Income_Prediction_Model

Amedee Jacques

2025-06-03

Introduction

This analysis utilizes the Adult “Income” dataset from the Irvine Machine Learning Repository to develop a predictive model that forecasts whether an individual’s annual income exceeds $50,000, based on various demographic variables. The target variable is categorical, representing income as either below or above this threshold, making this a binary classification task.

Several models were tested during this analysis, broadly falling into two categories: those implemented using the caret package and those using the glmnet package. The models were evaluated and compared through a performance heat map that included metrics such as AUC, F1 score, accuracy, sensitivity, specificity, and others. It is worth noting that the importance of these metrics may vary depending on the end user’s objectives.

Initially, I considered using RMSE (as in the previous Movie Ratings project), but it was not suitable for this task due to the categorical nature of the outcome variable. Consequently, AUC and F1 score were chosen as more appropriate evaluation criteria.

In my view, the penalized Elastic Net Regression model implemented using glmnet performed best. It showed strong results across several metrics, achieving an AUC of 0.91 and an accuracy of 84% in predicting whether an individual earns more than $50K annually. Although the XGBoost model had a slightly higher accuracy (87%), it underperformed in terms of AUC and sensitivity, making Elastic Net the preferred choice.

This report outlines the analysis from data import and preprocessing to feature engineering, model fitting, and evaluation. Ultimately, all models are compared via a consolidated performance heat map.

The report is structured into four main sections, each with further subdivisions:

1.1 Initialization

This section initializes the R environment and imports the necessary data for the analysis. It also ensures that required packages are loaded and the data file is unzipped and ready for use.

```r
# Import dataset from the ICS database
initial_data <- "adult.zip"
if(!file.exists(initial_data))
  download.file("https://archive.ics.uci.edu/static/public/2/adult.zip",
initial_data)

adult_data <- "adult.data"
if(!file.exists(adult_data))
  unzip(initial_data, adult_data)

# Load data
adult_income <- read.csv(adult_data, header = FALSE, strip.white = TRUE,
stringsAsFactors = FALSE)
```

2. METHOD ANALYSIS

This analysis is based on the following assumptions/hypotheses:

Individual income is influenced by a variety of factors, including age, education level, marital status, gender, work class, and other explanatory variables.

2.1 Features engineering

Minor adjustments were made to the raw dataset, such as renaming columns and converting certain variables into factors. The income column was transformed into a binary factor (0 and 1) to avoid issues related to logistic regression, a challenge that took me a while to resolve. Additionally, dummy variables were created to prepare for the glmnet modeling step.

```r
# Name columns
colnames(adult_income) <- c("age", "workclass", "fnlwgt", "education",
"education_num",
                            "marital_status", "occupation", "relationship",
"race",
                            "sex", "capital_gain", "capital_loss",
"hours_per_week",
                            "native_country", "income")

# Set non-numerical variables as factors
adult_income <- adult_income %>% drop_na() %>%
  mutate(across(c(workclass, education, marital_status, occupation,
                  relationship, race, sex, native_country, income),
as.factor))

# Split data into a training and a test set
set.seed(123)
train_index <- createDataPartition(adult_income$income, p = 0.7, list =
FALSE)
train_data <- adult_income[train_index, ]
test_data <- adult_income[-train_index, ]

# Create two versions of the income column to accommodate different models
convert_income <- function(data, target = "income", type = c("factor",
"numeric")) {
  type <- match.arg(type)
  if (type == "factor") {
    data[[target]] <- factor(data[[target]], levels = c("<=50K", ">50K"),
labels = c("low", "high"))
  } else {
    data[[target]] <- ifelse(data[[target]] == ">50K", 1, 0)
  }
  return(data)
}

# Refine the training set and the test set for different models
train_data_factor <- convert_income(train_data, type = "factor")
test_data_factor <- convert_income(test_data, type = "factor")
train_data_numeric <- convert_income(train_data, type = "numeric")
test_data_numeric <- convert_income(test_data, type = "numeric")

ctrl <- trainControl(method = "cv", number = 5, classProbs = TRUE,
                     summaryFunction = twoClassSummary, savePredictions =
TRUE)

# Create common dummyVars object. Glmnet will need them.
xform <- dummyVars(income ~ ., data = train_data_factor)
train_x <- predict(xform, newdata = train_data_factor) %>% as.matrix()

test_x <- predict(xform, newdata = test_data_factor) %>% as.matrix()

train_y <- train_data_factor$income
test_y <- test_data_factor$income

# Pre-stage the models' evaluation and comparison
evaluate_model <- function(pred_class, prob, true_labels, label) {
  cm <- confusionMatrix(pred_class, true_labels, positive = "high")
  roc_obj <- roc(true_labels, prob, levels = c("low", "high"), direction =
">")
  TP <- cm$table[2, 2]; FP <- cm$table[2, 1]; FN <- cm$table[1, 2]
  precision <- ifelse((TP + FP) == 0, NA, TP / (TP + FP))
  recall <- ifelse((TP + FN) == 0, NA, TP / (TP + FN))
  f1 <- ifelse(is.na(precision) | is.na(recall) | (precision + recall == 0),
NA,
               2 * precision * recall / (precision + recall))
  logloss <- logLoss(ifelse(true_labels == "high", 1, 0), prob)
  data.frame(Model = label, AUC = as.numeric(auc(roc_obj)), F1 = f1,
             Accuracy = cm$overall["Accuracy"],
             Sensitivity = cm$byClass["Sensitivity"],
             Specificity = cm$byClass["Specificity"],
             Precision = precision,
             LogLoss = logloss)
}
results <- list()
```

