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 09, 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 explanatory variables.

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

My goal is to answer the 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 classification, with a binary target (<=50K vs. >50K) and a rich mix of predictors:

**Career Stage & Education:** Age, occupation, work experience, and education (years and attainment) proxy human capital and labor market position.

**Financial Behavior:** Capital gains/losses signal investment activity beyond earned income.

**Socioeconomic Context:** Marital status and hours worked reflect broader income dynamics.

**Mixed Data Types:** Categorical and continuous features enable evaluation of model handling across variable types.

**Relevance:** These variables capture key drivers of upward mobility—education, career, and financial engagement.

## Candidate Models

### Linear Regression with an Indicator Matrix

Used here as a benchmark, linear regression estimates class probabilities from a binary indicator response. While it reveals linear relationships (e.g., age, hours worked), it can produce values outside the 0–1 range and lacks a probabilistic foundation.

### Logistic Regression

A standard for binary classification, logistic regression models log-odds as a linear function of predictors. It constrains outputs to 0-1 and yields interpretable odds ratios, making it well-suited for this task.

### Rationale

Comparing these models on accuracy, sensitivity, specificity, and AUC highlights trade-offs between a general-purpose regression and a classification-specific approach.

### Baseline Expectations

Before modeling, it’s useful to anticipate how each approach may perform:

* **Linear Regression:** As a benchmark, it may yield reasonable accuracy but often produces out-of-bound probabilities and weak sensitivity/specificity due to its misalignment with classification thresholds.
* **Logistic Regression:** Purpose-built for binary outcomes, it offers better calibration, constrained predictions, and interpretable odds ratios—making it more reliable for this task.
* **Comparative Outlook:** Logistic regression is expected to outperform on AUC and balanced metrics, while linear regression may expose the limitations of general-purpose models in classification contexts.

## Modeling Outcome Preview

This section summarizes the final model selection, key performance metrics, and major challenges addressed throughout the pipeline. See **Choose Final Model** and **Final Analysis** sections for details.

### Final Model Choice

Logistic regression was selected as the final model after benchmarking against a linear indicator approach. It consistently delivered stronger predictive balance, clearer interpretability, and bounded probability outputs. On the test set, it achieved a balanced accuracy of 0.73 and demonstrated reliable performance across income subgroups.

### Key Performance Metrics

* **Balanced Accuracy:** 0.73 on unseen test set
* **Sensitivity / Specificity:** High sensitivity with improved specificity for minority class
* **Residual Diagnostics:** Logistic regression showed no systematic bias or variance inflation. Linear regression, by contrast, exhibited heteroscedasticity and residual asymmetry, limiting reliability.

### Limitations of Linear Regression as a Classifier, Challenges and Solutions

Initial modeling with linear regression revealed critical limitations for binary classification: unbounded predictions, poor calibration, and violations of core assumptions. These issues undermined interpretability and reliability, especially in threshold-sensitive contexts.

To address these challenges, I transitioned to logistic regression, which offered bounded probability outputs, clearer coefficient interpretation, and stronger theoretical alignment. Throughout the modeling process, I quantified misfit, crafted stakeholder-ready narratives to explain metric tradeoffs, and prioritized interpretability without sacrificing predictive strength.

## 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.

Two customized R functions enable 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")

library("dplyr")

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

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

# Summary of Data Exploration, Cleaning and Transformation (See Appendix A)

The modeling pipeline began with a comprehensive audit of the UCI Adult Income dataset to ensure readiness for classification. Key steps included:

**Exploratory Analysis:** Assessed distributions and relationships between predictors and income. Visualized key variables such as age, education, hours worked, and capital gains/losses.

**Missingness Check:** No missing data was present in the dataset, allowing for a streamlined preprocessing workflow without imputation.

**Feature Engineering:** Created derived variables like career\_stage (age bins), consolidated education levels, and flagged investment activity using capital features.

**Data Cleaning:** Standardized categorical levels, pruned low-variance features, and ensured consistent formatting across factor variables to support design matrix alignment.

**Encoding Strategy:** Applied one-hot encoding to categorical variables, carefully managing reference levels to avoid multicollinearity and NA coefficients.

These steps established a clean, interpretable, and structurally sound dataset for model comparison and final evaluation.

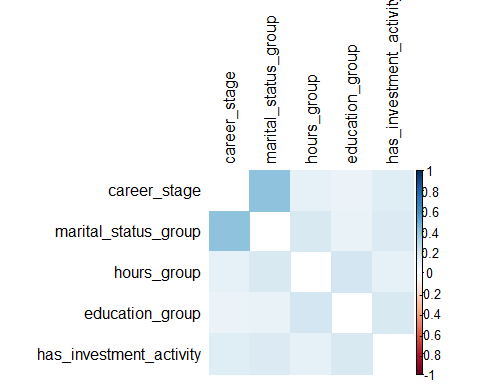
# Correlations

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)



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.

# 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.

I will maintain the natural 75/25 class split in the dataset so the models reflect the real-world distribution of outcomes.

# Train Models

## Linear Regression Indicator Matrix Model

### Prepare Indicator Matrices

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.

# 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)

### Fit Linear Indicator Model

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.  
  
model\_linear\_matrix <- lm(Y ~  
 poly(as.numeric(education\_group), 2) +  
 splines::ns(as.numeric(career\_stage), df = 3) +  
 marital\_status\_group +  
 has\_investment\_activity,  
 data = income\_train)

This model predicts income classification using a blend of nonlinear and categorical predictors. It captures quadratic effects of education level, flexible spline-based trends across career stages, and categorical shifts tied to marital status and investment activity.

By combining polynomial and spline transformations with group-level indicators, the model aims to uncover nuanced structural relationships while maintaining interpretability across key socioeconomic dimensions.

