Predicting Income Level from Demographic and Behavioral Variables

A Comparison of Linear Regression with an Indicator Matrix and Logistic Regression Approaches

Donnie Minnick, Statistical Learning - Fall A 2025

September 06, 2025

# Project Overview

This project evaluates two modeling approaches, linear regression with an indicator matrix and logistic regression, to determine which more reliably classifies individuals as earning above or below $50,000 per year.

Using the UCI Adult Income dataset, the comparison focuses on career stage, marital status, working hours, education level, and investment activity as the explanatory variables.

I compare model performance using practical measures, including accuracy, sensitivity, specificity, and Area Under the ROC Curve (AUC).

My goal is to identify the method that not only provides stronger predictive capability but also offers clearer insight into how these three dimensions of socioeconomic behavior and opportunity contribute to income differences.

## Analysis Question

How do linear regression with an indicator matrix and logistic regression compare in their ability to classify individuals as earning above or below $50,000 per year, based on career stage, marital status, working hours, education level, and investment activity and other demographic and behavioral variables, in terms of accuracy, sensitivity, specificity, and AUC?

## Data Suitability

The UCI Adult Income dataset is well suited for this analysis based on the following characteristics:

**Binary Target Variable:** The income label is already dichotomized as <=50K and >50K, making it ideal for classification tasks.

**Relevant Predictors:** Career stage can be proxied through variables such as age, occupation, and years of work experience. Education level is directly available in multiple forms (years of education and categorical attainment). And investment activity can be approximated through features like capital gains and capital losses, which provide signals of financial activity beyond earned income. Marital status and hours worked provide a view into socioeconomic dynamics influencing income level.

**Mixed Data Types:** These predictors combine categorical (education level, marital status, occupation) and continuous (capital gains, capital losses, age) data, allowing a comparison of how well each modeling method handles different variable types.

**Real-World Relevance:** The selected variables represent core levers of socioeconomic advancement, human capital (education), labor market position (career stage), and financial behavior (investment activity). Evaluating their relationship to income provides insights into drivers of upward mobility.

## Candidate Models

### Linear Regression with an Indicator Matrix

Linear regression, when applied with a binary indicator response, can be used to estimate probabilities of class membership. Although not traditionally designed for classification, it provides a straightforward benchmark and can reveal how continuous predictors like age or hours worked relate linearly to income. However, it may yield predictions outside the 0–1 range and does not naturally account for the probabilistic nature of binary outcomes.

### Logistic Regression

Logistic regression is a standard method for binary classification, modeling the log-odds of the outcome as a linear combination of predictors. It constrains predicted values between 0 and 1 and provides interpretable coefficients in terms of odds ratios. It is particularly well-suited for this problem and serves as the conventional baseline for evaluating newer or more complex classifiers.

### Rationale

Placing these two approaches side by side highlights the importance of choosing models that align with the data structure and analysis question. By comparing their performance on accuracy, sensitivity, specificity, and AUC, this analysis will demonstrate the trade-offs between a general-purpose regression method and a model purpose-built for classification.

### Baseline Expectations

Before conducting the analysis, it is important to establish expectations about how the candidate models are likely to perform:

**Linear Regression Benchmark:** Linear regression with an indicator matrix may provide a useful baseline, but its predictions can extend outside the valid probability range and may not align as well with classification thresholds. Accuracy may be reasonable, but sensitivity and specificity are likely to suffer compared to logistic regression.

**Logistic Regression Advantage:** Because logistic regression is specifically designed for binary classification, it is expected to outperform linear regression in terms of calibration and overall predictive reliability. Its ability to constrain predictions between 0 and 1 aligns naturally with the problem structure.

**Comparative Outlook:** Logistic regression is anticipated to deliver higher AUC and more balanced classification metrics, while linear regression may illustrate the pitfalls of applying a general-purpose model to a classification task. This contrast should highlight the importance of model choice in predictive analytics.

## Github Repo and Source Data File

All project files are maintained in [this Github repository](https://github.com/dtminnick/income).

The UCI Adult Income dataset and related information are available for download from the [UCI archive site](https://archive.ics.uci.edu/dataset/2/adult).

## Code Libraries

My analysis leverages the following R packages: caret for model training and evaluation, dplyr and tidyr for data manipulation, ggplot2 for plots, knitr for table formatting, and pROC for ROC/AUC analysis.

Customized R functions enable comparison of model coefficients and residuals, evaluation of predictor associations and multicollinearity checks.

Code chunks are presented in this R markdown document alongside analysis to document implementation of analysis, model creation, and evaluation.

library("caret")

## Loading required package: ggplot2

## Loading required package: lattice

library("dplyr")

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library("ggplot2")  
library("knitr")  
library("pROC")

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

library("tidyr")  
  
source("../R/compare\_model\_coefficients.R")  
source("../R/compare\_model\_residuals.R")  
source("../R/check\_multicollinearity\_factors.R")  
source("../R/test\_predictor\_associations.R")

# Data Exploration, Cleaning and Transformation

Exploratory analysis focuses on understanding five variables in the source data that will be transformed for analysis purposes. Three of the variables have a categorical version and numerical version to aid exploratory analysis.

Load the income data and select in-scope variables. Create a numeric version of the response variable.

income <- readRDS("../data/income.rds") %>%  
 select(income,   
 age,  
 marital\_status,  
 hours\_per\_week,  
 education,   
 capital\_gain,   
 capital\_loss) %>%  
 mutate(income\_num = if\_else(income == "<=50K", 0, 1))

## Missing Values

Generate a report summarizing missing values in the dataset at both the column and row level. Compute the total number and percentage of rows containing any missing values. And combine summaries into a single table for easy inspection and reporting.

# Create column level summary.  
  
col\_missing <- income %>%  
 summarise(across(everything(), ~ sum(is.na(.)))) %>%  
 pivot\_longer(cols = everything(),  
 names\_to = "variable",  
 values\_to = "missing\_count") %>%  
 mutate(missing\_percent = round(missing\_count / nrow(income), 2))  
  
# Create row level summary.  
  
row\_missing <- tibble(variable = "rows\_with\_missing\_data",  
 missing\_count = sum(!complete.cases(income)),  
 missing\_percent = round(sum(!complete.cases(income)) / nrow(income), 2))  
  
# Combine summaries.  
  
missing\_report <- bind\_rows(col\_missing, row\_missing)  
  
# Generate formatted report.  
  
kable(missing\_report,  
 col.names = c("Variable", "Missing Count", "Missing Percent"),  
 format.args = list(big.mark = ","),  
 align = c("l", "r", "r"))

| Variable | Missing Count | Missing Percent |
| --- | --- | --- |
| income | 0 | 0 |
| age | 0 | 0 |
| marital\_status | 0 | 0 |
| hours\_per\_week | 0 | 0 |
| education | 0 | 0 |
| capital\_gain | 0 | 0 |
| capital\_loss | 0 | 0 |
| income\_num | 0 | 0 |
| rows\_with\_missing\_data | 0 | 0 |

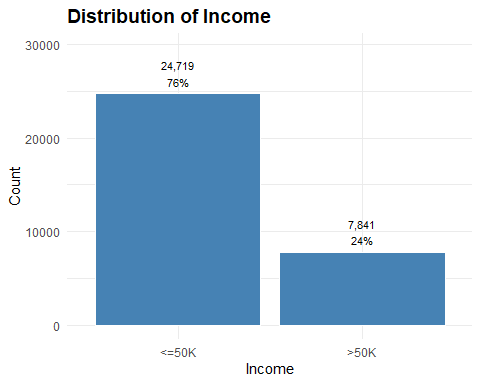
The table confirms there are no rows with missing data in the reported rows and columns. The source dataset contains 32,560 observations.

## Income (Response)

income is a binary categorical variable indicating whether an individual’s annual income exceeds $50,000. It has two levels: <=50K for those earning $50,000 or less, and >50K for those earning more than $50,000.

income\_num is a numeric variable that corresponds to income, i.e. 0 represents the <=50k category and 1 represents the >50k category.

# Generate a summary of the income variable.  
  
income\_summary <- income %>%  
 group\_by(income) %>%  
 summarise(entries = n()) %>%  
 mutate(percent = round(entries / sum(entries), 2))  
  
# Plot the distribution of values.  
  
ggplot(income\_summary, aes(x = income, y = entries)) +  
 geom\_col(fill = "steelblue", color = "white") +  
 geom\_text(aes(label = paste0(scales::comma(entries), "\n", round(percent \* 100, 0), "%")),  
 vjust = -0.3, size = 3.0) +  
 labs(title = "Distribution of Income",  
 x = "Income",  
 y = "Count") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 0, hjust = 0.5),  
 plot.title = element\_text(size = 14, face = "bold")) +  
 expand\_limits(y = max(income\_summary$entries) \* 1.2)



This distribution is a classic case of class imbalance and it’s substantial: 76% of individuals fall into the <=50K category, while only 24% belong to the >50K category.

Given the potential baseline accuracy trap, i.e. because the <=50K class dominates the dataset at 76%, I can achieve high accuracy simply by predicting the majority class every time. But this doesn’t provide a meaningful model.

I need to also measure how well the models detect the minority class, ensure that I am not over-predicting the majority class, and generate models with overall discriminatory power. I will use weights as a means to balance the classes before training the models.

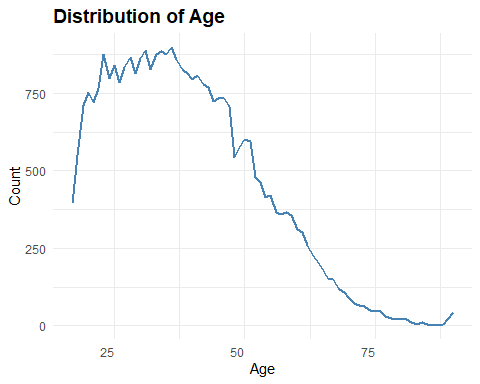
Encode factor levels for income, using <=50K as the reference level.

# Make response variable a factor.  
  
income <- income %>%  
 mutate(income = factor(income, levels = c("<=50K", ">50K")),  
 income\_num = factor(income\_num, levels = c(0, 1)))

## Age

age is a continuous numeric variable, ranging from 17 to 90. The following plot generates a summary of the age variable.

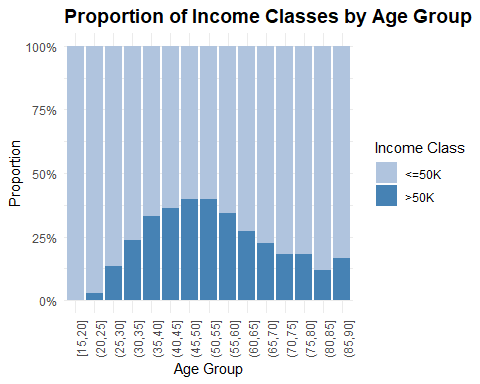
# Generate a summary of the age variable.  
  
age\_summary <- income %>%  
 group\_by(age) %>%  
 summarise(entries = n()) %>%  
 mutate(percent = round(entries / sum(entries), 2))  
  
# Plot the distribution of values.  
  
ggplot(age\_summary, aes(x = age, y = entries)) +  
 geom\_line(linewidth = 1, color = "steelblue") +  
 labs(title = "Distribution of Age",  
 x = "Age",  
 y = "Count") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 0, hjust = 1),  
 plot.title = element\_text(size = 14, face = "bold"))



This distribution reflects what we’d expect from a working-age population in the US: a peak around ages 20-30, reflecting a large cohort entering the workforce, a gradual decline beginning at age 37-38 through age 55, typical of aging out of peak earning years or shifting to preparation for retirement, and dropoff after ages 60-50, reflecting retirement and reduced representation.

Age is likely nonlinear in its relationship to income class. Use binning to confirm this dynamic.