2.2 Early sub-optimal models

Below are some early models I built using the caret package. These models—logistic regression, random forest, XGBoost, GBM, and a neural network—were ultimately sub-optimal, particularly in terms of AUC performance. While some of them achieved decent accuracy, they were not as robust or interpretable compared to the Elastic Net approach used later.

```
# Logistic Regression
m1 <- train(income ~ ., data = train_data_factor, method = "glm", family =
"binomial", metric = "ROC", trControl = ctrl)

p1 <- predict(m1, test_data_factor)
p1_prob <- predict(m1, test_data_factor, type = "prob")$high
results[["Logistic Regression"]] <- evaluate_model(p1, p1_prob,
test_data_factor$income, "Logistic Regression")

# Random Forest
m2 <- train(income ~ ., data = train_data_factor, method = "rf", metric =
"ROC", trControl = ctrl)
p2 <- predict(m2, test_data_factor)
p2_prob <- predict(m2, test_data_factor, type = "prob")$high
results[["Random Forest"]] <- evaluate_model(p2, p2_prob,
test_data_factor$income, "Random Forest")

# XGBoost
m3 <- train(income ~ ., data = train_data_factor, method = "xgbTree", metric
= "ROC", trControl = ctrl)

p3 <- predict(m3, test_data_factor)
p3_prob <- predict(m3, test_data_factor, type = "prob")$high
results[["XGBoost"]] <- evaluate_model(p3, p3_prob, test_data_factor$income,
"XGBoost")

# GBM
```

```r
m4 <- train(income ~ ., data = train_data_factor, method = "gbm", metric =
"ROC", trControl = ctrl, verbose = FALSE)

p4 <- predict(m4, test_data_factor)
p4_prob <- predict(m4, test_data_factor, type = "prob")$high
results[["GBM"]] <- evaluate_model(p4, p4_prob, test_data_factor$income,
"GBM")

# Neural Net
m5 <- train(income ~ ., data = train_data_factor, method = "nnet", metric =
"ROC", trControl = ctrl, trace = FALSE)
p5 <- predict(m5, test_data_factor)
p5_prob <- predict(m5, test_data_factor, type = "prob")$high
results[["Neural Net"]] <- evaluate_model(p5, p5_prob,
test_data_factor$income, "Neural Net")
```

2.3 Superior Model (GLMNET - Elastic Net Regression)

At this point in the analysis, it became clear that the penalized Elastic Net Regression model from the glmnet package delivered superior results, especially in terms of AUC. Interestingly, its performance surpassed that of the same model wrapped inside the caret framework.

