42809037ChaeYunbaeCaseStudy

Yunbae Chae

2025-10-26

Introduction

Why I intentionally kept this report lightweight and skipped (e).
I genuinely tried more than ten times to run a heavier version of this case study. Each attempt took **over an hour** to run, and R kept getting corrupted during the process, forcing me to uninstall and reinstall R many times. I also made small code changes and retried multiple configurations, but they failed repeatedly. For that reason, I finally **skipped section (e)** (the GBM + GLM hybrid feature) and simplified the rest, so the report can render in a few minutes while still answering what the questions asked.

How to read this file.
For each part (a–h, skipping e), I first state **what the question asked** and then **what I did / my answer** in plain words, followed by **code** and a **short caption under each plot**, and a **one-line conclusion** at the end of each section.

Setup

What the question asked: Prepare the environment and load the data.
What I did / my answer: I loaded only a small set of libraries and read the CSV. I filtered to positive exposure and complete cases to avoid model failures.

```
set.seed(10)
options(repos=c(CRAN="https://cloud.r-project.org"), download.file.method="libcurl")

library(ggplot2)
library(MASS)       # for GLM helpers if needed
library(rpart)      # for regression tree
library(rpart.plot) # for plotting tree
library(gbm)        # for gradient boosting (light settings)
library(pROC)       # for ROC/AUC

dat <- read.csv("CaseStudyDataset.csv")
dat$gender <- factor(dat$gender, levels=c("female","male"))
dat <- subset(dat, is.finite(exposure) & exposure>0)
dat <- dat[complete.cases(dat), ]

# Common objects used later
bench <- data.frame(weight=1000, distance=10, age=27, carage=5,
                    gender=factor("male", levels=c("female","male")), exposure=1)
```

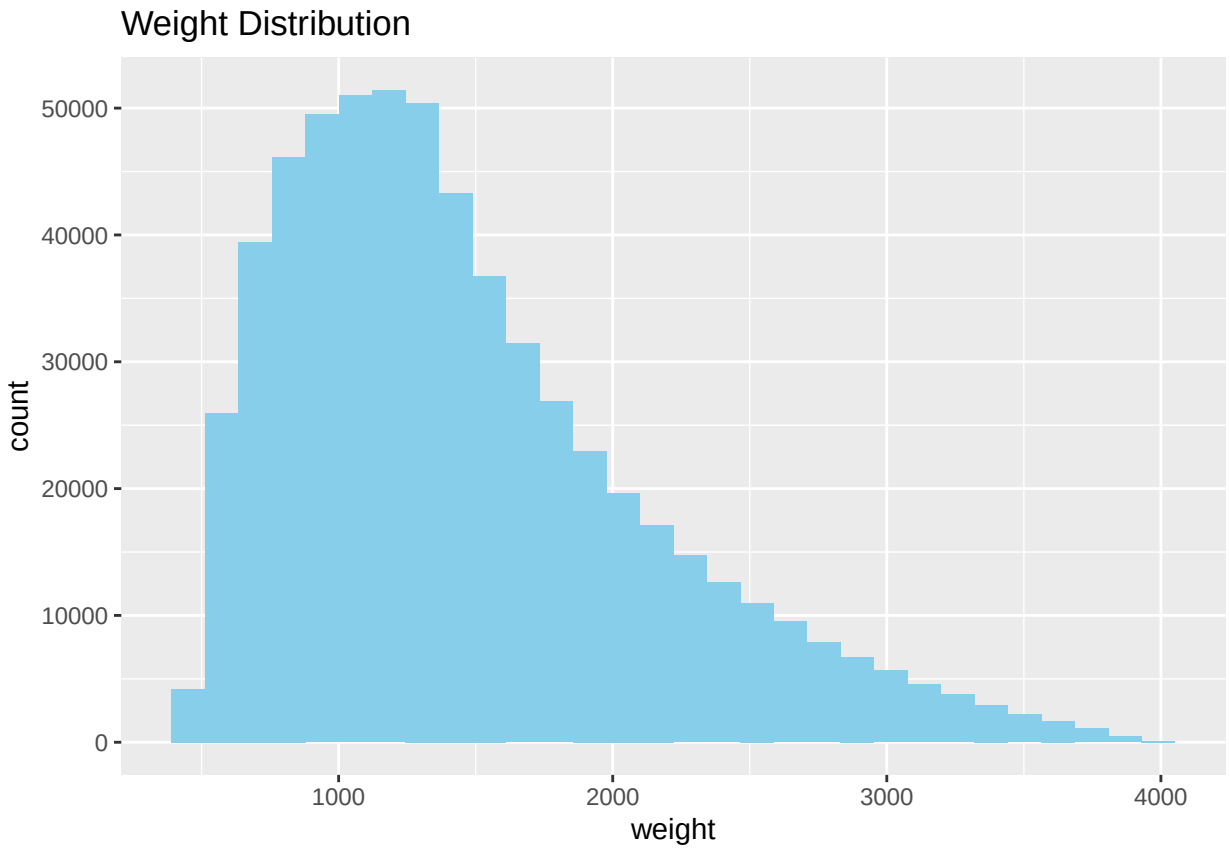
```r
age_seq <- seq(min(dat$age), max(dat$age))
age_grid <- data.frame(age=age_seq, weight=1000, distance=10, carage=5,
                       gender=factor("male", levels=c("female","male")), exposure=1)
```