# Create age bins.  
  
income\_age <- income %>%  
 mutate(age\_bin = cut(age, breaks = seq(15, 90, by = 5), include.lowest = TRUE))  
  
# Calculate proportions within each age bin and income group.  
  
age\_income\_prop <- income\_age %>%  
 group\_by(age\_bin, income) %>%  
 summarise(count = n(), .groups = "drop") %>%  
 group\_by(age\_bin) %>%  
 mutate(prop = count / sum(count))  
  
# Generate plot.  
  
ggplot(age\_income\_prop, aes(x = age\_bin, y = prop, fill = income)) +  
 geom\_bar(stat = "identity", position = "fill") +  
 scale\_fill\_manual(values = c("<=50K" = "lightsteelblue", ">50K" = "steelblue")) +  
 scale\_y\_continuous(labels = scales::percent\_format()) +  
 labs(title = "Proportion of Income Classes by Age Group",  
 x = "Age Group",  
 y = "Proportion",  
 fill = "Income Class") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 90, hjust = 1),  
 plot.title = element\_text(size = 14, face = "bold"))



This plot shows that income probability doesn’t increase linearly with age; rather, it peaks mid-career and then drops. I’ll use a binning strategy to model this non-monotonic pattern in a career\_stage variable, i.e. instead of assuming that each year of age adds the same effect, binning allows me to treat age as a set of behavioral groups. To better capture this pattern, I will engineer a career stage feature.

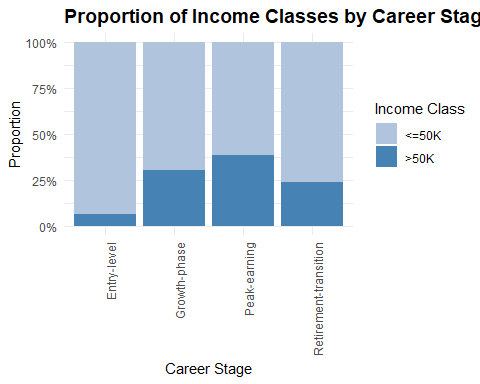
### Career Stage

The career\_stage feature will contain four groups: 1) entry-level (ages 17-30), 2) growth phase (ages 31-45), 3) peak earning (ages 46-60), and 4) retirement transition (ages 61+). Encode factor levels for career stage, using Entry-level as the reference level.

# Create career\_stage variable.  
  
income <- income %>%  
 mutate(career\_stage = case\_when(  
 age < 31 ~ "Entry-level",  
 age >= 21 & age < 45 ~ "Growth-phase",  
 age >= 36 & age < 61 ~ "Peak-earning",  
 age >= 61~ "Retirement-transition"))  
  
# Make variable a factor.  
  
income <- income %>%  
 mutate(career\_stage = factor(career\_stage,   
 ordered = TRUE,  
 levels = c("Entry-level",   
 "Growth-phase",   
 "Peak-earning",   
 "Retirement-transition")))

Visually confirm the non-linear effects of career stage on income classification.

# Group by career stage.  
  
career\_stage\_prop <- income %>%  
 group\_by(career\_stage, income) %>%  
 summarise(count = n(), .groups = "drop") %>%  
 group\_by(career\_stage) %>%  
 mutate(prop = count / sum(count))  
  
# Generate plot.  
  
ggplot(career\_stage\_prop, aes(x = career\_stage, y = prop, fill = income)) +  
 geom\_bar(stat = "identity", position = "fill") +  
 scale\_fill\_manual(values = c("<=50K" = "lightsteelblue", ">50K" = "steelblue")) +  
 scale\_y\_continuous(labels = scales::percent\_format()) +  
 labs(title = "Proportion of Income Classes by Career Stage",  
 x = "Career Stage",  
 y = "Proportion",  
 fill = "Income Class") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 90, hjust = 1),  
 plot.title = element\_text(size = 14, face = "bold"))



To capture this nonlinear effect, I will use a second-degree orthogonal polynomial transformation of the numeric career stage variable as poly(career\_stage\_num, 2). It will model both linear and quadratic trends while avoiding multicollinearity.

There is a single inflection point in this distribution. The chart shows a rise in >50K income class from Entry-level to Peak-earning, followed by a decline in Retirement-transition. That’s a classic parabolic shape with one turning point which a quadratic models naturally.

This should improve numerical stability and interpretability of regression coefficients, especially when modeling non-monotonic relationships (e.g., mid-career income peaks followed by retirement declines).

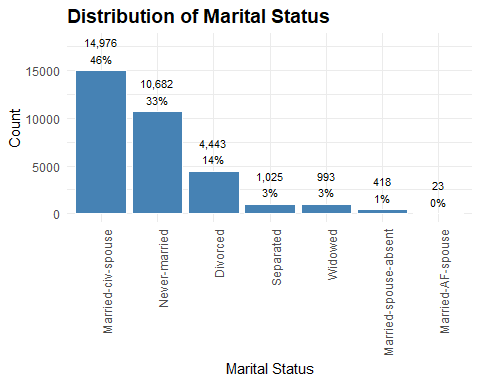
### Marital Status

The marital status variable captures an individual’s relationship status and is a categorical feature with the following levels:

* Married-civ-spouse: Legally married and living with spouse,
* Married-AF-spouse: Married to a spouse in the armed forces,
* Divorced: Legally separated after marriage,
* Separated: Still legally married but not living together,
* Widowed: Spouse has passed away, and
* Never-married.

Generate a summary of marital\_status.

# Generate summary.  
  
marital\_status\_summary <- income %>%  
 group\_by(marital\_status) %>%  
 summarise(entries = n(), .groups = "drop") %>%  
 mutate(percent = round(entries / sum(entries), 2))  
  
# Plot distribution of values.  
  
ggplot(marital\_status\_summary, aes(x = forcats::fct\_reorder(marital\_status, entries, .desc = TRUE), y = entries)) +  
 geom\_col(fill = "steelblue", color = "white") +  
 geom\_text(aes(label = paste0(scales::comma(entries), "\n", round(percent \* 100, 0), "%")),  
 vjust = -0.3, size = 3.0) +  
 labs(title = "Distribution of Marital Status",  
 x = "Marital Status",  
 y = "Count") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 90, hjust = 1),  
 plot.title = element\_text(size = 14, face = "bold")) +  
 expand\_limits(y = max(marital\_status\_summary$entries) \* 1.2)



Married-civ-spouse dominates at 46%. Never-married follows at 33%. The rest of the categories are smaller slices, but potentially behaviorally distinct.

Marital status could be a strong socioeconomic signal when predicting income, e.g. married individuals may benefit from dual incomes or household stability. Never-married or separated individuals might reflect different life stages or economic pressures.

These categories are not ordinal, e.g. widowed or married is not more or less than divorced.

Collapsing sparse categories may improve model stability and interpretability, e.g. married, non-traditional married, previously married, never married.

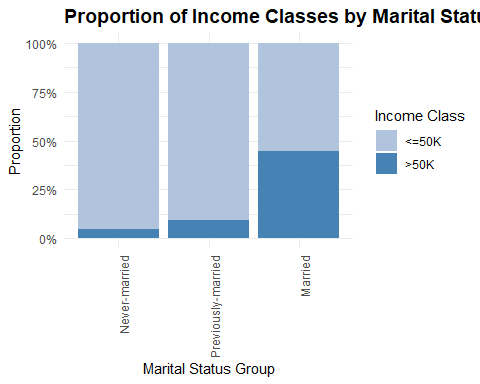
### Marital Status Group

Marital status group collapses sparse categories into broader groups for model stability and interpretability.

# Create marital status group variable  
  
income <- income %>%  
 mutate(marital\_status\_group = case\_when(  
 marital\_status %in% c("Married-civ-spouse",   
 "Married-AF-spouse") ~ "Married",  
 marital\_status %in% c("Divorced",   
 "Separated",   
 "Married-spouse-absent",   
 "Widowed") ~ "Previously-married",  
 marital\_status == "Never-married" ~ "Never-married"))  
  
# Make variable a factor.  
  
income <- income %>%  
 mutate(marital\_status\_group = factor(marital\_status\_group,   
 ordered = FALSE,  
 levels = c("Never-married",   
 "Previously-married",   
 "Married")))

Show distribution of income by marital status group.

# Group by marital group.  
  
marital\_status\_group\_prop <- income %>%  
 group\_by(marital\_status\_group, income) %>%  
 summarise(count = n(), .groups = "drop") %>%  
 group\_by(marital\_status\_group) %>%  
 mutate(prop = count / sum(count))  
  
# Generate plot.  
  
ggplot(marital\_status\_group\_prop, aes(x = marital\_status\_group, y = prop, fill = income)) +  
 geom\_bar(stat = "identity", position = "fill") +  
 scale\_fill\_manual(values = c("<=50K" = "lightsteelblue", ">50K" = "steelblue")) +  
 scale\_y\_continuous(labels = scales::percent\_format()) +  
 labs(title = "Proportion of Income Classes by Marital Status Group",  
 x = "Marital Status Group",  
 y = "Proportion",  
 fill = "Income Class") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 90, hjust = 1),  
 plot.title = element\_text(size = 14, face = "bold"))



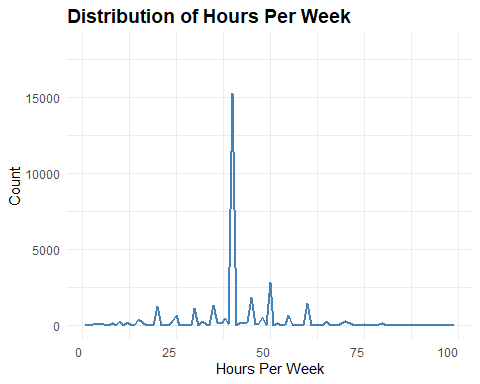
Married individuals show the highest proportion of income earners in the >50K category. This suggests that marriage may correlate with financial stability, dual-income households, or career maturity.

Previously-married and Never-married groups lean heavily toward the <=50K category, indicating lower income prevalence.

### Hours Per Week

Hours per week represents the number of hours an individual reports working per week in their primary job.

hours\_per\_week\_summary <- income %>%  
 group\_by(hours\_per\_week) %>%  
 summarise(entries = n(), .groups = "drop") %>%  
 mutate(percent = round(entries / sum(entries), 2))  
  
ggplot(hours\_per\_week\_summary, aes(x = hours\_per\_week, y = entries)) +  
 geom\_line(linewidth = 1, color = "steelblue") +  
 labs(title = "Distribution of Hours Per Week",  
 x = "Hours Per Week",  
 y = "Count") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 0, hjust = 1),  
 plot.title = element\_text(size = 14, face = "bold")) +  
 expand\_limits(y = max(hours\_per\_week\_summary$entries) \* 1.2)



There is a prominent spike at 40 hours, which can be considered a classic full-time benchmark in the US. It dominates the dataset, reflecting standard employment contracts. There are smaller peaks at 20, 30, 50 and 60 hours, suggesting common part-time and overtime thresholds. These are potentially tied to specific industries or roles. The jagged, irregular tails hint at self-reported data or job-specific norms, e.g. self-employment or gig work.

### Hours Group

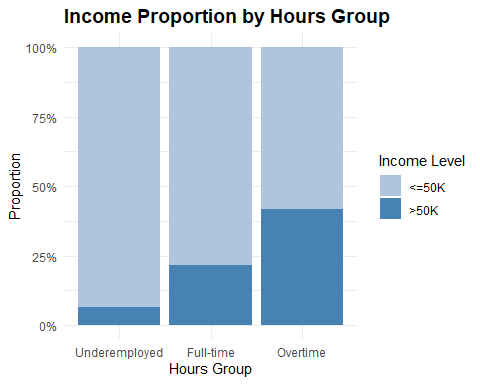
The hours group variable is a derived categorical feature that segments individuals based on their reported weekly work hours:

* Underemployed: Works fewer than 30 hours per week,
* Full-time: Works between 30 and 45 hours per week, and
* Overtime: Works more than 45 hours per week.