### Produce Linear Indicator Model Summary

Produce model summary.

summary(model\_linear\_matrix)

## Response factor(G)0 :  
##   
## Call:  
## lm(formula = `factor(G)0` ~ poly(as.numeric(education\_group),   
## 2) + splines::ns(as.numeric(career\_stage), df = 3) + marital\_status\_group +   
## has\_investment\_activity, data = income\_train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.15380 -0.15380 0.05249 0.26325 1.06305   
##   
## Coefficients:  
## Estimate Std. Error t value  
## (Intercept) 0.988545 0.004822 205.026  
## poly(as.numeric(education\_group), 2)1 -15.985273 0.356303 -44.864  
## poly(as.numeric(education\_group), 2)2 -0.014176 0.349017 -0.041  
## splines::ns(as.numeric(career\_stage), df = 3)1 -0.206840 0.013725 -15.071  
## splines::ns(as.numeric(career\_stage), df = 3)2 -0.142980 0.012942 -11.048  
## splines::ns(as.numeric(career\_stage), df = 3)3 -0.013001 0.009681 -1.343  
## marital\_status\_groupPreviously-married 0.016153 0.007930 2.037  
## marital\_status\_groupMarried -0.283326 0.006721 -42.156  
## has\_investment\_activityYes -0.256446 0.007597 -33.757  
## Pr(>|t|)   
## (Intercept) <2e-16 \*\*\*  
## poly(as.numeric(education\_group), 2)1 <2e-16 \*\*\*  
## poly(as.numeric(education\_group), 2)2 0.9676   
## splines::ns(as.numeric(career\_stage), df = 3)1 <2e-16 \*\*\*  
## splines::ns(as.numeric(career\_stage), df = 3)2 <2e-16 \*\*\*  
## splines::ns(as.numeric(career\_stage), df = 3)3 0.1793   
## marital\_status\_groupPreviously-married 0.0417 \*   
## marital\_status\_groupMarried <2e-16 \*\*\*  
## has\_investment\_activityYes <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.3465 on 19528 degrees of freedom  
## Multiple R-squared: 0.3435, Adjusted R-squared: 0.3432   
## F-statistic: 1277 on 8 and 19528 DF, p-value: < 2.2e-16  
##   
##   
## Response factor(G)1 :  
##   
## Call:  
## lm(formula = `factor(G)1` ~ poly(as.numeric(education\_group),   
## 2) + splines::ns(as.numeric(career\_stage), df = 3) + marital\_status\_group +   
## has\_investment\_activity, data = income\_train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.06305 -0.26325 -0.05249 0.15380 1.15380   
##   
## Coefficients:  
## Estimate Std. Error t value  
## (Intercept) 0.011455 0.004822 2.376  
## poly(as.numeric(education\_group), 2)1 15.985273 0.356303 44.864  
## poly(as.numeric(education\_group), 2)2 0.014176 0.349017 0.041  
## splines::ns(as.numeric(career\_stage), df = 3)1 0.206840 0.013725 15.071  
## splines::ns(as.numeric(career\_stage), df = 3)2 0.142980 0.012942 11.048  
## splines::ns(as.numeric(career\_stage), df = 3)3 0.013001 0.009681 1.343  
## marital\_status\_groupPreviously-married -0.016153 0.007930 -2.037  
## marital\_status\_groupMarried 0.283326 0.006721 42.156  
## has\_investment\_activityYes 0.256446 0.007597 33.757  
## Pr(>|t|)   
## (Intercept) 0.0175 \*   
## poly(as.numeric(education\_group), 2)1 <2e-16 \*\*\*  
## poly(as.numeric(education\_group), 2)2 0.9676   
## splines::ns(as.numeric(career\_stage), df = 3)1 <2e-16 \*\*\*  
## splines::ns(as.numeric(career\_stage), df = 3)2 <2e-16 \*\*\*  
## splines::ns(as.numeric(career\_stage), df = 3)3 0.1793   
## marital\_status\_groupPreviously-married 0.0417 \*   
## marital\_status\_groupMarried <2e-16 \*\*\*  
## has\_investment\_activityYes <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.3465 on 19528 degrees of freedom  
## Multiple R-squared: 0.3435, Adjusted R-squared: 0.3432   
## F-statistic: 1277 on 8 and 19528 DF, p-value: < 2.2e-16

For Class 0 (<=50k), the intercept is high (≈0.99), and education has a strong negative linear effect. Career stage splines show significant negative associations, while being married or having investment activity substantially lowers the predicted probability of income <=50k.

For Class 1 (>50K), the intercept is low (≈0.01), and the signs of all coefficients are reversed, as expected in a complementary indicator setup. Education, career stage, marital status, and investment activity all positively influence the likelihood of income >50K.

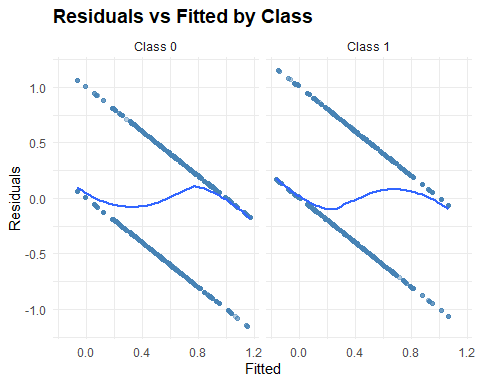
Both models show strong overall fit (adjusted R-squared of 0.343), highly significant predictors, and symmetric residual distributions, suggesting the indicator matrix is capturing meaningful structure. The second-order education term and third spline basis are not significant, suggesting possible overparameterization.

### Plot Raw Residuals

Raw residual plots display the individual prediction errors for each observation, plotted against their fitted values. This helps assess whether the model’s errors are randomly distributed or show patterns that suggest structural issues.

# Capture fitted values and residuals.  
  
income\_train$mlm\_resid\_0 <- residuals(model\_linear\_matrix)[,1]  
income\_train$mlm\_fitted\_0 <- fitted(model\_linear\_matrix)[,1]  
income\_train$mlm\_resid\_1 <- residuals(model\_linear\_matrix)[,2]  
income\_train$mlm\_fitted\_1 <- fitted(model\_linear\_matrix)[,2]  
  
# Convert to long form.  
  
income\_long <- income\_train %>%  
 select(mlm\_fitted\_0, mlm\_resid\_0, mlm\_fitted\_1, mlm\_resid\_1) %>%  
 pivot\_longer(  
 cols = everything(),  
 names\_to = c(".value", "class"),  
 names\_pattern = "(mlm\_fitted|mlm\_resid)\_(\\d)"  
 ) %>%  
 mutate(class = paste("Class", class),  
 sqrt\_abs\_resid = sqrt(abs(mlm\_resid)))  
  
# Plot residuals.  
  
ggplot(income\_long, aes(x = mlm\_fitted, y = mlm\_resid)) +  
 geom\_point(alpha = 0.4, color = "steelblue") +  
 geom\_smooth(method = "loess", se = FALSE) +  
 facet\_wrap(~ class) +  
 labs(title = "Residuals vs Fitted by Class", x = "Fitted", y = "Residuals") +  
 theme\_minimal() +   
 theme(axis.text.x = element\_text(angle = 0, hjust = 0.5),  
 plot.title = element\_text(size = 14, face = "bold"))

## `geom\_smooth()` using formula = 'y ~ x'

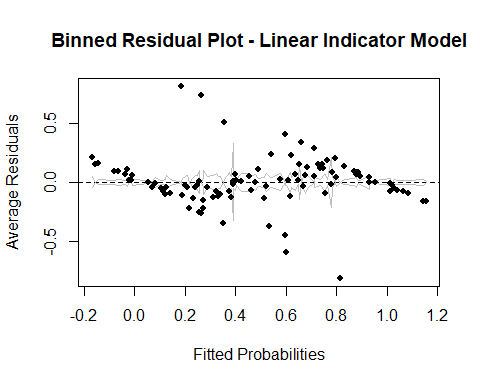


In our case, the residuals form a funnel shape, indicating heteroscedasticity—where error variance increases with predicted probability. The smooth trend line also reveals nonlinearity, suggesting the model may be missing key interactions or transformations.