I must admit, I don’t fully understand the underlying reasons. My data science expertise is still evolving. However, this observation aligns with a pattern I noticed during my earlier Movie Rating project. The implementation below uses cross-validation to fine-tune the model and determine the best classification threshold based on F1 scores.

```r
# GLMNET Tuned Elastic Net Regression
set.seed(123)
glmnet_cv <- cv.glmnet(train_x, as.numeric(train_data_numeric$income), family
= "binomial", type.measure = "auc", alpha = 0.5)
pred_prob <- predict(glmnet_cv, newx = test_x, s = "lambda.min", type =
"response")
thresh_seq <- seq(0.1, 0.9, by = 0.05)
metrics <- sapply(thresh_seq, function(t) {
  pred_class <- ifelse(pred_prob >= t, 1, 0)
  cm <- confusionMatrix(factor(pred_class, levels = c(0, 1)),
factor(test_data_numeric$income, levels = c(0, 1)))
  TP <- cm$table[2, 2]; FP <- cm$table[2, 1]; FN <- cm$table[1, 2]
  precision <- ifelse((TP + FP) == 0, NA, TP / (TP + FP))
  recall <- ifelse((TP + FN) == 0, NA, TP / (TP + FN))
  f1 <- ifelse(is.na(precision) | is.na(recall) | (precision + recall == 0),
NA,
               2 * precision * recall / (precision + recall))
  return(f1)
})
best_thresh <- thresh_seq[which.max(metrics)]
```

```
final_pred <- ifelse(pred_prob >= best_thresh, 1, 0)
roc_glmnet <- roc(test_data_numeric$income, pred_prob)

logloss_glmnet <- logLoss(test_data_numeric$income, pred_prob)
cm_glmnet <- confusionMatrix(factor(final_pred, levels = c(0, 1)),
factor(test_data_numeric$income, levels = c(0, 1)))
TP <- cm_glmnet$table[2,2]; FP <- cm_glmnet$table[2,1]; FN <-
cm_glmnet$table[1,2]
precision <- ifelse((TP + FP) == 0, NA, TP / (TP + FP))
recall <- ifelse((TP + FN) == 0, NA, TP / (TP + FN))
f1 <- ifelse(is.na(precision) | is.na(recall) | (precision + recall == 0),
NA, 2 * precision * recall / (precision + recall))
results[["GLMNET Tuned"]] <- data.frame(Model = "GLMNET Tuned", AUC =
as.numeric(auc(roc_glmnet)),
                                        F1 = f1,
                                        Accuracy =
cm_glmnet$overall["Accuracy"],
                                        Sensitivity =
cm_glmnet$byClass["Sensitivity"],
                                        Specificity =
cm_glmnet$byClass["Specificity"],
                                        Precision = precision,
                                        LogLoss = logloss_glmnet)
```

3. Results

3.1 Initial Results

The analysis results are visualized using a heat map that provides a side-by-side comparison of all models applied to the Adult Income dataset. Evaluation metrics include AUC, F1 score, accuracy, sensitivity, specificity, precision, and log loss.

From an AUC standpoint, the Elastic Net regression model stands out with a value of 0.91, suggesting a strong model fit. Overall, it excels at identifying individuals with income ≤50K (class 0), as evidenced by its high accuracy and sensitivity. However, it is less effective at detecting those with income >50K (class 1), as shown by its lower specificity.

