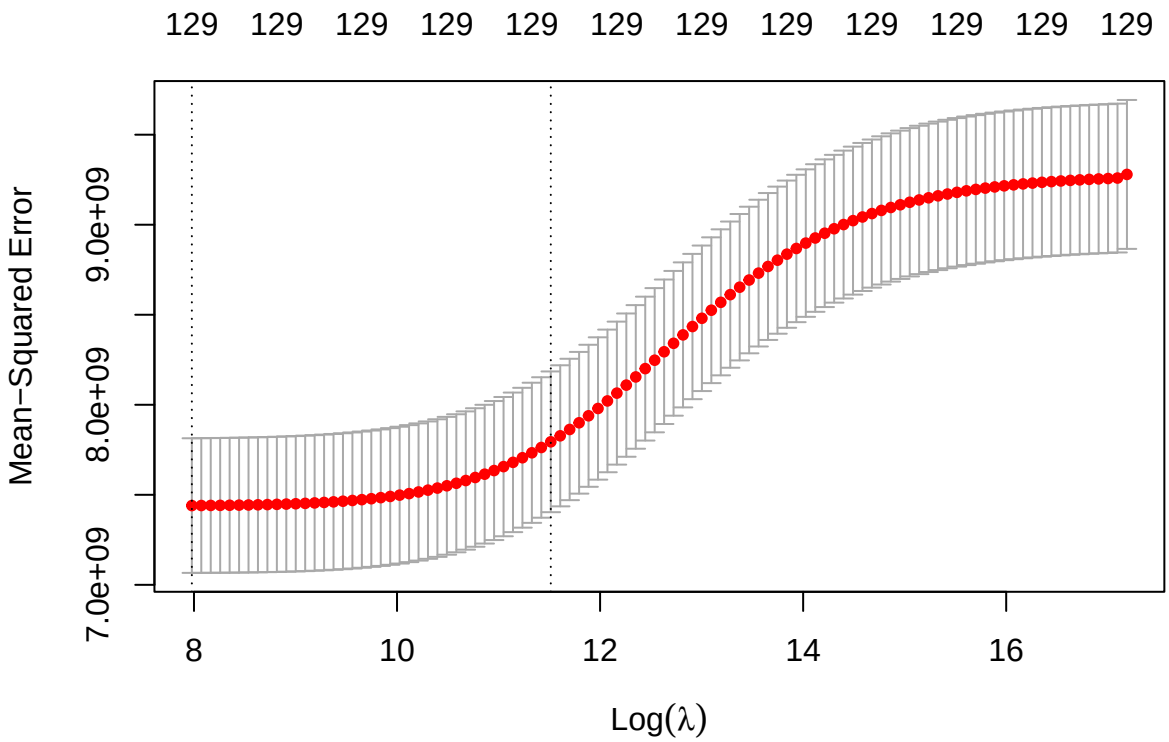
Interpretation:

From the results above, we observe that we have an R^2 value of approximately 0.3 and moderate Train, Validation and Test Error. We can interpret that, on average, when using LOG_INCTOT, our model would be off by ~0.7 units when predicted the Wage of an individual given all other features.

Model 2:

Here we model for the Raw INCTOT variable

```r
cv_ridge2 <- cv.glmnet(as.matrix(x2), y2, alpha = 0, type.measure = "mse", nfolds = 10)

# Plot the CV results to find the optimal lambda
plot(cv_ridge2)
```




```r
cv_error2 <- cv_ridge2$cvm

train_error2 <- mean(cv_error2)
validation_error2 <- min(cv_error2)

# Best lambda from CV
best_lambda2 <- cv_ridge2$lambda.min
print(paste("Optimal lambda:", best_lambda2))
```

```
## [1] "Optimal lambda: 2918.94391295606"
```

```
# Fit the model using the best lambda
final_ridge_model2 <- glmnet(x2, y2, alpha = 0, lambda = best_lambda2, nfolds = 10, standardize = TRUE)

# Predict on Test Data
x_test2 <- test2[, c("MET_CODE", "STATEFIP", "AGE", "SEX", "RACE", "CITIZEN", "OCC_CODE", "IND_CODE", "
x_test2 <- model.matrix(~., data = x_test2)
x_test2 <- x_test2[, c(selected_features2)]
y_test2 <- test2$INCTOT

predictions2 <- predict(final_ridge_model2, newx = x_test2)
rsq2 <- 1 - sum((y_test2 - predictions2)^2) / sum((y_test2 - mean(y_test2))^2)
mse_test2 <- mean((y_test2 - predictions2)^2)
rmse_test2 <- sqrt(mse_test2)
mae_test2 <- mean(abs(y_test2 - predictions2))

# Print the results
print(paste("Train MSE:", train_error2))
```

```
## [1] "Train MSE: 8310636937.84522"
```

```
print(paste("Validation MSE:", validation_error2))
```

```
## [1] "Validation MSE: 7440173342.40493"
```

```
print(paste("R-squared: ", rsq2))
```

```
## [1] "R-squared:  0.20109606465489"
```

```
print(paste("Test MSE: ", mse_test2))
```

```
## [1] "Test MSE:  7032242512.64366"
```

```
print(paste("Test RMSE: ", rmse_test2))
```

```
## [1] "Test RMSE:  83858.4671493801"
```

```
print(paste("Test MAE: ", mae_test2))
```

```
## [1] "Test MAE:  40548.9635521988"
```

Interpretation:

From the results above, we observe that we have an R^2 value of approximately 0.2 and extremely high Train, Validation and Test Error, so much so that they don't quite make sense in the context of the wages that we have in the dataset.

CONCLUSION

Based on above results, it is evident that our Log transformed model performs much better as compared to the raw INCTOT feature for our given dataset.