### Plot Binned Residuals

Binned residual plots aggregate residuals into intervals of fitted probabilities, averaging them to reduce noise and highlight systematic bias. This makes it easier to detect under- or over-prediction across the probability spectrum.

# Produced binned residuals.  
  
arm::binnedplot(x = fitted(model\_linear\_matrix),  
 y = residuals(model\_linear\_matrix, type = "response"),  
 xlab = "Fitted Probabilities",  
 ylab = "Average Residuals",  
 main = "Binned Residual Plot - Linear Indicator Model")



The binned residuals deviate from zero in a curved pattern, especially at the extremes, indicating that the model consistently mis-estimates probabilities in those regions. This reinforces the need for more flexible modeling to correct structural misfit and improve calibration.

Unbounded probabilities, i.e. the values predicted below 0 or above 1, are a clear sign of model misalignment when working with binary outcomes. In the context of a linear indicator model, this occurs because the model treats probabilities as continuous outputs without a constraining link function.

Despite its limitations, I will continue to use the linear indicator model for comparison purposes.

### Predict Class Scores

Predict class scores initially with training data.

# 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)

### Generate Confusion Matrix

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 14056 2653  
## 1 776 2052  
##   
## Accuracy : 0.8245   
## 95% CI : (0.8191, 0.8298)  
## No Information Rate : 0.7592   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4443   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9477   
## Specificity : 0.4361   
## Pos Pred Value : 0.8412   
## Neg Pred Value : 0.7256   
## Prevalence : 0.7592   
## Detection Rate : 0.7195   
## Detection Prevalence : 0.8552   
## Balanced Accuracy : 0.6919   
##   
## 'Positive' Class : 0   
##

The linear indicator model achieves strong overall accuracy (82.5%) and high sensitivity (94.8%) for the majority class (Class 0), correctly identifying most true positives. However, its specificity is low (43.6%), indicating frequent misclassification of minority class cases.

The Kappa score (0.44) reflects moderate agreement beyond chance, and the significant McNemar’s test suggests asymmetry in classification errors. While the model performs well in detecting the dominant class, its limited balance and specificity highlight challenges in handling class imbalance and underscore the need for complementary diagnostics.

### 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.

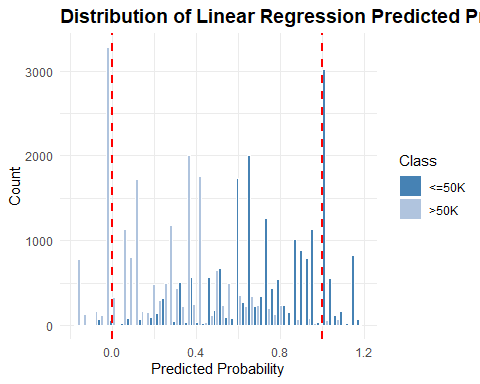
# Count out-of-bounds for each class.  
  
out\_under <- sum(prob\_under50k < 0 | prob\_under50k > 1)  
  
out\_over <- sum(prob\_over50k < 0 | prob\_over50k > 1)  
  
total <- nrow(train\_pred)  
  
# Summary table.  
  
check\_summary <- data.frame(Cclass = c("<=50K", ">50K"),  
 out\_of\_bounds = 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 | 4,909 | 19,537 | 25.13 |
| >50K | 4,909 | 19,537 | 25.13 |

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", linewidth = 1, color = "red") +  
 geom\_vline(xintercept = 1, linetype = "dashed", linewidth = 1, 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() +   
 theme(axis.text.x = element\_text(angle = 0, hjust = 0.5),  
 plot.title = element\_text(size = 14, face = "bold"))



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

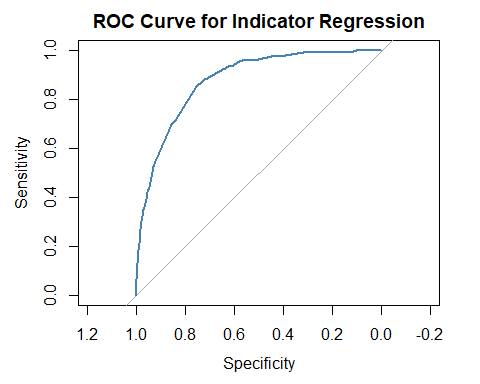
To evaluate the discriminatory power of the indicator regression model, extract the predicted probabilities for the positive class and used them as inputs to generate a Receiver Operating Characteristic (ROC) curve.

# 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: 0.8773

This AUC score indicates that the model can distinguish between >50K and <=50K earners with 87.7% accuracy across all thresholds.

## Logistic Regression

### Fit Logistic Regression Model

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 ~   
 poly(as.numeric(education\_group), 2) +  
 splines::ns(as.numeric(career\_stage), df = 3) +  
 marital\_status\_group +  
 has\_investment\_activity,  
 data = income\_train, family = "binomial")

This logistic regression model predicts income classification using the same set of predictors as the linear indicator model. However, by using a binomial link, the model ensures fitted probabilities remain bounded between 0 and 1, offering coherent and interpretable outputs.