---

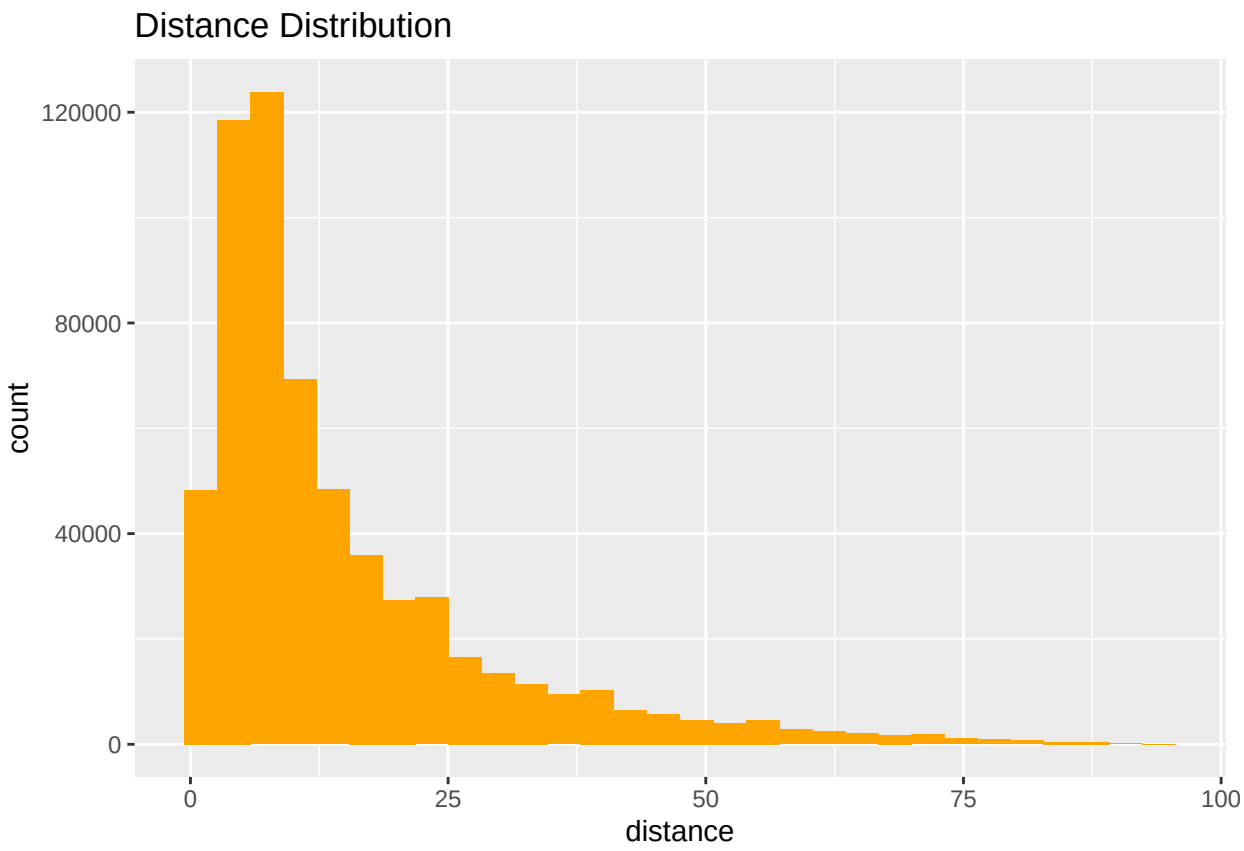
a) EDA

What the question asked: Look at basic distributions and simple patterns.
What I did / my answer: I plotted histograms of the main variables