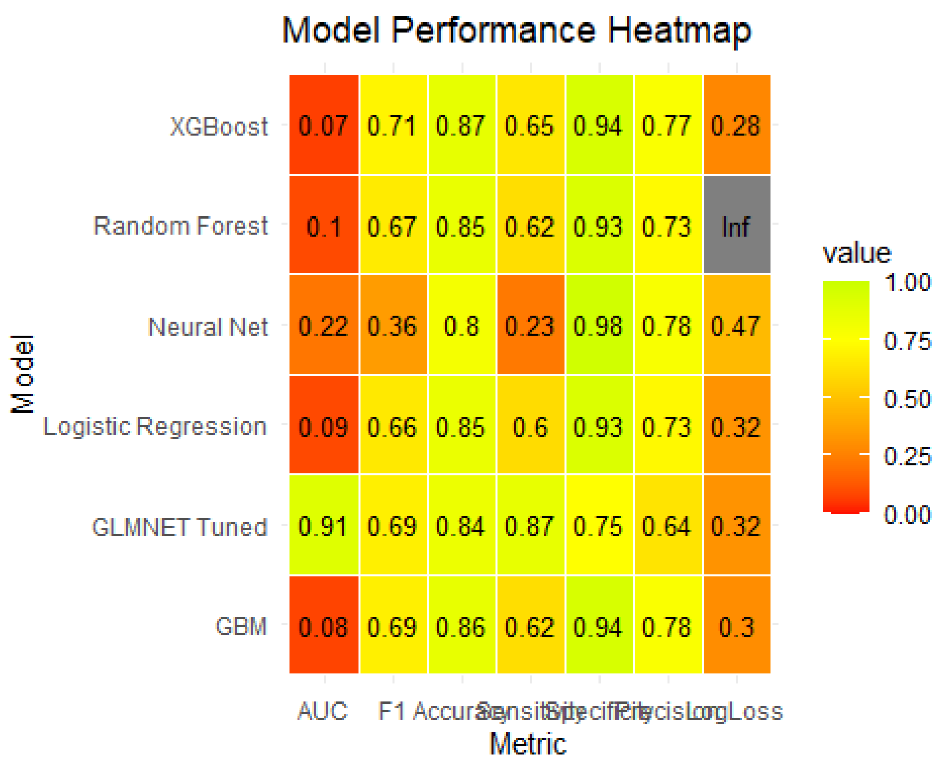
Summary of Key Metrics:

AUC (Area Under the ROC Curve): GLMNET Tuned outperforms all others with 0.91. Models like XGBoost, GBM, RF, and Logistic Regression fall below 0.10, indicating issues with ranking or probability calibration.

F1 Score: XGBoost achieved the highest F1 score (0.71), reflecting a good balance between precision and recall. Neural Net performed poorly at 0.36.

Log Loss: Lower is better. GBM (0.30), XGBoost (0.28), and Logistic Regression (0.32) are acceptable. Neural Net (0.47) and Random Forest (infinite) present concerns.

```r
# Combine and plot heatmap
final_results <- bind_rows(results)
final_results_rounded <- final_results %>% mutate(across(where(is.numeric),
round, 3))

print(final_results_rounded)

##                              Model   AUC    F1 Accuracy Sensitivity
Specificity
## Accuracy...1 Logistic Regression 0.094 0.660    0.850       0.605
0.928
## Accuracy...2       Random Forest 0.095 0.667    0.852       0.615
0.928
## Accuracy...3             XGBoost 0.072 0.708    0.870       0.652
0.940
## Accuracy...4                 GBM 0.081 0.687    0.865       0.616
0.944
## Accuracy...5          Neural Net 0.222 0.355    0.799       0.230
0.979
## Accuracy...6        GLMNET Tuned 0.906 0.689    0.838       0.866
0.748
##              Precision LogLoss
## Accuracy...1     0.726   0.322
## Accuracy...2     0.730     Inf
## Accuracy...3     0.774   0.280
## Accuracy...4     0.776   0.298
## Accuracy...5     0.780   0.471
## Accuracy...6     0.639   0.322

# Heatmap
heat_data <- melt(final_results_rounded, id.vars = "Model")
ggplot(heat_data, aes(x = variable, y = Model, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "red", high = "green", mid = "yellow", midpoint
= 0.75, limit = c(0, 1), space = "Lab") +
  geom_text(aes(label = round(value, 2)), color = "black", size = 3.5) +
  theme_minimal() +
  labs(title = "Model Performance Heatmap", x = "Metric", y = "Model")
```