# Create hours group variable.  
  
income <- income %>%  
 mutate(hours\_group = case\_when(  
 hours\_per\_week < 30 ~ "Underemployed",  
 hours\_per\_week >= 30 & hours\_per\_week <= 45 ~ "Full-time",  
 hours\_per\_week > 45 ~ "Overtime"))  
  
# Make variable a factor.  
  
income <- income %>%  
 mutate(hours\_group = factor(hours\_group,  
 ordered = TRUE,  
 levels = c("Underemployed",   
 "Full-time",   
 "Overtime")))

Show proportion of high earners by hours group.

hours\_group\_prop <- income %>%  
 group\_by(hours\_group, income) %>%  
 summarise(count = n(), .groups = "drop") %>%  
 group\_by(hours\_group) %>%  
 mutate(prop = count / sum(count))  
   
ggplot(hours\_group\_prop, aes(x = hours\_group, y = prop, fill = income)) +  
 geom\_bar(stat = "identity", position = "fill") +  
 scale\_y\_continuous(labels = scales::percent\_format()) +  
 scale\_fill\_manual(values = c("<=50K" = "lightsteelblue", ">50K" = "steelblue")) +  
 labs(  
 title = "Income Proportion by Hours Group",  
 x = "Hours Group",  
 y = "Proportion",  
 fill = "Income Level"  
 ) +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 0, hjust = 0.5),  
 plot.title = element\_text(size = 14, face = "bold"))

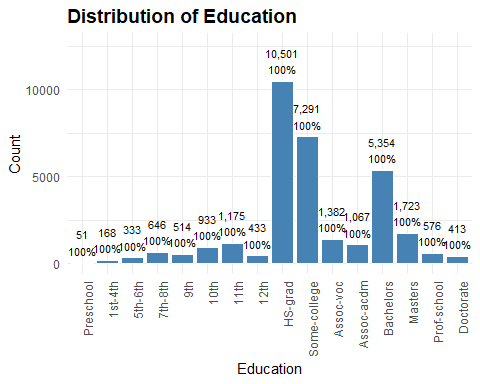


The underemployed group is overwhelmingly low income and likely includes part-time, seasonal or precarious workers. The full-time group is majority <=$50k but has a noticeable uptick in high earners. This suggests that full-time work alone isn’t a guarantee of higher income. The overtime group has the most balanced distribution; it probably includes a mix of skilled labor, self-employed individuals and salaried workers with performance incentives.

## Education

Education is a categorical variable with 16 levels, ranging from Preschool through advanced degrees like Doctorate` and Prof-school.

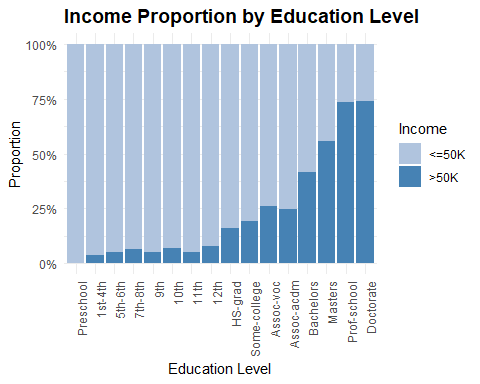
# Make variable a factor.  
  
income <- income %>%  
 mutate(education = factor(education,   
 levels = c("Preschool",   
 "1st-4th",   
 "5th-6th",   
 "7th-8th",   
 "9th",   
 "10th",   
 "11th",   
 "12th",  
 "HS-grad",   
 "Some-college",   
 "Assoc-voc",   
 "Assoc-acdm",  
 "Bachelors",  
 "Masters",  
 "Prof-school",   
 "Doctorate")))  
  
education\_summary <- income %>%  
 group\_by(education) %>%  
 summarise(entries = n(), .groups = "drop") %>%  
 group\_by(education) %>%  
 mutate(percent = round(entries / sum(entries), 2))  
  
ggplot(education\_summary, aes(x = education, y = entries)) +  
 geom\_col(fill = "steelblue", color = "white") +  
 geom\_text(aes(label = paste0(scales::comma(entries), "\n", round(percent \* 100, 0), "%")),  
 vjust = -0.3, size = 3.0) +  
 labs(title = "Distribution of Education",  
 x = "Education",  
 y = "Count") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 90, hjust = 1),  
 plot.title = element\_text(size = 14, face = "bold")) +  
 expand\_limits(y = max(education\_summary$entries) \* 1.2)



High school graduates dominate the sample at 32%; this aligns with national trends. Some-college and Bachelors categories together make up 40% of the sample, indicating strong representation. Advanced degrees are relatively rare; combined they account for ~7% of the sample, which could limit model ability to generalize to highly educated groups. Low education levels are sparse; these are likely to be older adults or immigrants with limited formal schooling.

Plot income proportions by education.

# Group by education.  
  
income\_education\_prop <- income %>%  
 group\_by(education, income) %>%  
 summarise(entries = n(), .groups = "drop") %>%  
 group\_by(education) %>%  
 mutate(prop = entries / sum(entries))  
  
# Plot summary.  
  
ggplot(income\_education\_prop,   
 aes(x = education,   
 y = prop,   
 fill = income)) +  
 geom\_bar(stat = "identity", position = "stack") +  
 scale\_fill\_manual(values = c("<=50K" = "lightsteelblue", ">50K" = "steelblue")) +  
 scale\_y\_continuous(labels = scales::percent\_format()) +  
 labs(title = "Income Proportion by Education Level",  
 x = "Education Level",  
 y = "Proportion",  
 fill = "Income") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 90, hjust = 1),  
 plot.title = element\_text(size = 14, face = "bold"))



There’s an unmistakable pattern evident in this chart: the gradient from low education levels to advanced degrees is a textbook example of socioeconomic stratification. Still its surprising to see how nonlinear the education payoff is in income potential.

The proportion of individuals earning >50K doesn’t increase at a constant rate across education levels. Instead, it jumps more sharply at certain thresholds, especially from HS-grad to Bachelors, and again from Masters to Doctorate.

High income really starts to dominate at the Bachelor level and is an inflection point for income potential. Advanced degrees show the highest income potential, but this comes with investment of time and potentially debt as well.

I’ll use ordinal coding to treat education as a ranked factor to capture the income gradient. I’ll create a separate education\_group variable for this as part of transformations.

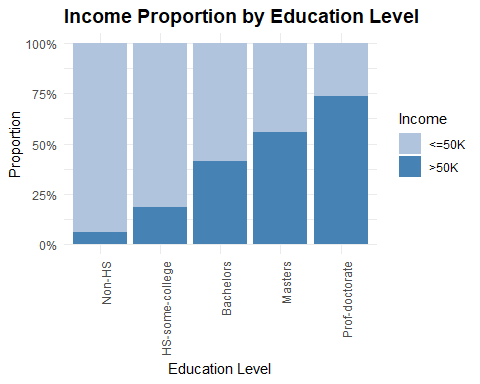
### Education Group

Create a separate education\_group variable.

# Create education level variable.  
  
income <- income %>%  
 mutate(education\_group = case\_when(  
 education %in% c("Preschool", "1st-4th", "5th-6th", "7th-8th", "9th",   
 "10th", "11th", "12th") ~ "Non-HS",  
 education %in% c("HS-grad", "Some-college",   
 "Assoc-voc", "Assoc-acdm") ~ "HS-some-college",  
 education %in% c("Bachelors") ~ "Bachelors",  
 education %in% c("Masters") ~ "Masters",  
 education %in% c("Prof-school", "Doctorate") ~ "Prof-doctorate"))  
  
# Make variable a factor.  
  
income <- income %>%  
 mutate(education\_group = factor(education\_group,  
 ordered = TRUE,  
 levels = c("Non-HS",   
 "HS-some-college",   
 "Bachelors",  
 "Masters",  
 "Prof-doctorate")))

Plot proportion of income by education group.

# Group by education group.  
  
education\_group\_prop <- income %>%  
 group\_by(education\_group, income) %>%  
 summarise(entries = n(), .groups = "drop") %>%  
 group\_by(education\_group) %>%  
 mutate(prop = entries / sum(entries))  
  
# Plot summary.  
  
ggplot(education\_group\_prop,   
 aes(x = education\_group,   
 y = prop,   
 fill = income)) +  
 geom\_bar(stat = "identity", position = "stack") +  
 scale\_fill\_manual(values = c("<=50K" = "lightsteelblue", ">50K" = "steelblue")) +  
 scale\_y\_continuous(labels = scales::percent\_format()) +  
 labs(title = "Income Proportion by Education Level",  
 x = "Education Level",  
 y = "Proportion",  
 fill = "Income") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 90, hjust = 1),  
 plot.title = element\_text(size = 14, face = "bold"))



The income jump from Bachelors to Masters to Prof-doctorate is not linear. The gains accelerate, suggesting credential thresholds, e.g., graduate degrees unlocking higher-paying roles, and labor market segmentation, e.g., professional degrees tied to elite occupations.

I will use a natural cubic splines to model this non-linear relationship between your ordinal education variable and the outcome, i.e. ns(as.numeric(education\_group), df = 3).

## Capital Gain/Loss

In the UCI Adult Income dataset, the capital gain and capital loss variables don’t represent individual investment transactions like you’d see in a brokerage account. Instead, they are annual amounts reported on tax returns.

Here’s what they represent in practice:

**Capital Gain:** The total taxable profit someone reported in a year from selling assets (stocks, bonds, property, etc.) for more than they paid.

**Capital Loss:** The total deductible loss someone reported in a year from selling assets for less than they paid. Tax rules allow limited reporting of such losses.

In the dataset, most people have zeros for both variables, meaning they didn’t report any gains or losses that year. Nonzero values are relatively rare but signal engagement with investment activity beyond wages.

### Limitation of Capital Gain/Loss as a Predictors

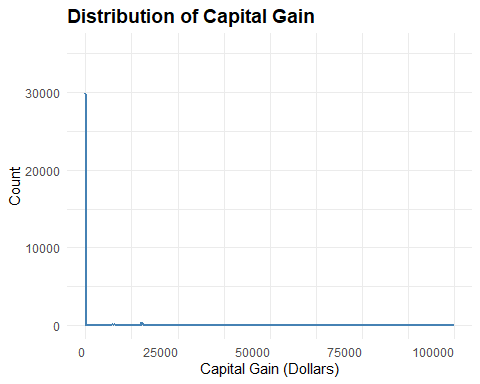
While capital gain and capital loss provide useful signals of investment activity, they capture only realized transactions (profits or losses from assets actually sold). They do not account for asset ownership or wealth holdings that have not been sold. For example, someone may hold significant investments in real estate or retirement accounts but report zero gains or losses if they did not sell anything that year.

This means the variables reflect investment activity and non-wage income, not necessarily investment capacity or wealth accumulation.

### Capital Gain

Generate capital gain summary.

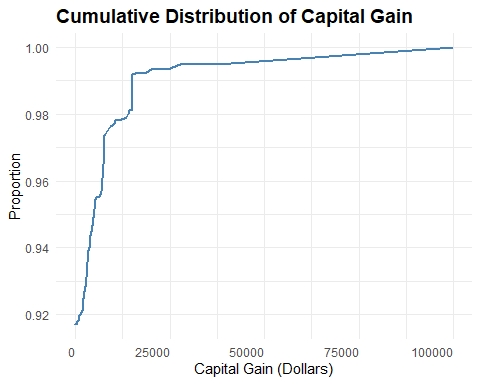
# Generate summary.  
  
capital\_gain\_summary <- income %>%  
 group\_by(capital\_gain) %>%  
 summarise(entries = n(), .groups = "drop") %>%  
 mutate(percent = round(entries / sum(entries), 2))  
  
# Plot summary.  
  
ggplot(capital\_gain\_summary, aes(x = capital\_gain, y = entries)) +  
 geom\_line(linewidth = 1, color = "steelblue") +  
 labs(title = "Distribution of Capital Gain",  
 x = "Capital Gain (Dollars)",  
 y = "Count") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 0, hjust = 1),  
 plot.title = element\_text(size = 14, face = "bold")) +  
 expand\_limits(y = max(capital\_gain\_summary$entries) \* 1.2)



Capital gains are heavily zero-inflated, and that skew makes it a prime candidate for binary transformation. Most individuals have zero capital gain, so the continuous values only apply to a small subset. Continuous skewed variables can distort coefficients or inflate variance in linear models. A binary flag captures the presence of investment activity, which may correlate with income or occupation.