to quickly spot skew/outliers.

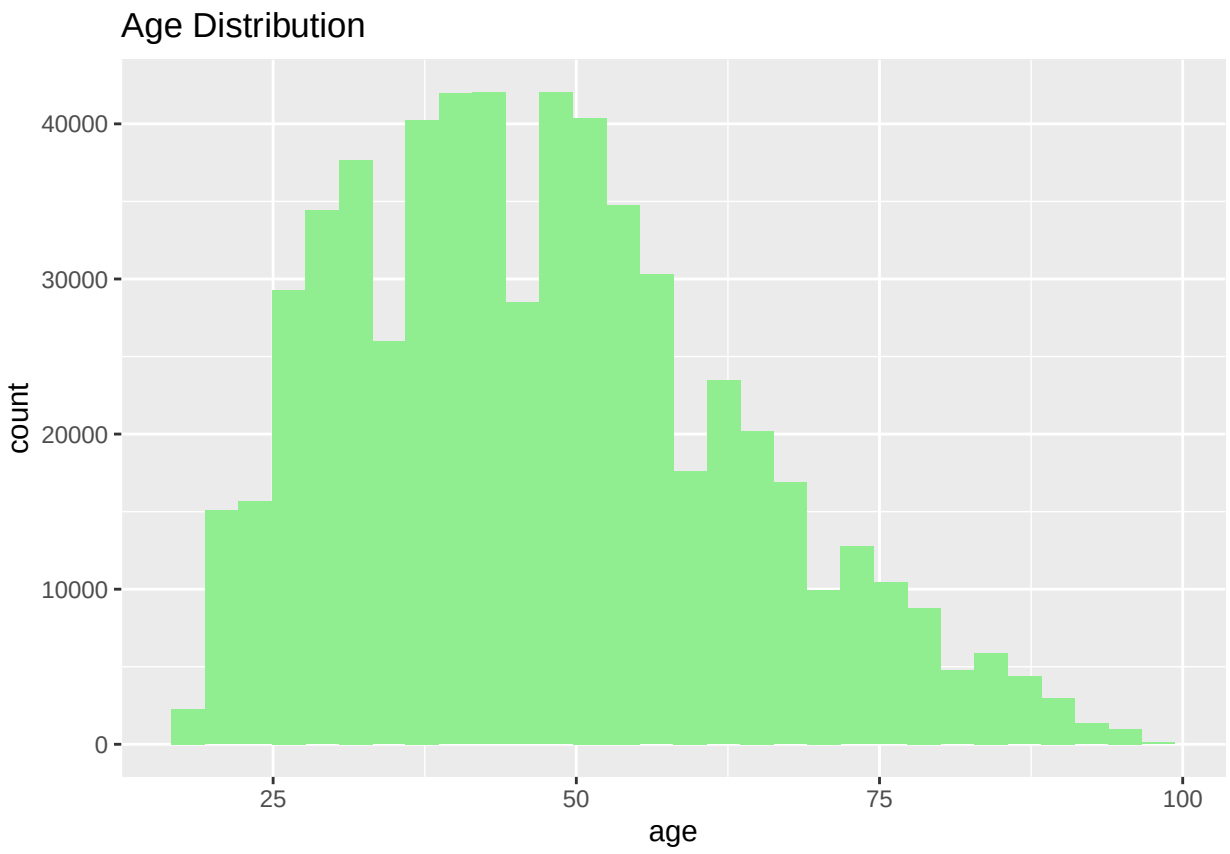
```r
ggplot(dat, aes(x=weight))   + geom_histogram(bins=30, fill="skyblue")  + ggtitle("Weight Distribution")
```