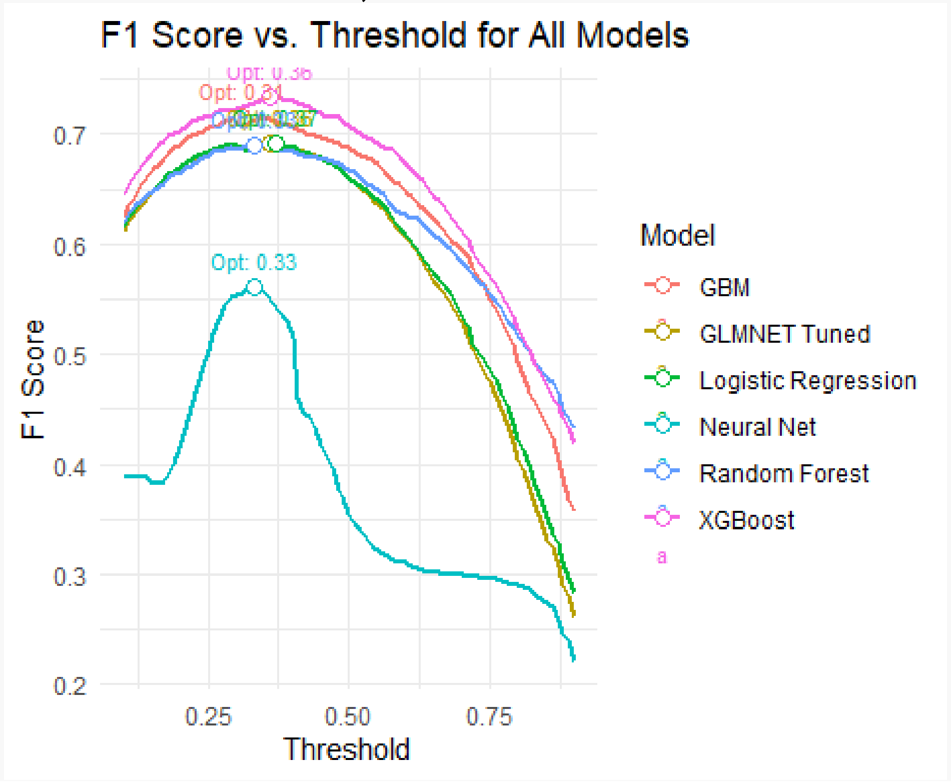
Model	AUC	F1	Accuracy	Sensitivity	Specificity	Precision	LogLoss
XGBoost	0.07	0.71	0.87	0.65	0.94	0.77	0.28
Random Forest	0.1	0.67	0.85	0.62	0.93	0.73	Inf
Neural Net	0.22	0.36	0.8	0.23	0.98	0.78	0.47
Logistic Regression	0.09	0.66	0.85	0.6	0.93	0.73	0.32
GLMNET Tuned	0.91	0.69	0.84	0.87	0.75	0.64	0.32
GBM	0.08	0.69	0.86	0.62	0.94	0.78	0.3

# 3.2 Further Results Analysis

While the heat map effectively summarizes model performance, deeper insights can be gained by plotting F1 scores across various classification thresholds. The following chart illustrates how the F1 score changes with the threshold for each model. Interestingly, the optimal thresholds cluster closely together, reinforcing the consistency of this analysis.