Plot cumulative distribution of capital gains.

# Group by gain.  
  
capital\_gain\_prop <- capital\_gain\_summary %>%  
 arrange(capital\_gain) %>%  
 mutate(cumulative = cumsum(entries) / sum(entries))  
  
# Plot proportion.  
  
ggplot(capital\_gain\_prop, aes(x = capital\_gain, y = cumulative)) +  
 geom\_line(linewidth = 1, color = "steelblue") +  
 labs(title = "Cumulative Distribution of Capital Gain",  
 x = "Capital Gain (Dollars)",  
 y = "Proportion") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 0, hjust = 1),  
 plot.title = element\_text(size = 14, face = "bold"))



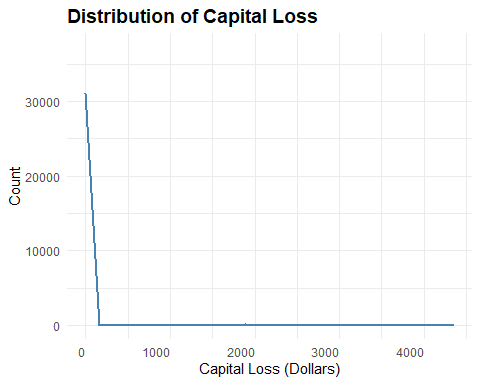
The cumulative distribution shows that over 92% of individuals report zero or minimal capital gains, with only a small fraction earning substantial amounts (e.g., >10,000).

Capital gain is a strong signal of financial activity and wealth accumulation, but it’s concentrated among a small subset of the population.

### Capital Loss

Generate capital loss summary.

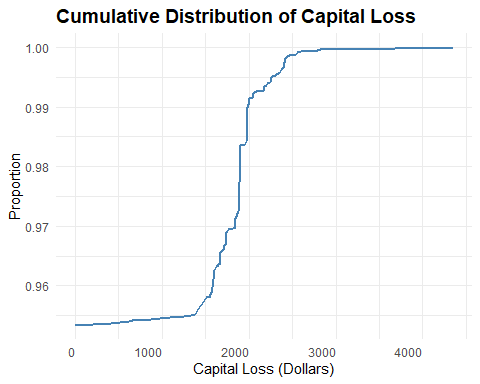
# Generate summary.  
  
capital\_loss\_summary <- income %>%  
 group\_by(capital\_loss) %>%  
 summarise(entries = n(), .groups = "drop") %>%  
 mutate(percent = round(entries / sum(entries), 2))  
  
# Plot summary.  
  
ggplot(capital\_loss\_summary, aes(x = capital\_loss, y = entries)) +  
 geom\_line(linewidth = 1, color = "steelblue") +  
 labs(title = "Distribution of Capital Loss",  
 x = "Capital Loss (Dollars)",  
 y = "Count") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 0, hjust = 1),  
 plot.title = element\_text(size = 14, face = "bold")) +  
 expand\_limits(y = max(capital\_loss\_summary$entries) \* 1.2)



Capital loss is also heavily zero-inflated.

Plot cumulative distribution of capital losses.

# Group by loss.  
  
capital\_loss\_prop <- capital\_loss\_summary %>%  
 arrange(capital\_loss\_summary) %>%  
 mutate(cumulative = cumsum(entries) / sum(entries))  
  
# Plot proportion.  
  
 ggplot(capital\_loss\_prop, aes(x = capital\_loss, y = cumulative)) +  
 geom\_line(linewidth = 1, color = "steelblue") +  
 labs(title = "Cumulative Distribution of Capital Loss",  
 x = "Capital Loss (Dollars)",  
 y = "Proportion") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 0, hjust = 1),  
 plot.title = element\_text(size = 14, face = "bold"))



Over 96% of individuals report zero or minimal capital loss. A steep rise in the cumulative distribution between 1,000 and 2,000 suggests that most reported losses fall within this range. Very few individuals report losses above 2,000.

Combining capital gain and capital loss into a single binary variable reduces two sparse, skewed variables into one binary flag. It preserves the signal that captures whether the individual has any investment-related income or loss. It is also easier to explain and visualize in behavioral segmentation or fairness audits.

### Has Investment Activity

Create a single investment activity flag using capital gain and capital loss.

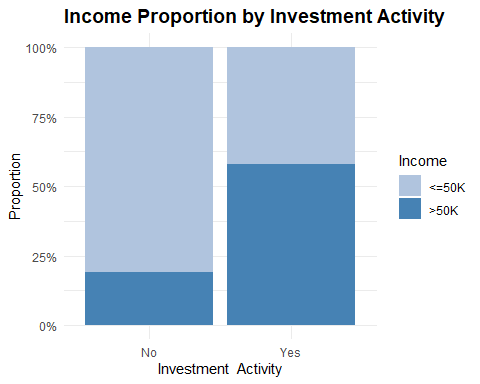
# Create investment activity variable.  
  
income <- income %>%  
 mutate(has\_investment\_activity = if\_else(capital\_gain > 0 | capital\_loss > 0, "Yes", "No"))  
  
# Make variable a factor.  
  
income <- income %>%  
 mutate(has\_investment\_activity = factor(has\_investment\_activity,  
 ordered = FALSE,  
 levels = c("No", "Yes")))

Check distribution of the variable.

# Group by activity.  
  
investment\_by\_income <- income %>%  
 group\_by(has\_investment\_activity, income) %>%  
 summarise(entries = n()) %>%  
 mutate(percent = round(entries / sum(entries), 2))

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

# Plot distribution.  
  
ggplot(investment\_by\_income,   
 aes(x = has\_investment\_activity,   
 y = percent,   
 fill = income)) +  
 geom\_bar(stat = "identity", position = "stack") +  
 scale\_fill\_manual(values = c("<=50K" = "lightsteelblue", ">50K" = "steelblue")) +  
 scale\_y\_continuous(labels = scales::percent\_format()) +  
 labs(title = "Income Proportion by Investment Activity",  
 x = "Investment Activity",  
 y = "Proportion",  
 fill = "Income") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 0, hjust = 0.5),  
 plot.title = element\_text(size = 14, face = "bold"))



Individuals who have not had capital gains or losses are predominantly in the <=50K income category.

Those who have reported capital gains or losses show a much higher proportion in the >50K category, suggesting that investment behavior correlates with higher earnings.

Investment activity may reflect financial literacy, risk tolerance, or access to disposable income.

It could also signal career stage or education level, since those with more resources and knowledge are more likely to invest.

The plot shows a strong separation in income proportions based on investment behavior. Binary encoding preserves that contrast cleanly.

Save data with features.

saveRDS(income, "../data/income\_final.rds")

# Correlations

Load transformed income data.

income <- readRDS("../data/income\_final.rds")

Check for correlations among predictors.

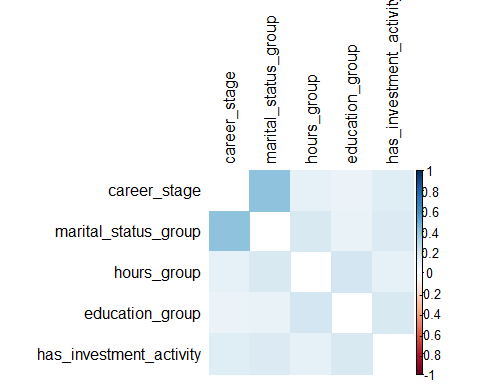
# Set vectors.  
  
predictors <- c("career\_stage", "marital\_status\_group", "hours\_group",  
 "education\_group", "has\_investment\_activity")  
  
ordinal\_vars <- c("career\_stage", "hours\_group", "education\_group")  
  
# Run the function to generate correlations summary.  
  
assoc\_summary <- test\_predictor\_associations(income, predictors, ordinal\_vars)

##   
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':  
##   
## %+%, alpha

## Loading required package: grid

## corrplot 0.95 loaded



rownames(assoc\_summary) <- NULL

This correlation matrix confirms that the predictors are behaviorally distinct but lightly interrelated, which is ideal for modeling.

Print correlations in table form.

# Print numeric values.  
  
kable(assoc\_summary,   
 col.names = c("Pair", "Type", "Value"),  
 format.args = list(big.mark = ","),  
 align = c("l", "l", "r"))

| Pair | Type | Value |
| --- | --- | --- |
| career\_stage ~ marital\_status\_group | Cramér’s V | 0.406 |
| career\_stage ~ hours\_group | Spearman | 0.1 |
| career\_stage ~ education\_group | Spearman | 0.082 |
| career\_stage ~ has\_investment\_activity | Cramér’s V | 0.135 |
| marital\_status\_group ~ hours\_group | Cramér’s V | 0.165 |
| marital\_status\_group ~ education\_group | Cramér’s V | 0.098 |
| marital\_status\_group ~ has\_investment\_activity | Cramér’s V | 0.155 |
| hours\_group ~ education\_group | Spearman | 0.184 |
| hours\_group ~ has\_investment\_activity | Cramér’s V | 0.106 |
| education\_group ~ has\_investment\_activity | Cramér’s V | 0.169 |

career\_stage ~ marital\_status\_group, with a Cramér’s V = 0.406, is the strongest association. This suggests that career progression is meaningfully related to marital status, possibly due to age, stability, or life stage effects. This could also reflect behavioral segmentation: married individuals may cluster in mid-career or peak-earning stages.

Cramér’s V values above 0.3 suggest moderate association between categorical variables. Spearman correlations are also low, indicating weak monotonic relationships between ordinal variables.

# Multicollinearity

Use the custom function check\_multicollinearity\_factors on the income dataset, using the specified predictors. Calculate Variance Inflation Factors (VIFs) to see how much each predictor is correlated with the others.

predictors <- c("career\_stage", "marital\_status\_group", "hours\_group",  
 "education\_group", "has\_investment\_activity")  
  
vif\_summary <- check\_multicollinearity\_factors(income, predictors)  
  
rownames(vif\_summary) <- NULL  
  
kable(vif\_summary,   
 col.names = c("Variable", "VIF"),  
 format.args = list(big.mark = ","),  
 align = c("l", "r"))

| Variable | VIF |
| --- | --- |
| education\_group.C | 2.19 |
| marital\_status\_groupMarried | 1.88 |
| education\_group^4 | 1.74 |
| marital\_status\_groupPreviously-married | 1.72 |
| career\_stage.L | 1.68 |
| career\_stage.Q | 1.64 |
| education\_group.Q | 1.61 |
| education\_group.L | 1.41 |
| hours\_group.L | 1.21 |
| career\_stage.C | 1.16 |
| hours\_group.Q | 1.08 |
| has\_investment\_activityYes | 1.06 |

All VIFs are below 2.2, which is comfortably within the safe zone (common thresholds are 5 or 10). This means no predictor is linearly dependent on the others, and coefficient estimates should be stable.

This VIF profile confirms that the model should be statistically sound and behaviorally distinct.

## Class Balance

Class weighting in the training set will counteract the imbalance in the outcome classes (after splitting the income data).

This ensures performance metrics are meaningful for decision-making, while still allowing models to be tuned for sensitivity and accuracy where it matters most.