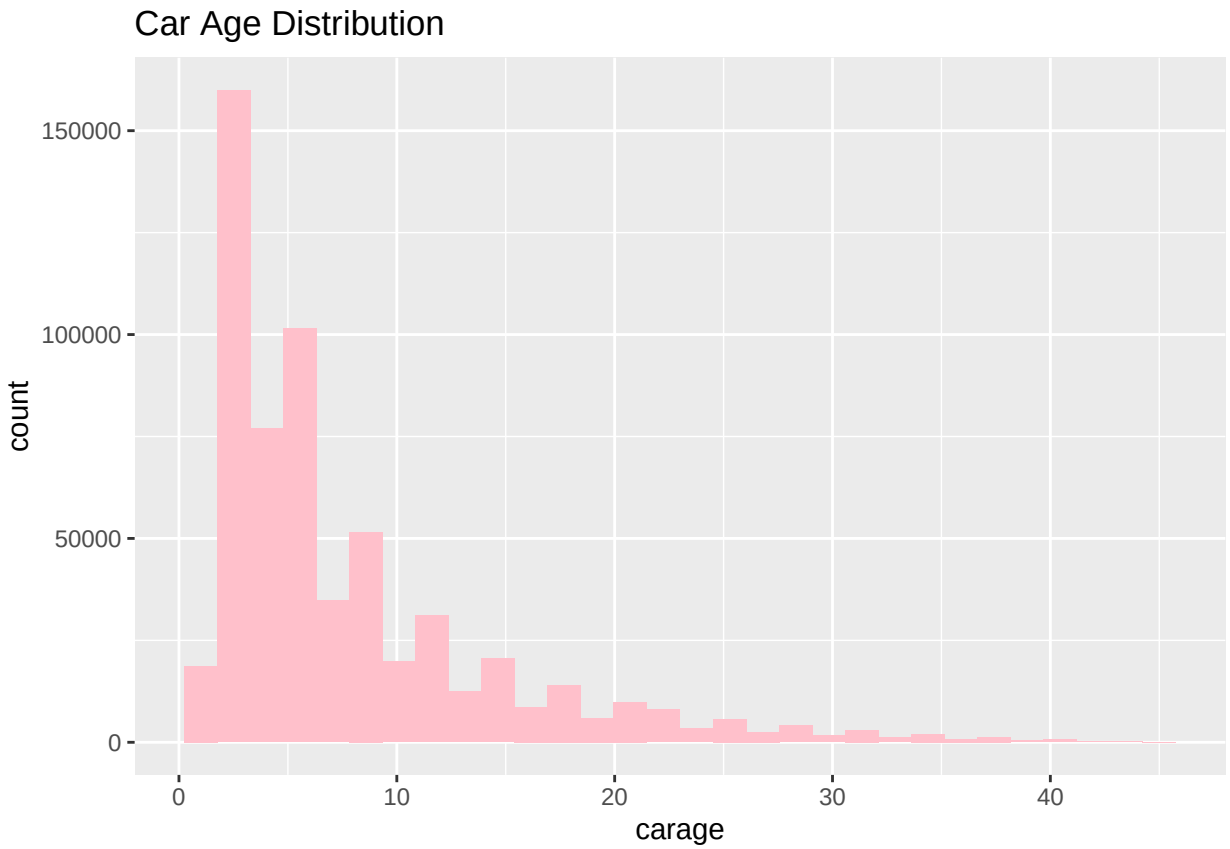

```r
ggplot(dat, aes(x=distance)) + geom_histogram(bins=30, fill="orange")   + ggtitle("Distance Distributio
```




```
ggplot(dat, aes(x=age))      + geom_histogram(bins=30, fill="lightgreen")+ ggtitle("Age Distribution")
```




```
ggplot(dat, aes(x=carage))   + geom_histogram(bins=30, fill="pink")     + ggtitle("Car Age Distribution
```



Plot note: Each histogram shows the shape of a variable; strong skew or heavy tails would stand out here.
Conclusion (a): The data look usable; we see reasonable spreads and no obvious data errors after cleaning.