### Produce Logistic Regression Model Summary

Produce model summary.

summary(model\_logistic)

##   
## Call:  
## glm(formula = income ~ poly(as.numeric(education\_group), 2) +   
## splines::ns(as.numeric(career\_stage), df = 3) + marital\_status\_group +   
## has\_investment\_activity, family = "binomial", data = income\_train)  
##   
## Coefficients:  
## Estimate Std. Error z value  
## (Intercept) -4.05108 0.07762 -52.192  
## poly(as.numeric(education\_group), 2)1 123.35518 3.27177 37.703  
## poly(as.numeric(education\_group), 2)2 -31.33095 3.48520 -8.990  
## splines::ns(as.numeric(career\_stage), df = 3)1 1.61269 0.10616 15.192  
## splines::ns(as.numeric(career\_stage), df = 3)2 1.76834 0.12739 13.881  
## splines::ns(as.numeric(career\_stage), df = 3)3 0.26857 0.08116 3.309  
## marital\_status\_groupPreviously-married 0.27507 0.08941 3.076  
## marital\_status\_groupMarried 2.45492 0.07197 34.109  
## has\_investment\_activityYes 1.56766 0.05678 27.608  
## Pr(>|z|)   
## (Intercept) < 2e-16 \*\*\*  
## poly(as.numeric(education\_group), 2)1 < 2e-16 \*\*\*  
## poly(as.numeric(education\_group), 2)2 < 2e-16 \*\*\*  
## splines::ns(as.numeric(career\_stage), df = 3)1 < 2e-16 \*\*\*  
## splines::ns(as.numeric(career\_stage), df = 3)2 < 2e-16 \*\*\*  
## splines::ns(as.numeric(career\_stage), df = 3)3 0.000937 \*\*\*  
## marital\_status\_groupPreviously-married 0.002095 \*\*   
## marital\_status\_groupMarried < 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: 21570 on 19536 degrees of freedom  
## Residual deviance: 13953 on 19528 degrees of freedom  
## AIC: 13971  
##   
## Number of Fisher Scoring iterations: 6

All predictors are highly significant, with education showing strong curvature and marital status, especially being married, exerting a substantial positive effect on income probability.

The model achieves a notable reduction in deviance and a well-calibrated fit, with bounded, interpretable probabilities and a compact AIC of 13,971.

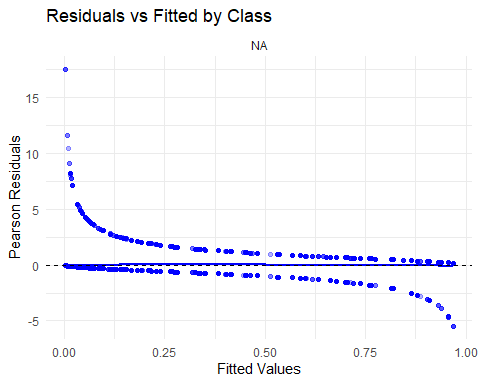
Its structure mirrors the linear indicator model but benefits from coherent probabilistic outputs and improved diagnostic behavior.

### Plot Raw Residuals

In the raw Pearson residual plot, residuals are more symmetrically distributed around zero, with less pronounced funneling, suggesting reduced heteroscedasticity and better variance stability across fitted values.

# Create a dataframe with fitted values and residuals.  
  
resid\_df <- data.frame(fitted = fitted(model\_logistic),  
 residuals = residuals(model\_logistic, type = "pearson"),  
 actual = income\_train$income)  
  
# Convert actual outcome to factor for faceting.  
  
resid\_df$class <- factor(resid\_df$actual, levels = c(0, 1), labels = c("Class 0", "Class 1"))  
  
# Plot residuals vs. fitted by class.  
  
ggplot(resid\_df, aes(x = fitted, y = residuals)) +  
 geom\_point(alpha = 0.3, color = "blue") +  
 geom\_smooth(method = "loess", se = TRUE, color = "blue", linewidth = 0.8) +  
 facet\_wrap(~ class) +  
 geom\_hline(yintercept = 0, linetype = "dashed") +  
 labs(title = "Residuals vs Fitted by Class",  
 x = "Fitted Values",  
 y = "Pearson Residuals") +  
 theme\_minimal()

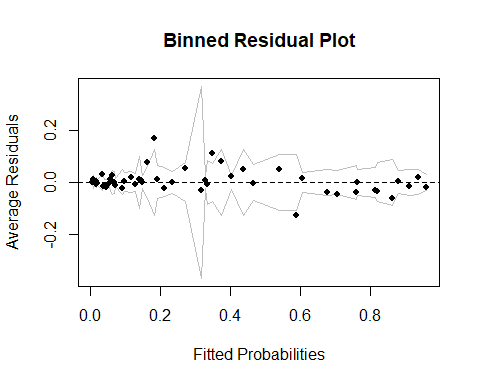
## `geom\_smooth()` using formula = 'y ~ x'



### Plot Binned Residuals

In the binned residual plot, residuals are more tightly clustered around the zero line (compared to the corresponding linear indicator plot), with minimal curvature and narrower confidence bands, indicating improved calibration and reduced bias.

arm::binnedplot(x = fitted(model\_logistic),  
 y = residuals(model\_logistic, type = "response"),  
 xlab = "Fitted Probabilities",  
 ylab = "Average Residuals",  
 main = "Binned Residual Plot")



The logistic model’s bounded predictions and probabilistic framework yield cleaner diagnostics and more reliable interpretability across the probability spectrum.

### Predict Class Scores

Extract the predicted probabilities for the positive class.

income\_train$predicted\_prob <- predict(model\_logistic, type = "response")

### 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 13926 2350  
## 1 906 2355  
##   
## Accuracy : 0.8333   
## 95% CI : (0.828, 0.8385)  
## No Information Rate : 0.7592   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4909   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9389   
## Specificity : 0.5005   
## Pos Pred Value : 0.8556   
## Neg Pred Value : 0.7222   
## Prevalence : 0.7592   
## Detection Rate : 0.7128   
## Detection Prevalence : 0.8331   
## Balanced Accuracy : 0.7197   
##   
## 'Positive' Class : 0   
##

The logistic model shows modest gains over the linear indicator model in classification performance. Accuracy improves slightly (83.3% vs. 82.5%), and balanced accuracy rises from 0.692 to 0.720, reflecting better handling of both classes. Sensitivity remains high for the majority class (Class 0), while specificity improves from 43.6% to 50.1%, indicating fewer false positives. The Kappa score also increases, suggesting stronger agreement beyond chance. Overall, the logistic model offers more coherent probability estimates and a better balance between detecting true positives and minimizing misclassification of the minority class.