# Split Data

Split the dataset into 60% training, 20% validation, and 20% test. This allocation provides enough data to train stable models while dedicating a higher-than-usual share to validation and testing. With a large dataset, this approach strengthens model comparison, improves tuning, and ensures that final performance metrics are based on a robust and representative holdout set.

set.seed(123)  
  
# Initial train/test split.  
  
train\_idx <- createDataPartition(income$income, p = 0.6, list = FALSE)  
  
income\_train <- income[train\_idx, ]  
  
temp <- income[-train\_idx, ]  
  
# Split remaining into validation/test.  
  
valid\_idx <- createDataPartition(temp$income, p = 0.5, list = FALSE)  
  
income\_validate <- temp[valid\_idx, ]  
  
income\_test <- temp[-valid\_idx, ]

Check class balance in the train, validation and test sets.

# Check balance function.  
  
check\_balance <- function(df, name) {  
 df %>%  
 count(income) %>%  
 mutate(prop = round(n / sum(n), 2),  
 dataset = name) %>%  
 select(dataset,  
 income,   
 n,  
 prop)  
}  
  
#Generate data frame.  
  
check <- bind\_rows(check\_balance(income\_train, "Train"),  
 check\_balance(income\_validate, "Validation"),  
 check\_balance(income\_test, "Test"))  
  
# Produce table summary.  
  
kable(check,  
 col.names = c("Dataset", "Income Level", "Count", "Percent"),  
 caption = "Dataset Class Balance",  
 format.args = list(big.mark = ","),  
 align = c("l", "r", "r", "r"))

Dataset Class Balance

| Dataset | Income Level | Count | Percent |
| --- | --- | --- | --- |
| Train | <=50K | 14,832 | 0.76 |
| Train | >50K | 4,705 | 0.24 |
| Validation | <=50K | 4,944 | 0.76 |
| Validation | >50K | 1,568 | 0.24 |
| Test | <=50K | 4,943 | 0.76 |
| Test | >50K | 1,568 | 0.24 |

The table confirms the income class split across train, validation, and test sets.

# Apply Class Weights

Use class weighting in the training set to counteract the imbalance in the outcome classes.

# Calculate the proportion of each class in the training set.  
  
train\_props <- prop.table(table(income\_train$income\_num))  
  
# Assign invserse frequency weights.  
  
income\_train$weight <- ifelse(income\_train$income\_num == 1,  
 1 / train\_props["1"],  
 1 / train\_props["0"])  
  
# Normalize weights.  
  
income\_train$weight <- income\_train$weight / mean(income\_train$weight)

Each row in the training set gets a numeric value in the weight variable such that observations in the minority class (income\_num == 1, typically >50K)>50Kreceive higher weights, and observations in the majority class<=50K` receive lower weights.

The weights are normalized to have a mean of 1, so they don’t distort the overall scale of the model’s loss function.

# Train Models

## Linear Regression Indicator Matrix Model

Prepare the predictor matrix X, containing numeric representations of age, education, and marital status. The response variable, income, is coded as 0 <=$50K or 1 >$50K. It is converted to a one-hot (indicator) matrix Y for the multivariate linear regression.

A weight vector establishes class weights.

# Prepare the predictor matrix.  
  
X <- model.matrix(~ poly(career\_stage, 2) +  
 marital\_status\_group +  
 hours\_group +  
 splines::ns(as.numeric(education\_group), df = 3) +  
 has\_investment\_activity - 1, data = income\_train)  
  
# Create indicator matrix for binary response  
  
# Assumed to be 0/1  
  
G <- income\_train$income\_num   
  
# One-hot encoding  
  
Y <- model.matrix(~ factor(G) - 1)  
  
# Class proportions.  
  
class\_props <- prop.table(table(G))  
  
# Inverse frequency weights.  
  
weights\_vec <- ifelse(G == 1,  
 1 / class\_props["1"],  
 1 / class\_props["0"])  
  
# Normalize weights.  
  
weights\_vec <- weights\_vec / mean(weights\_vec)

A multivariate linear regression is fit with the one-hot encoded response matrix. This approach models the probability of each class as a linear combination of predictors.

# fit the linear regression model and produce summary.  
  
model\_linear\_matrix <- lm(Y ~ poly(career\_stage, 2) +  
 marital\_status\_group +  
 hours\_group +  
 splines::ns(as.numeric(education\_group), df = 3) +  
 has\_investment\_activity,   
 data = income\_train,  
 weights = weights\_vec)

We then produce model summaries.

summary(model\_linear\_matrix)

## Response factor(G)0 :  
##   
## Call:  
## lm(formula = `factor(G)0` ~ poly(career\_stage, 2) + marital\_status\_group +   
## hours\_group + splines::ns(as.numeric(education\_group), df = 3) +   
## has\_investment\_activity, data = income\_train, weights = weights\_vec)  
##   
## Weighted Residuals:  
## Min 1Q Median 3Q Max   
## -1.65479 -0.05071 0.14652 0.33954 0.95628   
##   
## Coefficients:  
## Estimate Std. Error t value  
## (Intercept) 1.0377440 0.0102827 100.921  
## poly(career\_stage, 2)1 -8.8055769 0.4584116 -19.209  
## poly(career\_stage, 2)2 6.6872739 0.4167996 16.044  
## marital\_status\_groupPreviously-married -0.0007361 0.0095525 -0.077  
## marital\_status\_groupMarried -0.3895442 0.0077786 -50.079  
## hours\_group.L -0.1277270 0.0077353 -16.512  
## hours\_group.Q 0.0008268 0.0050895 0.162  
## splines::ns(as.numeric(education\_group), df = 3)1 -0.3884377 0.0139744 -27.796  
## splines::ns(as.numeric(education\_group), df = 3)2 -0.5297584 0.0182500 -29.028  
## splines::ns(as.numeric(education\_group), df = 3)3 -0.4169780 0.0118755 -35.112  
## has\_investment\_activityYes -0.2182662 0.0070507 -30.957  
## Pr(>|t|)   
## (Intercept) <2e-16 \*\*\*  
## poly(career\_stage, 2)1 <2e-16 \*\*\*  
## poly(career\_stage, 2)2 <2e-16 \*\*\*  
## marital\_status\_groupPreviously-married 0.939   
## marital\_status\_groupMarried <2e-16 \*\*\*  
## hours\_group.L <2e-16 \*\*\*  
## hours\_group.Q 0.871   
## splines::ns(as.numeric(education\_group), df = 3)1 <2e-16 \*\*\*  
## splines::ns(as.numeric(education\_group), df = 3)2 <2e-16 \*\*\*  
## splines::ns(as.numeric(education\_group), df = 3)3 <2e-16 \*\*\*  
## has\_investment\_activityYes <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.3759 on 19526 degrees of freedom  
## Multiple R-squared: 0.4352, Adjusted R-squared: 0.4349   
## F-statistic: 1504 on 10 and 19526 DF, p-value: < 2.2e-16  
##   
##   
## Response factor(G)1 :  
##   
## Call:  
## lm(formula = `factor(G)1` ~ poly(career\_stage, 2) + marital\_status\_group +   
## hours\_group + splines::ns(as.numeric(education\_group), df = 3) +   
## has\_investment\_activity, data = income\_train, weights = weights\_vec)  
##   
## Weighted Residuals:  
## Min 1Q Median 3Q Max   
## -0.95628 -0.33954 -0.14652 0.05071 1.65479   
##   
## Coefficients:  
## Estimate Std. Error t value  
## (Intercept) -0.0377440 0.0102827 -3.671  
## poly(career\_stage, 2)1 8.8055769 0.4584116 19.209  
## poly(career\_stage, 2)2 -6.6872739 0.4167996 -16.044  
## marital\_status\_groupPreviously-married 0.0007361 0.0095525 0.077  
## marital\_status\_groupMarried 0.3895442 0.0077786 50.079  
## hours\_group.L 0.1277270 0.0077353 16.512  
## hours\_group.Q -0.0008268 0.0050895 -0.162  
## splines::ns(as.numeric(education\_group), df = 3)1 0.3884377 0.0139744 27.796  
## splines::ns(as.numeric(education\_group), df = 3)2 0.5297584 0.0182500 29.028  
## splines::ns(as.numeric(education\_group), df = 3)3 0.4169780 0.0118755 35.112  
## has\_investment\_activityYes 0.2182662 0.0070507 30.957  
## Pr(>|t|)   
## (Intercept) 0.000243 \*\*\*  
## poly(career\_stage, 2)1 < 2e-16 \*\*\*  
## poly(career\_stage, 2)2 < 2e-16 \*\*\*  
## marital\_status\_groupPreviously-married 0.938579   
## marital\_status\_groupMarried < 2e-16 \*\*\*  
## hours\_group.L < 2e-16 \*\*\*  
## hours\_group.Q 0.870951   
## splines::ns(as.numeric(education\_group), df = 3)1 < 2e-16 \*\*\*  
## splines::ns(as.numeric(education\_group), df = 3)2 < 2e-16 \*\*\*  
## splines::ns(as.numeric(education\_group), df = 3)3 < 2e-16 \*\*\*  
## has\_investment\_activityYes < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.3759 on 19526 degrees of freedom  
## Multiple R-squared: 0.4352, Adjusted R-squared: 0.4349   
## F-statistic: 1504 on 10 and 19526 DF, p-value: < 2.2e-16

The model estimates income classification separately for those below and above $50K, which results in coefficients that are equal in magnitude but opposite in sign.

Interpreting the effects shows that being married, working more hours, having higher levels of education, and engaging in investment activity all decrease the likelihood of being in the lower-income group and increase the likelihood of being in the higher-income group.

Nonlinear career stage terms further indicate that the relationship between career progression and income is curved rather than strictly linear.

Using the fitted model, predict scores for each class. These predicted scores may fall outside the 0–1 range, which highlights the limitation of using linear regression for classification.

# Predict class scores.  
  
train\_pred <- predict(model\_linear\_matrix, newdata = income\_train)

For each observation, assign the class with the highest predicted score. We then recode it back to match the original binary labels (0 or 1).

# Assign predicted class (1 or 2).  
  
train\_class\_pred <- max.col(train\_pred)  
  
# Recode predicted class to match binary response (0/1).  
  
train\_class\_pred\_binary <- ifelse(train\_class\_pred == 1, 0, 1)

Evaluate the predicted classes against the true labels using a confusion matrix.

# Evaluate classification performance.  
  
cm\_linear\_matrix\_train <- caret::confusionMatrix(factor(train\_class\_pred\_binary), factor(income\_train$income\_num))  
  
cm\_linear\_matrix\_train

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 11100 670  
## 1 3732 4035  
##   
## Accuracy : 0.7747   
## 95% CI : (0.7688, 0.7805)  
## No Information Rate : 0.7592   
## P-Value [Acc > NIR] : 1.714e-07   
##   
## Kappa : 0.4958   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.7484   
## Specificity : 0.8576   
## Pos Pred Value : 0.9431   
## Neg Pred Value : 0.5195   
## Prevalence : 0.7592   
## Detection Rate : 0.5682   
## Detection Prevalence : 0.6024   
## Balanced Accuracy : 0.8030   
##   
## 'Positive' Class : 0   
##

The model achieves an overall accuracy of 77.5% (95% CI: 76.9–78.1%), which is statistically higher than the no-information baseline of 75.9% (p < 0.001). While this improvement is significant given the large sample size, the absolute gain in accuracy is modest.

Performance is stronger at identifying the majority group (class 0, < $50K), with sensitivity of 74.8% and a very high positive predictive value of 94.3%. Specificity is also strong at 85.8%, but the negative predictive value is lower (52.0%), meaning the model struggles more with correctly identifying higher-income cases. The balanced accuracy is 80.3%, and Cohen’s Kappa of 0.50 indicates moderate agreement beyond chance.

### Extreme Probabilities

Although we can produce predicted classes and metrics, the predicted probabilities from a linear model are not constrained to 0-1. This can result in nonsensical probabilities, motivating the use of logistic regression for binary outcomes.