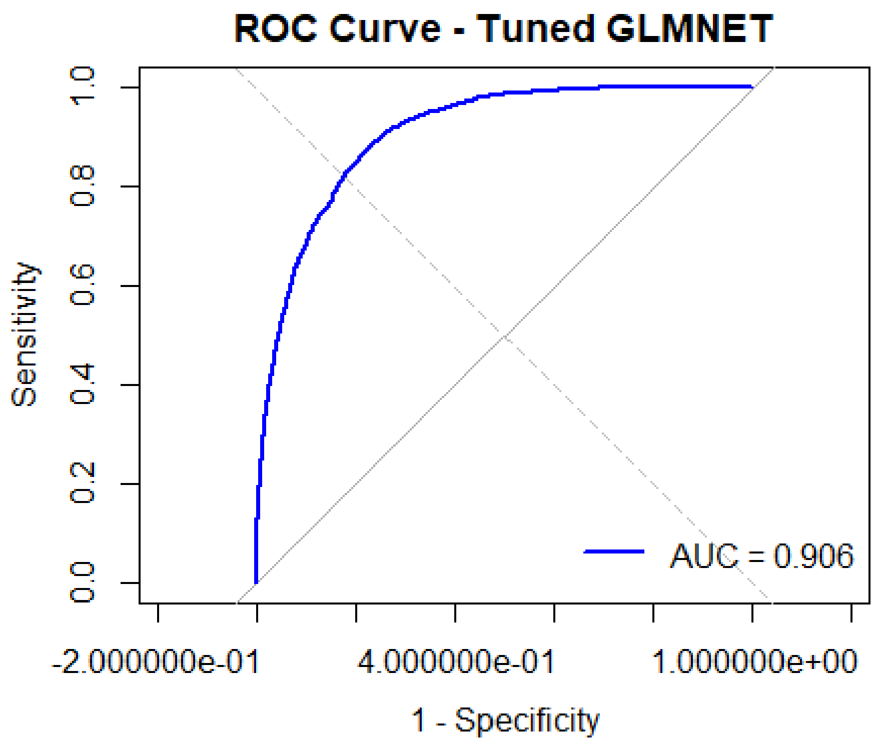
```
# Define thresholds to evaluate
thresholds <- seq(0.1, 0.9, by = 0.01)

# Helper function to compute F1 score at different thresholds
compute_f1_curve <- function(probs, true_labels, model_name) {
  sapply(thresholds, function(t) {
    preds <- ifelse(probs >= t, "high", "low") %>% factor(levels = c("low",
"high"))
    cm <- confusionMatrix(preds, true_labels, positive = "high")
    TP <- cm$table[2, 2]; FP <- cm$table[2, 1]; FN <- cm$table[1, 2]
    precision <- ifelse((TP + FP) == 0, NA, TP / (TP + FP))
    recall <- ifelse((TP + FN) == 0, NA, TP / (TP + FN))
    if (is.na(precision) | is.na(recall) | (precision + recall == 0))
return(NA)
    return(2 * precision * recall / (precision + recall))
  }) %>%
    data.frame(threshold = thresholds, F1 = ., model = model_name)
}

# Compute F1 score curves for each model
```

```r
f1_curves <- bind_rows(
  compute_f1_curve(p1_prob, test_y, "Logistic Regression"),
  compute_f1_curve(p2_prob, test_y, "Random Forest"),
  compute_f1_curve(p3_prob, test_y, "XGBoost"),
  compute_f1_curve(p4_prob, test_y, "GBM"),
  compute_f1_curve(p5_prob, test_y, "Neural Net"),
  compute_f1_curve(as.vector(pred_prob), factor(test_data_numeric$income,
levels = c(0, 1), labels = c("low", "high")), "GLMNET Tuned")
)

# Find optimal threshold points
opt_f1 <- f1_curves %>%
  group_by(model) %>%
  slice_max(F1, with_ties = FALSE)

# Plot
ggplot(f1_curves, aes(x = threshold, y = F1, color = model)) +
  geom_line(size = 1) +
  geom_point(data = opt_f1, aes(x = threshold, y = F1), size = 3, shape = 21,
fill = "white") +
  geom_text(data = opt_f1, aes(label = paste0("Opt: ", round(threshold, 2))),
vjust = -1, size = 3) +
  theme_minimal() +
  labs(title = "F1 Score vs. Threshold for All Models",
       x = "Threshold",
       y = "F1 Score",
       color = "Model")
```