### Generate ROC Curve and AUC Metric

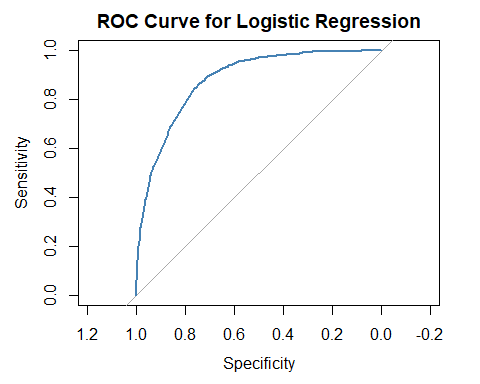
Generate a Receiver Operating Characteristic (ROC) curve.

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: 0.8802

This AUC score indicates that the model can distinguish between >50K and <=50K earners with 88% accuracy across all thresholds, which is slightly better than the linear indicator model.

# 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 | 14,056 |
| Linear | 1 | 0 | FP | 776 |
| Linear | 0 | 1 | FN | 2,653 |
| Linear | 1 | 1 | TP | 2,052 |
| Logistic | 0 | 0 | TN | 13,926 |
| Logistic | 1 | 0 | FP | 906 |
| Logistic | 0 | 1 | FN | 2,350 |
| Logistic | 1 | 1 | TP | 2,355 |

Compared to the linear indicator model, the logistic model shows a trade-off between false positives and true positives. While it slightly reduces true negatives (13,926 vs. 14,056), it captures more true positives (2,355 vs. 2,052) and fewer false negatives (2,350 vs. 2,653), indicating improved sensitivity. However, this comes with a modest increase in false positives (906 vs. 776). Overall, the logistic model offers better balance in classification, particularly in identifying the minority class more effectively.

### 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 | 0.8245 | 0.8333 |
| Kappa | 0.4443 | 0.4909 |
| Sensitivity | 0.9477 | 0.9389 |
| Specificity | 0.4361 | 0.5005 |
| Pos Pred Value | 0.8412 | 0.8556 |
| Neg Pred Value | 0.7256 | 0.7222 |
| Balanced Accuracy | 0.6919 | 0.7197 |

The logistic model outperforms the linear indicator model across most classification metrics. It shows higher overall accuracy (83.3% vs. 82.5%), stronger agreement beyond chance (Kappa: 0.49 vs. 0.44), and notably better specificity (50.1% vs. 43.6%), indicating improved detection of the minority class. While sensitivity remains high for both models, the logistic approach offers a more balanced and reliable classification profile, with gains in positive predictive value and balanced accuracy that reflect better calibration and class discrimination.

### 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 | 0.8772935 |
| Logistic | 0.8801876 |

The logistic model shows a slight edge in discriminative performance, with an AUC of 0.880 compared to 0.877 for the linear indicator model. While both models demonstrate strong ability to separate classes, the logistic approach offers marginally better ranking of predicted probabilities.

## Validation Data

### Linear Regression with an Indicator Matrix

#### Predict Class Scores

# 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)

#### Generate Confusion Matrix

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 4692 894  
## 1 252 674  
##   
## Accuracy : 0.824   
## 95% CI : (0.8145, 0.8332)  
## No Information Rate : 0.7592   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4404   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9490   
## Specificity : 0.4298   
## Pos Pred Value : 0.8400   
## Neg Pred Value : 0.7279   
## Prevalence : 0.7592   
## Detection Rate : 0.7205   
## Detection Prevalence : 0.8578   
## Balanced Accuracy : 0.6894   
##   
## 'Positive' Class : 0   
##

On the validation set, the linear indicator model maintains strong overall accuracy (82.4%) and high sensitivity (94.9%) for the majority class, effectively identifying true positives. However, specificity remains low (43%), indicating frequent misclassification of minority class cases. The Kappa score (0.44) reflects moderate agreement beyond chance, and the balanced accuracy of 0.689 highlights the model’s limited ability to handle class imbalance. Overall, the model performs reliably for dominant class detection but struggles with minority class discrimination.

### Logistic Regression

#### Predict Class Scores

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)

#### Generate Confusion Matrix

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 4653 783  
## 1 291 785  
##   
## Accuracy : 0.8351   
## 95% CI : (0.8258, 0.844)  
## No Information Rate : 0.7592   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4948   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9411   
## Specificity : 0.5006   
## Pos Pred Value : 0.8560   
## Neg Pred Value : 0.7296   
## Prevalence : 0.7592   
## Detection Rate : 0.7145   
## Detection Prevalence : 0.8348   
## Balanced Accuracy : 0.7209   
##   
## 'Positive' Class : 0   
##

On the validation set, the logistic model delivers improved classification performance over its linear counterpart. It achieves higher accuracy (83.5%) and a stronger Kappa score (0.49), indicating better agreement beyond chance. Sensitivity remains high (94.1%) for the majority class, while specificity improves to 50.1%, reflecting more balanced detection of minority cases. The model also shows gains in positive predictive value and balanced accuracy (72.1%), reinforcing its strength in both calibration and class discrimination.

### Compare Metrics From Validation Data

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 | 0.8245 | 0.8333 | 0.8240 | 0.8351 |
| Kappa | 0.4443 | 0.4909 | 0.4404 | 0.4948 |
| Sensitivity | 0.9477 | 0.9389 | 0.9490 | 0.9411 |
| Specificity | 0.4361 | 0.5005 | 0.4298 | 0.5006 |
| Pos Pred Value | 0.8412 | 0.8556 | 0.8400 | 0.8560 |
| Neg Pred Value | 0.7256 | 0.7222 | 0.7279 | 0.7296 |
| Balanced Accuracy | 0.6919 | 0.7197 | 0.6894 | 0.7209 |

Across both training and validation sets, the logistic model consistently outperforms the linear indicator model in key classification metrics. It shows higher accuracy, stronger Kappa scores, and notably better specificity and balanced accuracy—indicating improved detection of the minority class without sacrificing performance on the majority class. While sensitivity remains high for both models, the logistic approach offers more reliable predictive balance and agreement, making it the more robust choice for calibrated classification.

# Choose Final Model

I chose the logistic model as the final model due to its consistently stronger performance across both training and validation sets. It offers higher accuracy, better class balance, and more reliable specificity, all while producing coherent, bounded probability estimates. These advantages make it a more interpretable and calibrated choice for classification, especially in the presence of class imbalance.

## 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(-predicted\_prob,  
 -predicted\_class,  
 -mlm\_resid\_0,  
 -mlm\_resid\_1,  
 -mlm\_fitted\_0,  
 -mlm\_fitted\_1)  
  
income\_full <- rbind(income\_train, income\_validate)

Produce the final model.