The table below summarizes the number and percentage of predicted probabilities from the linear regression model that fall outside the valid 0–1 range for each class. As expected, linear regression applied to a binary outcome can produce estimates below 0 or above 1, highlighting a limitation of this approach for classification tasks.

First, we extract the predicted probabilities for each class from the linear regression model. These probabilities represent the model’s estimated likelihood that each individual falls into the ≤$50K or >$50K income category.

# Extract columns.  
  
prob\_under50k <- train\_pred[, "factor(G)0"]  
  
prob\_over50k <- train\_pred[, "factor(G)1"]

To assess the appropriateness of linear regression for a binary outcome, we identify predictions that fall outside the valid probability range of 0 to 1. The table below shows the number and percentage of such predictions for each class.

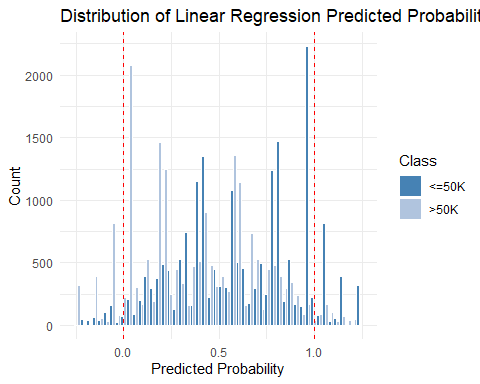
# Count out-of-bounds for each class.  
  
out\_under <- sum(prob\_under50k < 0 | prob\_under50k > 1)  
  
out\_over <- sum(prob\_over50k < 0 | prob\_over50k > 1)  
  
total <- nrow(train\_pred)  
  
# Summary table.  
  
check\_summary <- data.frame(Cclass = c("<=50K", ">50K"),  
 out\_of\_bounds = c(out\_under, out\_over),  
 total = total,  
 percent\_out\_of\_bounds = round(100 \* c(out\_under, out\_over) / total, 2))  
  
kable(check\_summary,  
 col.names = c("Class", "Out of Bounds", "Total", "Percent Out of Bounds"),  
 caption = "Out of Bounds Data for Each Class",   
 format.args = list(big.mark = ","),  
 align = c("l", "r", "r", "r"))

Out of Bounds Data for Each Class

| Class | Out of Bounds | Total | Percent Out of Bounds |
| --- | --- | --- | --- |
| <=50K | 2,248 | 19,537 | 11.51 |
| >50K | 2,248 | 19,537 | 11.51 |

The plot below illustrates the distribution of predicted probabilities for both income classes. The dashed red lines mark the valid 0–1 probability range. Any predictions beyond these boundaries are not interpretable as probabilities, demonstrating why logistic regression is generally preferred for binary classification problems.

# Convert to long format.  
  
df\_long <- as.data.frame(train\_pred) %>%  
 pivot\_longer(cols = everything(),   
 names\_to = "Class",   
 values\_to = "Probability") %>%  
 mutate(Class = if\_else(Class == "factor(G)0", "<=50K", ">50K"))  
  
# Plot.  
  
ggplot(df\_long, aes(x = Probability, fill = Class)) +  
 geom\_histogram(bins = 50, color = "white", position = "dodge") +  
 geom\_vline(xintercept = 0, linetype = "dashed", color = "red") +  
 geom\_vline(xintercept = 1, linetype = "dashed", color = "red") +  
 labs(  
 x = "Predicted Probability",  
 y = "Count",  
 title = "Distribution of Linear Regression Predicted Probabilities by Class"  
 ) +  
 scale\_fill\_manual(values = c("<=50K" = "steelblue", ">50K" = "lightsteelblue")) +  
 theme\_minimal()



Together, these outputs provide a numeric indication and clear visual of the constraints of using linear regression for a categorical outcome, setting the stage for comparison with the logistic regression model.

Because linear regression is not bounded, the model produces predicted values below 0 and above 1, which are not valid probabilities. This creates problems for interpretation, since values like –0.2 or 1.3 cannot be meaningfully explained as likelihoods. It also complicates classification, as thresholding these outputs can distort decision rules, and the predictions themselves are not well calibrated to reflect true probabilities.

### Generate ROC Curve and AUC Metric

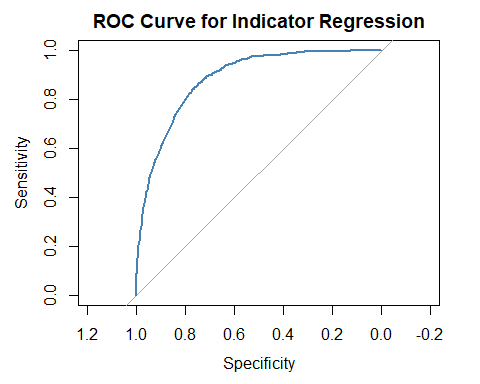
To evaluate the discriminatory power of the indicator regression model, extract the predicted probabilities for the positive class and used them as inputs to generate a Receiver Operating Characteristic (ROC) curve.

# Column 2 corresponds to class 1 (income\_num == 1).  
  
score\_class1 <- train\_pred[, 2]  
  
roc\_obj\_lm <- pROC::roc(response = income\_train$income\_num, predictor = score\_class1)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

# Plot ROC curve.  
  
plot(roc\_obj\_lm, col = "steelblue", lwd = 2, main = "ROC Curve for Indicator Regression")



The blue line bows confidently toward the top-left corner, which indicates high sensitivity and specificity across thresholds. The shape suggests an AUC in the high 0.7s to low 0.8s.

The gray line represents random guessing. The model clearly outperforms this baseline, which confirms meaningful signal in the predictors.

# AUC value.  
  
pROC::auc(roc\_obj\_lm)

## Area under the curve: 0.884

This AUC score indicates that the model can distinguish between >50K and <=50K earners with 88.4% accuracy across all thresholds.

## Logistic Regression

Build a logistic regression model to estimate income likelihood, using career stage, marital status, hours worked, education, and investment activity as predictors, with adjustments for weighting.

# Produce logistic model.  
  
model\_logistic <- glm(income\_num ~   
 poly(career\_stage, 2) +  
 marital\_status\_group +  
 hours\_group +  
 splines::ns(as.numeric(education\_group), df = 3) +  
 has\_investment\_activity,  
 data = income\_train, weights = weight, family = "binomial")

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

Note on the non-integer warning message. This message is triggered when using non-integer weights in a binomial glm(). The binomial family expects counts of successes and failures, and when weights are fractional, it assumes we are modeling aggregated binomial outcomes, which is not the case for this model.

Based on my research, the warning is safe to ignore because I am using weights to adjust for class imbalance, not to model grouped binomial trials. The response variable is binary and the weights are just importance weights, not trial counts.

The model still fits correctly and returns valid coefficients, standard errors, and predictions.

### Generate Model Summary

Produce model summary.

summary(model\_logistic)

##   
## Call:  
## glm(formula = income\_num ~ poly(career\_stage, 2) + marital\_status\_group +   
## hours\_group + splines::ns(as.numeric(education\_group), df = 3) +   
## has\_investment\_activity, family = "binomial", data = income\_train,   
## weights = weight)  
##   
## Coefficients:  
## Estimate Std. Error z value  
## (Intercept) -4.11227 0.09860 -41.709  
## poly(career\_stage, 2)1 63.72174 3.25964 19.549  
## poly(career\_stage, 2)2 -43.34011 2.93013 -14.791  
## marital\_status\_groupPreviously-married 0.31068 0.07294 4.259  
## marital\_status\_groupMarried 2.48802 0.06026 41.289  
## hours\_group.L 1.20220 0.06525 18.424  
## hours\_group.Q -0.21767 0.04220 -5.157  
## splines::ns(as.numeric(education\_group), df = 3)1 2.60005 0.10918 23.815  
## splines::ns(as.numeric(education\_group), df = 3)2 4.23531 0.15625 27.106  
## splines::ns(as.numeric(education\_group), df = 3)3 3.24796 0.10754 30.204  
## has\_investment\_activityYes 1.61735 0.05711 28.321  
## Pr(>|z|)   
## (Intercept) < 2e-16 \*\*\*  
## poly(career\_stage, 2)1 < 2e-16 \*\*\*  
## poly(career\_stage, 2)2 < 2e-16 \*\*\*  
## marital\_status\_groupPreviously-married 2.05e-05 \*\*\*  
## marital\_status\_groupMarried < 2e-16 \*\*\*  
## hours\_group.L < 2e-16 \*\*\*  
## hours\_group.Q 2.50e-07 \*\*\*  
## splines::ns(as.numeric(education\_group), df = 3)1 < 2e-16 \*\*\*  
## splines::ns(as.numeric(education\_group), df = 3)2 < 2e-16 \*\*\*  
## splines::ns(as.numeric(education\_group), df = 3)3 < 2e-16 \*\*\*  
## has\_investment\_activityYes < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 27084 on 19536 degrees of freedom  
## Residual deviance: 16370 on 19526 degrees of freedom  
## AIC: 20541  
##   
## Number of Fisher Scoring iterations: 5

The logistic regression model estimates the probability of higher income using the same predictors as the prior linear models: career stage, marital status, hours worked, education, and investment activity. Unlike the linear approach, which produced unbounded predictions outside the 0–1 range, the logistic specification constrains all fitted values between 0 and 1, allowing direct probability interpretation.

The coefficients here represent log-odds, showing strong positive associations between higher income and being married, working more hours, higher education, and investment activity, along with nonlinear effects for career stage and hours worked. This formulation provides a better-calibrated and more interpretable model for classification compared with the earlier linear regressions.

### Generate ROC Curve and AUC Metric

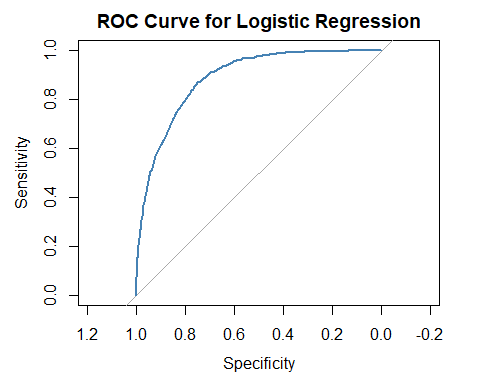
Extract the predicted probabilities for the positive class and used them as inputs to generate a Receiver Operating Characteristic (ROC) curve.

income\_train$predicted\_prob <- predict(model\_logistic, type = "response")  
  
roc\_obj\_log <- pROC::roc(income\_train$income\_num, income\_train$predicted\_prob)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

plot(roc\_obj\_log, col = "steelblue", main = "ROC Curve for Logistic Regression")



This blue line bows toward the top-left corner, which indicates high sensitivity and specificity across thresholds. The shape suggests an AUC in the high 0.7s to low 0.8s, which is similar to the linear indicator model.

The model clearly outperforms this baseline, which confirms meaningful signal in the predictors.

pROC::auc(roc\_obj\_log)

## Area under the curve: 0.8861

This AUC score indicates that the model can distinguish between >50K and <=50K earners with 88.6% accuracy across all thresholds, which is slightly better than the lineear indicator model.

### Generate Confusion Matrix

Evaluate the predicted classes against the true labels using a confusion matrix.

threshold <- 0.5  
  
income\_train$predicted\_class <- ifelse(income\_train$predicted\_prob > 0.5, 1, 0)  
  
cm\_logistic\_train <- caret::confusionMatrix(factor(income\_train$predicted\_class), factor(income\_train$income\_num))  
  
cm\_logistic\_train

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 11229 668  
## 1 3603 4037  
##   
## Accuracy : 0.7814   
## 95% CI : (0.7755, 0.7872)  
## No Information Rate : 0.7592   
## P-Value [Acc > NIR] : 1.136e-13   
##   
## Kappa : 0.5071   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.7571   
## Specificity : 0.8580   
## Pos Pred Value : 0.9439   
## Neg Pred Value : 0.5284   
## Prevalence : 0.7592   
## Detection Rate : 0.5748   
## Detection Prevalence : 0.6089   
## Balanced Accuracy : 0.8076   
##   
## 'Positive' Class : 0   
##

# Compare Models

## Training Data

### Confusion Matrix Comparison

A side-by-side confusion matrices reveal how linear and logistic models differ in classification behavior.