The final curve in this series is an ROC curve for the glmnet Elastic Regression. The curve appears well above the guessing line further highlighting one more time the strong performance of this model.

```
# Predict probabilities from tuned GLMNET model
pred_prob_glmnet <- predict(glmnet_cv, newx = test_x, s = "lambda.min", type
= "response")

# Compute ROC
roc_glmnet <- roc(test_data_numeric$income, pred_prob_glmnet)

# Plot ROC
plot(roc_glmnet, col = "blue", lwd = 2,
     main = "ROC Curve - Tuned GLMNET",
     legacy.axes = TRUE)
abline(a = 0, b = 1, lty = 2, col = "gray")

# Add AUC text
auc_val <- auc(roc_glmnet)
legend("bottomright", legend = paste0("AUC = ", round(auc_val, 3)),
       col = "blue", lwd = 2, bty = "n")
```



4. Conclusion

The best overall model in this analysis was the tuned GLMNET model, which demonstrated strong and balanced performance across multiple metrics. It achieved an AUC of 0.91, alongside favorable scores in accuracy, F1, and log loss. These attributes are particularly valuable in situations involving class imbalance, as is the case here where income levels are not evenly distributed. A significant majority of individuals fall into the “less than $50,000” income category.

In contrast, the Neural Network model was relatively ineffective. Despite showing reasonable precision, it suffered from poor recall (0.23) and a high log loss, making it unreliable for practical classification.

Models such as XGBoost and GBM recorded high F1 scores, yet their AUC values were surprisingly low. This suggests that while they perform well at a specific classification threshold, they struggle with probability ranking, which is critical for nuanced decision-making.

One particularly noteworthy observation was the “Inf” (infinite) log loss value produced by the Random Forest model. This result appears to be a red flag, likely caused by predicted probabilities being exactly 0 or 1, which leads to undefined values when computing log(0) during loss calculation. Although the exact mechanics merit further investigation, the behavior indicates a potential risk in relying on this model under the current configuration.

Reflections and Limitations It is important to emphasize that this inquiry was exploratory rather than exhaustive. I explored a wide range of modeling techniques, often following leads from online resources and tutorials. For example, the inclusion of the Neural Network model was inspired by articles such as Turing’s explanation of neural networks, even though it did not yield optimal results in this case.

Throughout the process, I discovered several nuanced differences in how models are implemented across packages. For instance, glmnet applies penalization by default, which differs from implementations in caret. These subtle distinctions can have a meaningful impact on model behavior and outcomes.

While the GLMNET model with an AUC of 0.91 emerged as the best-performing model in this exploration, I acknowledge that there may be better configurations or alternative models not yet tested. The goal was not to find a definitive solution, but rather to develop a deeper understanding of how different models behave in the context of income prediction.

5. Bibliography

Wickham, H., & Grolemund, G. (2023). R for data science (2nd ed.). Retrieved from https://r4ds.hadley.nz/

Hastie, T., & Tibshirani, R. (2014). Statistical learning with R. Stanford University. Retrieved from https://online.stanford.edu/courses/sohs-ystatslearning-statistical-learning

Leek, J., Peng, R. D., & Caffo, B. S. (2014). Data science specialization. Coursera. Retrieved from https://www.coursera.org/specializations/jhu-data-science

Brownlee, J. (2016). Machine learning mastery with R. Machine Learning Mastery. Retrieved from https://machinelearningmastery.com/machine-learning-with-r/

Kuhn, M., & Wickham, H. (2020). Tidymodels. Retrieved from https://www.tidymodels.org/

DataCamp. (n.d.). Data scientist with R. Retrieved from https://www.datacamp.com/tracks/data-scientist-with-r

R-bloggers. (n.d.). R-bloggers: R news and tutorials. Retrieved from https://www.r-bloggers.com/

Ismay, C., & Kim, A. Y. (2019). ModernDive: An introduction to statistical and data sciences via R. Retrieved from https://moderndive.com/