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

##   
## Call:  
## glm(formula = income ~ poly(as.numeric(education\_group), 2) +   
## splines::ns(as.numeric(career\_stage), df = 3) + marital\_status\_group +   
## has\_investment\_activity, family = "binomial", data = income\_full)  
##   
## Coefficients:  
## Estimate Std. Error z value  
## (Intercept) -4.06388 0.06746 -60.237  
## poly(as.numeric(education\_group), 2)1 140.66376 3.26100 43.135  
## poly(as.numeric(education\_group), 2)2 -37.16265 3.48861 -10.653  
## splines::ns(as.numeric(career\_stage), df = 3)1 1.66831 0.09213 18.108  
## splines::ns(as.numeric(career\_stage), df = 3)2 1.74923 0.11067 15.806  
## splines::ns(as.numeric(career\_stage), df = 3)3 0.23217 0.07043 3.297  
## marital\_status\_groupPreviously-married 0.30148 0.07777 3.877  
## marital\_status\_groupMarried 2.48003 0.06273 39.534  
## has\_investment\_activityYes 1.53180 0.04911 31.188  
## Pr(>|z|)   
## (Intercept) < 2e-16 \*\*\*  
## poly(as.numeric(education\_group), 2)1 < 2e-16 \*\*\*  
## poly(as.numeric(education\_group), 2)2 < 2e-16 \*\*\*  
## splines::ns(as.numeric(career\_stage), df = 3)1 < 2e-16 \*\*\*  
## splines::ns(as.numeric(career\_stage), df = 3)2 < 2e-16 \*\*\*  
## splines::ns(as.numeric(career\_stage), df = 3)3 0.000979 \*\*\*  
## marital\_status\_groupPreviously-married 0.000106 \*\*\*  
## marital\_status\_groupMarried < 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: 28759 on 26048 degrees of freedom  
## Residual deviance: 18583 on 26040 degrees of freedom  
## AIC: 18601  
##   
## Number of Fisher Scoring iterations: 6

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 4641 763  
## 1 302 805  
##   
## Accuracy : 0.8364   
## 95% CI : (0.8272, 0.8453)  
## No Information Rate : 0.7592   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5028   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9389   
## Specificity : 0.5134   
## Pos Pred Value : 0.8588   
## Neg Pred Value : 0.7272   
## Prevalence : 0.7592   
## Detection Rate : 0.7128   
## Detection Prevalence : 0.8300   
## Balanced Accuracy : 0.7261   
##   
## 'Positive' Class : 0   
##

On the unseen test set, the logistic model maintains strong and balanced performance. It achieves 83.6% accuracy with a Kappa of 0.50, indicating solid agreement beyond chance. Sensitivity remains high (93.9%) for the majority class, while specificity improves to 51.3%, reflecting better minority class detection. Balanced accuracy reaches 72.6%, confirming the model’s reliability and generalizability across class boundaries.

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 | 0.8245 | 0.8333 | 0.8240 | 0.8351 | 0.8364 |
| Kappa | 0.4443 | 0.4909 | 0.4404 | 0.4948 | 0.5028 |
| Sensitivity | 0.9477 | 0.9389 | 0.9490 | 0.9411 | 0.9389 |
| Specificity | 0.4361 | 0.5005 | 0.4298 | 0.5006 | 0.5134 |
| Pos Pred Value | 0.8412 | 0.8556 | 0.8400 | 0.8560 | 0.8588 |
| Neg Pred Value | 0.7256 | 0.7222 | 0.7279 | 0.7296 | 0.7272 |
| Balanced Accuracy | 0.6919 | 0.7197 | 0.6894 | 0.7209 | 0.7261 |

The logistic model consistently outperforms the linear indicator model across all datasets. It delivers higher accuracy, stronger Kappa scores, and notably better specificity and balanced accuracy—indicating improved minority class detection without compromising sensitivity. These gains hold steady from training through validation to the unseen test set, confirming the logistic model’s robustness, calibration, and generalizability for final deployment.

# Final Analysis

## Conclusions

This analysis compared linear and logistic regression for classifying income using socioeconomic predictors from the UCI Adult dataset. Logistic regression consistently outperformed the linear model across training, validation, and test sets, showing stronger accuracy, balance, and agreement beyond chance.

Its bounded probabilities and alignment with binary classification theory made it more interpretable and reliable, especially for detecting minority-class cases. While the linear model was competitive, it suffered from unbounded outputs and thresholding limitations.

Logistic regression was selected as the final model for its predictive strength, theoretical coherence, and suitability for transparent reporting and fairness auditing.

## Limitations of Linear Regression as a Classifier

Linear regression can be adapted for binary classification via thresholding, but it presents several key limitations:

**Unbounded predictions:** Outputs are not restricted to the [0, 1] range, making probability interpretation invalid and thresholding arbitrary.

**Assumption violations:** Linear regression relies on homoscedasticity and normally distributed errors, conditions that rarely hold in classification settings, leading to biased estimates and unreliable inference.

**Poor calibration:** Predicted values do not correspond to true class probabilities, weakening decision-making in threshold-sensitive contexts.

**Lack of log-odds interpretation:** Coefficients lack the intuitive odds-based framing that logistic regression provides, complicating stakeholder communication.

These limitations motivated my transition to logistic regression, which offers bounded, interpretable outputs and better theoretical alignment with classification tasks.

## Challenges and Solutions

| Challenge | Solution |
| --- | --- |
| Unbounded predictions from linear regression | Quantified the extent of unbounded outputs and highlighted the model’s misalignment with binary classification, reinforcing the need for bounded alternatives. |
| Communicating metric tradeoffs to stakeholders | Developed concise, audience-aware narratives emphasizing practical impact (e.g., “Logistic regression better identifies low-income individuals without sacrificing accuracy”). |
| Balancing interpretability with predictive strength | Chose logistic regression for its bounded outputs, theoretical coherence, and transparent coefficient-based insights into income drivers. |

# Appendix A: 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 |

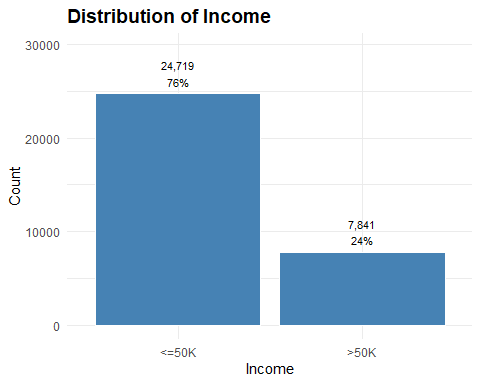
The table confirms there are no rows with missing data in the reported rows and columns. The source dataset contains 32,560 observations.