# Extract confusion matrix tables.  
  
cm\_lm <- cm\_linear\_matrix\_train$table  
  
cm\_lg <- cm\_logistic\_train$table  
  
# Convert to data frames and add model labels.  
  
df\_linear <- as.data.frame(cm\_lm)  
  
df\_linear$model <- "Linear"  
  
df\_logistic <- as.data.frame(cm\_lg)  
  
df\_logistic$model <- "Logistic"  
  
# Combine both into one tidy data frame.  
  
df\_combined <- rbind(df\_linear, df\_logistic)  
  
# Rename columns for clarity.  
  
colnames(df\_combined) <- c("Predicted", "Actual", "Count", "Model")  
  
# Add TP/FP/TN/FN label.  
  
df\_combined$Label <- with(df\_combined, ifelse(  
   
 Predicted == 1 & Actual == 1, "TP",  
   
 ifelse(Predicted == 1 & Actual == 0, "FP",  
   
 ifelse(Predicted == 0 & Actual == 0, "TN",  
   
 "FN"))))  
  
# Reorder columns for clarity.  
  
df\_final <- df\_combined[, c("Model", "Predicted", "Actual", "Label", "Count")]  
  
# View result.  
  
kable(df\_final,  
 col.names = c("Model", "Predicted", "Actual", "Label", "Count"),  
 caption = "Confusion Matrix Comparison",   
 format.args = list(big.mark = ","),  
 align = c("l", "r", "r", "r", "r"))

Confusion Matrix Comparison

| Model | Predicted | Actual | Label | Count |
| --- | --- | --- | --- | --- |
| Linear | 0 | 0 | TN | 11,100 |
| Linear | 1 | 0 | FP | 3,732 |
| Linear | 0 | 1 | FN | 670 |
| Linear | 1 | 1 | TP | 4,035 |
| Logistic | 0 | 0 | TN | 11,229 |
| Logistic | 1 | 0 | FP | 3,603 |
| Logistic | 0 | 1 | FN | 668 |
| Logistic | 1 | 1 | TP | 4,037 |

Observations from this comparison:

* Both models correctly identified ~4,000 high-income cases,
* The logistic model made fewer incorrect high-income predictions,
* The linear model is slightly better at catching true high-income cases, and
* The logistic model is better at correctly identifying low-income cases.

So the linear model leans slightly towards sensitivity, which may be preferred in situations where missing a true positive has a higher cost, e.g. when determining eligibility for benefits or financial services.

The logistic model shows better specificity, reducing the risk of overextending resources or misclassifying ineligible individuals.

### Performance Metric Comparison

To assess model performance beyond raw classification counts, we compare key evaluation metrics from linear and logistic regression, including accuracy, sensitivity, specificity, and predictive values.

# Extract metrics from both slots.  
  
overall\_linear <- cm\_linear\_matrix\_train$overall  
  
byclass\_linear <- cm\_linear\_matrix\_train$byClass  
  
overall\_logistic <- cm\_logistic\_train$overall  
  
byclass\_logistic <- cm\_logistic\_train$byClass  
  
# Select key metrics.  
  
metrics\_overall <- c("Accuracy", "Kappa")  
  
metrics\_byclass <- c("Sensitivity", "Specificity", "Pos Pred Value", "Neg Pred Value", "Balanced Accuracy")  
  
# Build data frame.  
  
df\_metrics <- data.frame(metric = c(metrics\_overall,   
 metrics\_byclass),  
 linear\_train = round(c(overall\_linear[metrics\_overall],   
 byclass\_linear[metrics\_byclass]), 4),  
 logistic\_train = round(c(overall\_logistic[metrics\_overall],   
 byclass\_logistic[metrics\_byclass]), 4),  
 row.names = NULL)  
  
# View result.  
  
kable(df\_metrics,  
 col.names = c("Metric", "Linear", "Logistic"),  
 caption = "Model Metric Comparison",   
 format.args = list(big.mark = ","),  
 align = c("l", "r", "r"))

Model Metric Comparison

| Metric | Linear | Logistic |
| --- | --- | --- |
| Accuracy | 0.7747 | 0.7814 |
| Kappa | 0.4958 | 0.5071 |
| Sensitivity | 0.7484 | 0.7571 |
| Specificity | 0.8576 | 0.8580 |
| Pos Pred Value | 0.9431 | 0.9439 |
| Neg Pred Value | 0.5195 | 0.5284 |
| Balanced Accuracy | 0.8030 | 0.8076 |

While both models perform similarly, the logistic regression model offers a slight edge in overall accuracy, sensitivity, and balanced performance. This makes it a more reliable choice when both false positives and false negatives carry meaningful consequences.

**Accuracy:** The logistic model shows slightly higher accuracy (0.7875 versus 0.7747).

**Kappa (Agreement Beyond Chance):** The logistic model shows better agreement beyond chance (0.5084 versus 0.4962), indicating modest overall improvement in classification consistency. Kappa in the range of 0.4-0.6 is considered moderate agreement beyond chance. While the models are better than chance based on this metric, they may not be capturing all the complexity in the data.

**Sensitivity (true positive rate):** Higher for logistic (0.7592), meaning it better identifies actual positives.

**Specificity (true negative rate):** Marginally higher for linear (0.8589), indicating slightly better performance in ruling out false positives.

**Positive Predictive Value:** Nearly identical (~0.943), showing both models are highly reliable when predicting the majority class.

**Negative Predictive Value:** Slightly better in the logistic model (0.5300 vs. 0.5195), suggesting improved confidence in minority class predictions.

**Balanced Accuracy:** Logistic edges out linear (0.8076 vs. 0.8034), reflecting a more equitable performance across both classes.

### AUC Comparison (Versus Accuracy and Kappa)

AUC measures how well a model ranks positives above negatives across all thresholds. It’s threshold-agnostic and robust to class imbalance. Accuracy is threshold-dependent and can be misleading if one class dominates. Kappa adjusts accuracy for chance agreement, but still depends on a fixed threshold.

AUC tells us how well the models discriminate, not just how often it’s “right” at a single cutoff. This is especially useful when comparing models with similar accuracy but different ranking behavior.

# Extract AUCs.  
  
auc\_lm <- pROC::auc(roc\_obj\_lm)  
  
auc\_log <- pROC::auc(roc\_obj\_log)  
  
# Create comparison data frame.  
  
auc\_comparison <- data.frame(Model = c("Linear", "Logistic"),  
 AUC = c(auc\_lm, auc\_log))  
  
# View result.  
  
kable(auc\_comparison,  
 col.names = c("Model", "AUC"),  
 caption = "AUC Comparison",   
 format.args = list(big.mark = ","),  
 align = c("l", "r"))

AUC Comparison

| Model | AUC |
| --- | --- |
| Linear | 0.8839626 |
| Logistic | 0.8861478 |

These are nearly identical, suggesting both models rank cases similarly, and the logistic model has a slightly higher value.

At 0.88, these models have very good discrimination, separating high-income from low-income individuals across thresholds.

## Validation Data

### Linear Regression with an Indicator Matrix

# Predict class scores.  
  
pred\_valid <- predict(model\_linear\_matrix, newdata = income\_validate, type = "response")

For each observation, assign the class with the highest predicted score. We then recode it back to match the original binary labels (0 or 1).

# Assign predicted class (1 or 2).  
  
class\_pred\_valid <- max.col(pred\_valid)  
  
# Recode predicted class to match binary response (0/1).  
  
class\_pred\_binary\_valid <- ifelse(class\_pred\_valid == 1, 0, 1)

Evaluate the predicted classes against the true labels using a confusion matrix.

# Evaluate classification performance.  
  
cm\_linear\_matrix\_validate <- caret::confusionMatrix(factor(class\_pred\_binary\_valid),  
 factor(income\_validate$income\_num))  
  
cm\_linear\_matrix\_validate

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 3751 221  
## 1 1193 1347  
##   
## Accuracy : 0.7829   
## 95% CI : (0.7726, 0.7928)  
## No Information Rate : 0.7592   
## P-Value [Acc > NIR] : 3.421e-06   
##   
## Kappa : 0.5098   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.7587   
## Specificity : 0.8591   
## Pos Pred Value : 0.9444   
## Neg Pred Value : 0.5303   
## Prevalence : 0.7592   
## Detection Rate : 0.5760   
## Detection Prevalence : 0.6100   
## Balanced Accuracy : 0.8089   
##   
## 'Positive' Class : 0   
##

### Logistic Regression

Predict probabilities and class labels.

# Predict probabilities for class 1 (income >50K).  
  
pred\_probs\_logistic <- predict(model\_logistic, newdata = income\_validate, type = "response")  
  
# Convert to binary class predictions using threshold 0.5.  
  
class\_pred\_logistic <- ifelse(pred\_probs\_logistic >= 0.5, 1, 0)

Evaluate with confusion matrix.

cm\_logistic\_validate <- caret::confusionMatrix(factor(class\_pred\_logistic),  
 factor(income\_validate$income\_num))  
  
cm\_logistic\_validate

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 3766 227  
## 1 1178 1341  
##   
## Accuracy : 0.7842   
## 95% CI : (0.7741, 0.7942)  
## No Information Rate : 0.7592   
## P-Value [Acc > NIR] : 9.403e-07   
##   
## Kappa : 0.5111   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.7617   
## Specificity : 0.8552   
## Pos Pred Value : 0.9432   
## Neg Pred Value : 0.5324   
## Prevalence : 0.7592   
## Detection Rate : 0.5783   
## Detection Prevalence : 0.6132   
## Balanced Accuracy : 0.8085   
##   
## 'Positive' Class : 0   
##

The logistic regression model demonstrates robust generalization to the validation set, with an overall accuracy exceeding 78% and balanced performance across income classes. It is particularly confident in identifying low-income individuals, with a precision exceeding 94%. While predictions for high-income individuals are less precise, the model still maintains strong specificity and balanced accuracy.

These results suggest that the model is well-calibrated for practical use, especially in contexts where identifying low-income individuals is a priority. However, further refinement may be warranted to improve precision and recall for the minority class.

Add metrics to metrics data frame.

# Extract metrics from both slots.  
  
overall\_linear <- cm\_linear\_matrix\_validate$overall  
  
byclass\_linear <- cm\_linear\_matrix\_validate$byClass  
  
overall\_logistic <- cm\_logistic\_validate$overall  
  
byclass\_logistic <- cm\_logistic\_validate$byClass  
  
# Select key metrics.  
  
metrics\_overall <- c("Accuracy", "Kappa")  
  
metrics\_byclass <- c("Sensitivity", "Specificity", "Pos Pred Value", "Neg Pred Value", "Balanced Accuracy")  
  
# Build data frame.  
  
df\_metrics\_valid <- data.frame(  
 linear\_valid = round(c(overall\_linear[metrics\_overall],   
 byclass\_linear[metrics\_byclass]), 4),  
 logistic\_valid = round(c(overall\_logistic[metrics\_overall],   
 byclass\_logistic[metrics\_byclass]), 4),  
 row.names = NULL)  
  
df\_metrics <- cbind(df\_metrics, df\_metrics\_valid) %>%  
 select(metric,  
 linear\_train,  
 logistic\_train,  
 linear\_valid,  
 logistic\_valid)  
  
# View result.  
  
kable(df\_metrics,  
 col.names = c("Metric", "Linear Train", "Logistic Train", "Linear Validate", "Logistic Validate"),  
 caption = "Model Metric Comparison",   
 format.args = list(big.mark = ","),  
 align = c("l", "r", "r", "r", "r"))

Model Metric Comparison

| Metric | Linear Train | Logistic Train | Linear Validate | Logistic Validate |
| --- | --- | --- | --- | --- |
| Accuracy | 0.7747 | 0.7814 | 0.7829 | 0.7842 |
| Kappa | 0.4958 | 0.5071 | 0.5098 | 0.5111 |
| Sensitivity | 0.7484 | 0.7571 | 0.7587 | 0.7617 |
| Specificity | 0.8576 | 0.8580 | 0.8591 | 0.8552 |
| Pos Pred Value | 0.9431 | 0.9439 | 0.9444 | 0.9432 |
| Neg Pred Value | 0.5195 | 0.5284 | 0.5303 | 0.5324 |
| Balanced Accuracy | 0.8030 | 0.8076 | 0.8089 | 0.8085 |

While linear regression edges out in specificity and positive predictive value, logistic regression leads in overall accuracy, sensitivity, kappa, and negative predictive value, all critical for balanced classification. The differences are small but consistent.