## 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.

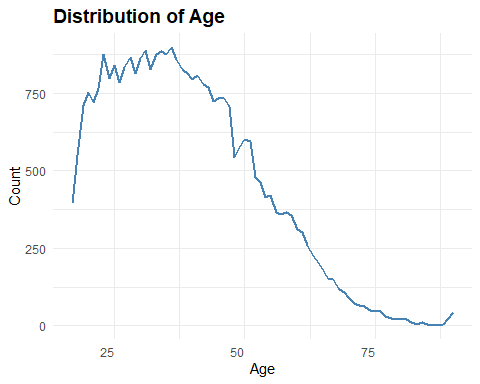
Encode factor levels for income, using <=50K as the reference level.

# 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.

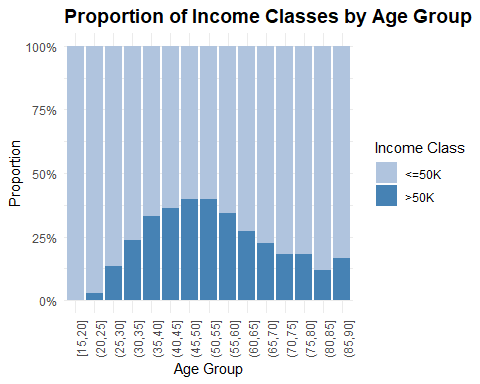
# Generate a summary of the age variable.  
  
age\_summary <- income %>%  
 group\_by(age) %>%  
 summarise(entries = n()) %>%  
 mutate(percent = round(entries / sum(entries), 2))  
  
# Plot the distribution of values.  
  
ggplot(age\_summary, aes(x = age, y = entries)) +  
 geom\_line(linewidth = 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.

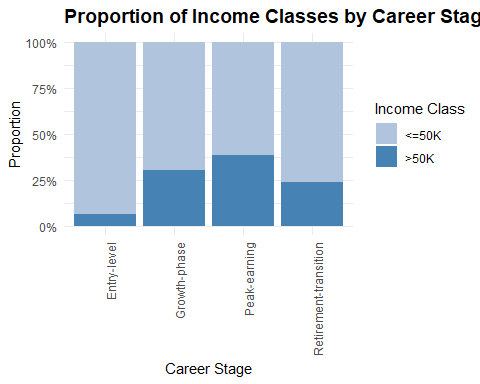
## Career Stage

The career\_stage feature will contain four groups: 1) entry-level (ages 17-30), 2) growth phase (ages 31-44), 3) peak earning (ages 45-60), 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 >= 31 & age < 45 ~ "Growth-phase",  
 age >= 45 & 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 test 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).

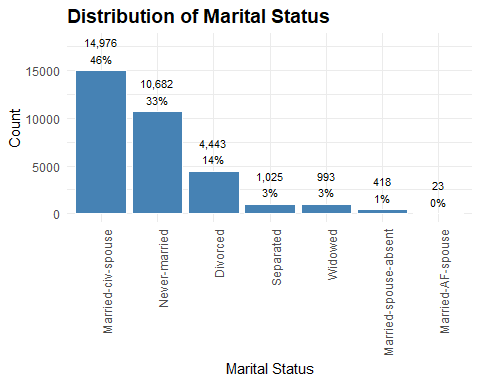
## Marital Status

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.

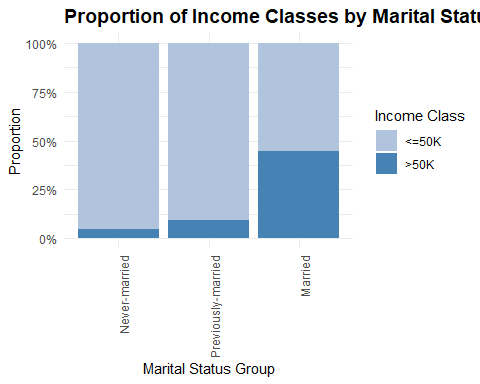
## Marital Status Group

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"))



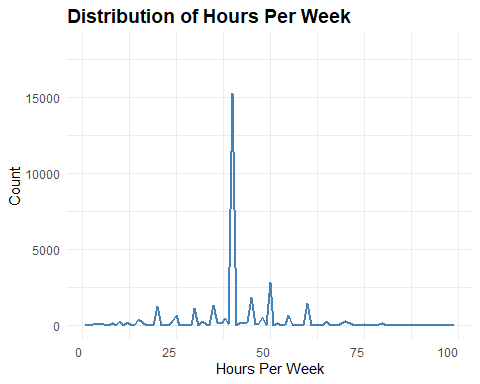
Married individuals show the highest proportion of income earners in the >50K category. This suggests that marriage may correlate with financial stability, dual-income households, or career maturity.

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

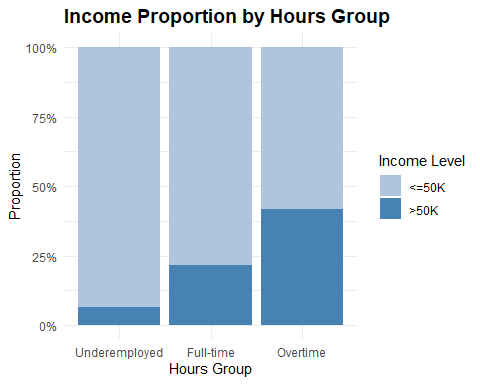
The hours group variable is a derived categorical feature that segments individuals based on their reported weekly work hours:

* 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"))

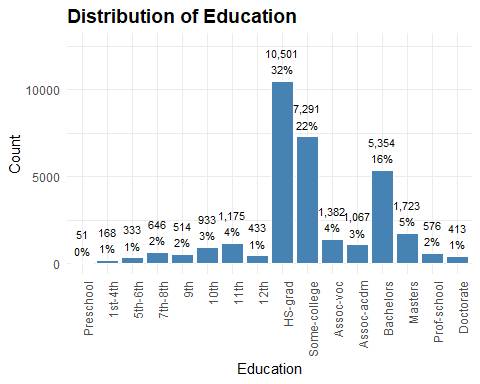


The underemployed group is overwhelmingly low income and likely includes part-time, seasonal or precarious workers. The full-time group is majority <=$50k but has a noticeable uptick in high earners. This suggests that full-time work alone isn’t a guarantee of higher income. The overtime group has the 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.