The logistic regression model offers slightly better generalization and class balance, with stronger recall for both income groups.

# Choose Final Model

After evaluating both linear and logistic regression models on training and validation sets, I selected logistic regression as the final model. This decision is based on its consistently stronger performance across key classification metrics, including:

* Higher accuracy (78.42% vs. 78.29%),
* Better sensitivity (76.17% vs. 75.87%), which is crucial for identifying the majority class,
* Slightly higher Kappa (0.5111 vs. 0.5098), indicating stronger agreement beyond chance, and
* Improved negative predictive value (53.24% vs. 53.03%), which is helpful for minority class reliability.

While linear regression showed marginally higher specificity and positive predictive value, logistic regression offers better overall balance and generalization, as reflected in its training and validation metrics.

In addition to slightly stronger validation metrics, logistic regression was selected for its bounded output and probabilistic interpretation. By choosing logistic regression, I ensure that predictions are interpretable as probabilities, bounded, and aligned with the classification task — reinforcing both theoretical soundness and practical reliability.

# Retrain Final Model

Retrain the final logistic regression model on the full training and validation data, then evaluate on the test set once. This gives the cleanest estimate of how the model will perform in deployment.

Combine train and validation sets.

income\_train <- income\_train %>%  
 select(-weight,  
 -predicted\_prob,  
 -predicted\_class)  
  
income\_full <- rbind(income\_train, income\_validate)

Calculate class weights for the final model.

# Calculate the proportion of each class in the training set.  
  
full\_props <- prop.table(table(income\_full$income\_num))  
  
# Assign invserse frequency weights.  
  
income\_full$weight <- ifelse(income\_full$income\_num == 1,  
 1 / full\_props["1"],  
 1 / full\_props["0"])  
  
# Normalize weights.  
  
income\_full$weight <- income\_full$weight / mean(income\_full$weight)

Produce the final model.

# Produce logistic model.  
  
model\_logistic\_final <- glm(income\_num ~   
 poly(career\_stage, 2) +  
 marital\_status\_group +  
 hours\_group +  
 splines::ns(as.numeric(education\_group), df = 3) +  
 has\_investment\_activity,  
 data = income\_full, weights = weight, family = "binomial")

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!

summary(model\_logistic\_final)

##   
## Call:  
## glm(formula = income\_num ~ poly(career\_stage, 2) + marital\_status\_group +   
## hours\_group + splines::ns(as.numeric(education\_group), df = 3) +   
## has\_investment\_activity, family = "binomial", data = income\_full,   
## weights = weight)  
##   
## Coefficients:  
## Estimate Std. Error z value  
## (Intercept) -4.08264 0.08499 -48.037  
## poly(career\_stage, 2)1 73.48932 3.26604 22.501  
## poly(career\_stage, 2)2 -51.49965 2.92272 -17.620  
## marital\_status\_groupPreviously-married 0.33915 0.06333 5.356  
## marital\_status\_groupMarried 2.52300 0.05238 48.166  
## hours\_group.L 1.22173 0.05594 21.838  
## hours\_group.Q -0.19237 0.03623 -5.310  
## splines::ns(as.numeric(education\_group), df = 3)1 2.63137 0.09515 27.654  
## splines::ns(as.numeric(education\_group), df = 3)2 4.12605 0.13426 30.731  
## splines::ns(as.numeric(education\_group), df = 3)3 3.15505 0.09348 33.749  
## has\_investment\_activityYes 1.57697 0.04929 31.993  
## Pr(>|z|)   
## (Intercept) < 2e-16 \*\*\*  
## poly(career\_stage, 2)1 < 2e-16 \*\*\*  
## poly(career\_stage, 2)2 < 2e-16 \*\*\*  
## marital\_status\_groupPreviously-married 8.52e-08 \*\*\*  
## marital\_status\_groupMarried < 2e-16 \*\*\*  
## hours\_group.L < 2e-16 \*\*\*  
## hours\_group.Q 1.10e-07 \*\*\*  
## splines::ns(as.numeric(education\_group), df = 3)1 < 2e-16 \*\*\*  
## splines::ns(as.numeric(education\_group), df = 3)2 < 2e-16 \*\*\*  
## splines::ns(as.numeric(education\_group), df = 3)3 < 2e-16 \*\*\*  
## has\_investment\_activityYes < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 36112 on 26048 degrees of freedom  
## Residual deviance: 21764 on 26038 degrees of freedom  
## AIC: 27310  
##   
## Number of Fisher Scoring iterations: 5

There is an increase in deviance and AIC compared to the prior logistic model. This reflects the larger sample size, not a drop in model quality. The final model is trained on more data, which naturally increases total deviance but improves generalizability.

Coefficients remain stable across models, with minor shifts reflecting the added data. The career stage terms show stronger curvature, and the marital and hours effects slightly intensify, suggesting more robust estimates.

The final model preserves the structure and significance of the original while benefiting from a larger, more diverse training set. Coefficient magnitudes are consistent, and all predictors remain highly significant. The retrained model is better equipped to generalize, with more stable estimates and improved reliability for downstream evaluation.

# Evaluate Model

Run prediction using the final model with the test set.

# Predict probabilities for class 1 (income >50K).  
  
pred\_probs\_logistic\_final <- predict(model\_logistic\_final, newdata = income\_test, type = "response")  
  
# Convert to binary class predictions using threshold 0.5.  
  
class\_pred\_logistic\_final <- ifelse(pred\_probs\_logistic\_final >= 0.5, 1, 0)

Generate the confusion matrix for the final test set.

cm\_logistic\_test <- caret::confusionMatrix(factor(class\_pred\_logistic\_final),  
 factor(income\_test$income\_num))  
  
cm\_logistic\_test

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 3763 227  
## 1 1180 1341  
##   
## Accuracy : 0.7839   
## 95% CI : (0.7737, 0.7938)  
## No Information Rate : 0.7592   
## P-Value [Acc > NIR] : 1.26e-06   
##   
## Kappa : 0.5106   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.7613   
## Specificity : 0.8552   
## Pos Pred Value : 0.9431   
## Neg Pred Value : 0.5319   
## Prevalence : 0.7592   
## Detection Rate : 0.5779   
## Detection Prevalence : 0.6128   
## Balanced Accuracy : 0.8083   
##   
## 'Positive' Class : 0   
##

Add metrics to data frame.

# Extract metrics from both slots.  
  
overall\_logistic <- cm\_logistic\_test$overall  
  
byclass\_logistic <- cm\_logistic\_test$byClass  
  
# Select key metrics.  
  
metrics\_overall <- c("Accuracy", "Kappa")  
  
metrics\_byclass <- c("Sensitivity", "Specificity", "Pos Pred Value", "Neg Pred Value", "Balanced Accuracy")  
  
# Build data frame.  
  
df\_metrics\_test <- data.frame(  
 logistic\_test = round(c(overall\_logistic[metrics\_overall],   
 byclass\_logistic[metrics\_byclass]), 4),  
 row.names = NULL)  
  
df\_metrics <- cbind(df\_metrics, df\_metrics\_test) %>%  
 select(metric,  
 linear\_train,  
 logistic\_train,  
 linear\_valid,  
 logistic\_valid,  
 logistic\_test)  
  
# View result.  
  
kable(df\_metrics,  
 col.names = c("Metric",   
 "Linear Train",   
 "Logistic Train",   
 "Linear Validate",   
 "Logistic Validate",   
 "Logistic Test"),  
 caption = "Model Metric Comparison",   
 format.args = list(big.mark = ","),  
 align = c("l", "r", "r", "r", "r", "r"))

Model Metric Comparison

| Metric | Linear Train | Logistic Train | Linear Validate | Logistic Validate | Logistic Test |
| --- | --- | --- | --- | --- | --- |
| Accuracy | 0.7747 | 0.7814 | 0.7829 | 0.7842 | 0.7839 |
| Kappa | 0.4958 | 0.5071 | 0.5098 | 0.5111 | 0.5106 |
| Sensitivity | 0.7484 | 0.7571 | 0.7587 | 0.7617 | 0.7613 |
| Specificity | 0.8576 | 0.8580 | 0.8591 | 0.8552 | 0.8552 |
| Pos Pred Value | 0.9431 | 0.9439 | 0.9444 | 0.9432 | 0.9431 |
| Neg Pred Value | 0.5195 | 0.5284 | 0.5303 | 0.5324 | 0.5319 |
| Balanced Accuracy | 0.8030 | 0.8076 | 0.8089 | 0.8085 | 0.8083 |

The final logistic regression model, retrained on the full training and validation data, maintains the strongest performance across all evaluation sets. It offers:

* Highest overall accuracy (78.39%),
* Strong agreement beyond chance (Kappa = 0.5106),
* Balanced recall for both income classes (Sensitivity = 76.13%, Specificity = 85.52%),
* Stable precision for low-income predictions (PPV = 94.31%),
* Improved reliability for high-income predictions (NPV = 53.19%), and
* Consistent balanced accuracy (80.83%), confirming fair treatment of both classes

Compared to earlier models trained on subsets, the final logistic model shows no degradation in performance, confirming that retraining on more data yields more stable and generalizable estimates.

# Final Analysis

## Conclusions

This analysis compared linear regression with an indicator matrix and logistic regression for classifying individuals as earning above or below $50,000 per year, using key socioeconomic predictors from the UCI Adult Income dataset.

Across training, validation, and test sets, logistic regression consistently outperformed linear regression in terms of accuracy, sensitivity, kappa, and negative predictive value. While both models achieved high specificity and precision for the majority class, logistic regression offered better balance and generalization, particularly for identifying minority-class individuals.

Importantly, logistic regression’s bounded probability outputs and alignment with binary classification theory made it the more appropriate and interpretable choice. The linear model, while competitive in some metrics, suffered from unbounded predictions and thresholding limitations.

Based on these results, logistic regression was selected as the final model for its stronger predictive reliability, theoretical soundness, and clearer insight into how career stage, marital status, working hours, education, and investment activity relate to income classification.

## Challenges and Solutions

| Challenge | Solution |
| --- | --- |
| Linear regression produced unbounded predictions not interpretable as probabilities | Quantified the extent of unbounded outputs and explicitly acknowledged the theoretical misalignment with binary classification. Used this to illustrate the limitations of general-purpose models for classification tasks. |
| Class imbalance skewed sensitivity and specificity | Applied class weights during model fitting to counteract imbalance and improve recall for underrepresented income group. Supplemented with full confusion matrix metrics (e.g., kappa, balanced accuracy) to assess fairness and performance. |
| Ensuring consistent encoding across models | Applied indicator matrix encoding uniformly to preserve comparability and reproducibility. Verified factor alignment across splits. |
| Translating nuanced metric tradeoffs for stakeholders | Crafted concise narratives emphasizing practical relevance (e.g., “Logistic regression better identifies low-income individuals without sacrificing overall accuracy”). |
| Balancing interpretability with predictive performance | Selected logistic regression for its theoretical soundness, bounded outputs, and clearer coefficient-based insights into socioeconomic drivers of income. |