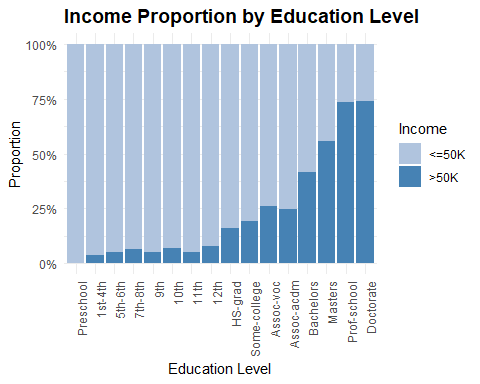
# Make variable a factor.  
  
income <- income %>%  
 mutate(education = factor(education,   
 levels = c("Preschool",   
 "1st-4th",   
 "5th-6th",   
 "7th-8th",   
 "9th",   
 "10th",   
 "11th",   
 "12th",  
 "HS-grad",   
 "Some-college",   
 "Assoc-voc",   
 "Assoc-acdm",  
 "Bachelors",  
 "Masters",  
 "Prof-school",   
 "Doctorate")))  
  
education\_summary <- income %>%  
 group\_by(education) %>%  
 summarise(entries = n()) %>%  
 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.

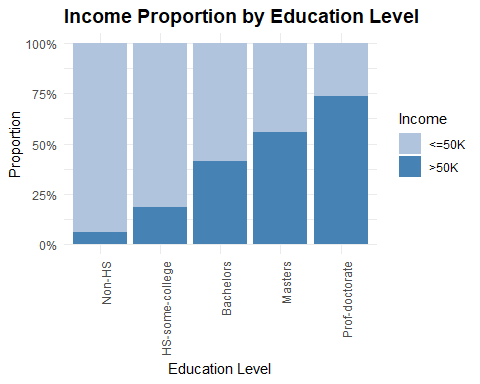
## Education Group

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 test 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

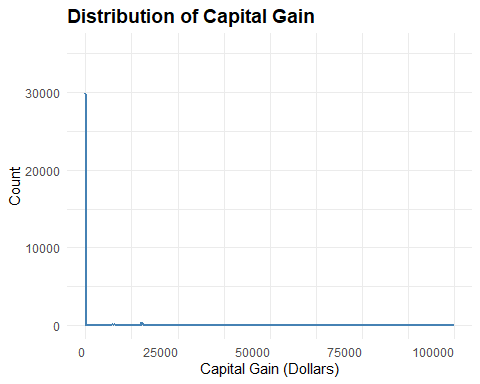
While capital gain and capital loss provide useful signals of investment activity, they capture only realized transactions (profits or losses from assets actually sold). They do not account for asset ownership or wealth holdings that have not been sold. For example, someone may hold significant investments 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.

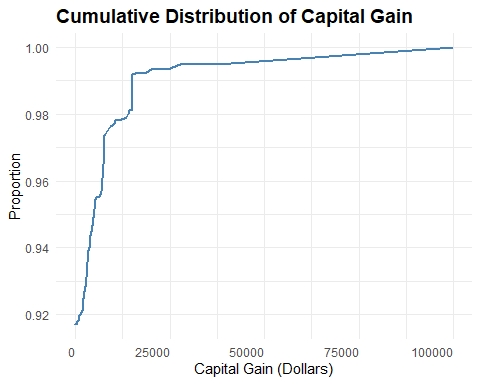
# Generate summary.  
  
capital\_gain\_summary <- income %>%  
 group\_by(capital\_gain) %>%  
 summarise(entries = n(), .groups = "drop") %>%  
 mutate(percent = round(entries / sum(entries), 2))  
  
# Plot summary.  
  
ggplot(capital\_gain\_summary, aes(x = capital\_gain, y = entries)) +  
 geom\_line(linewidth = 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"))



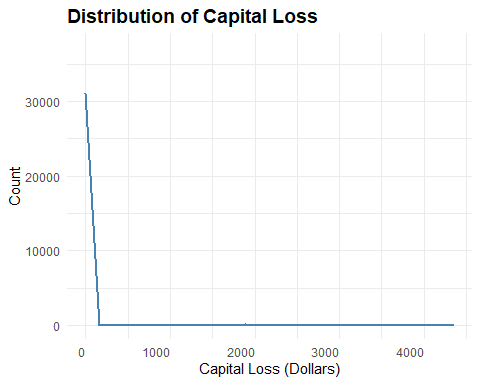
The cumulative distribution shows that over 92% of individuals report zero or minimal capital gains, with only a small fraction earning substantial amounts (e.g., >10,000).

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.

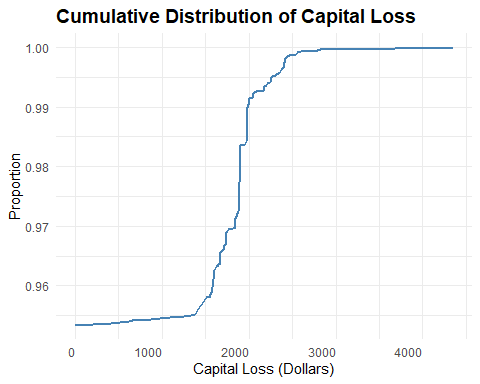
# Generate summary.  
  
capital\_loss\_summary <- income %>%  
 group\_by(capital\_loss) %>%  
 summarise(entries = n(), .groups = "drop") %>%  
 mutate(percent = round(entries / sum(entries), 2))  
  
# Plot summary.  
  
ggplot(capital\_loss\_summary, aes(x = capital\_loss, y = entries)) +  
 geom\_line(linewidth = 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.

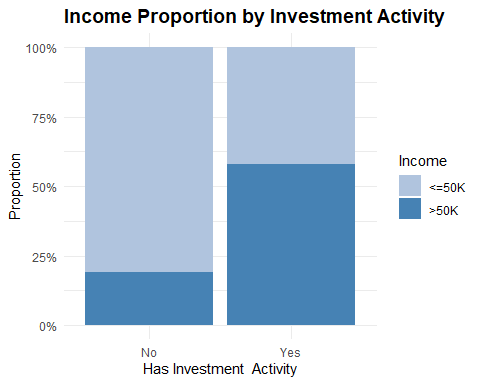
# Create investment activity variable.  
  
income <- income %>%  
 mutate(has\_investment\_activity = if\_else(capital\_gain > 0 | capital\_loss > 0, "Yes", "No"))  
  
# Make variable a factor.  
  
income <- income %>%  
 mutate(has\_investment\_activity = factor(has\_investment\_activity,  
 ordered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