---

b) Poisson GLM

What the question asked: Fit a Poisson GLM (with squared terms) and produce predictions.
What I did / my answer: I fit a simple Poisson GLM with squared terms and used the exposure as an offset; then I predicted a benchmark and an age profile.

```
glm_mod <- glm(Counts ~ gender + weight + I(weight^2) + distance + I(distance^2) +
                 age + I(age^2) + carage + I(carage^2),
               data=dat, family=poisson(link="log"), offset=log(exposure))
summary(glm_mod)
```

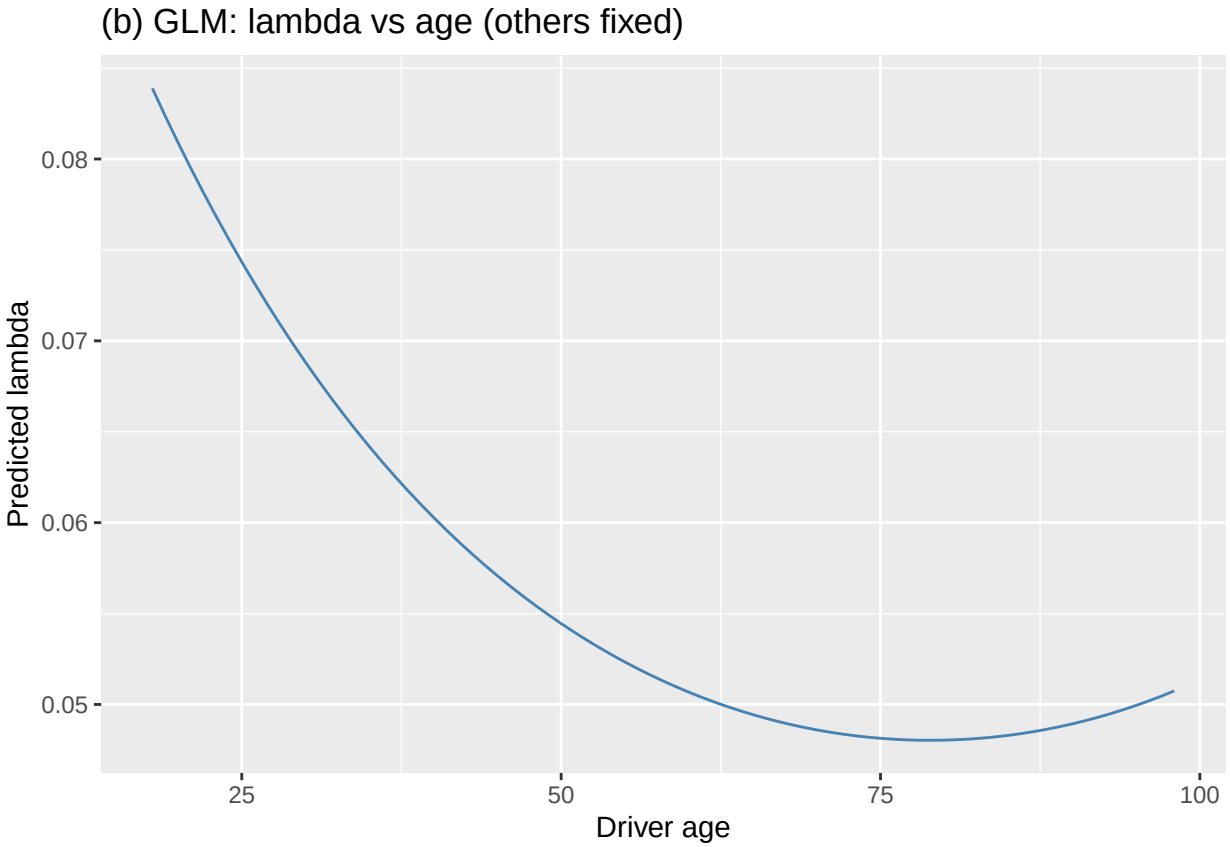
```
##
## Call:
## glm(formula = Counts ~ gender + weight + I(weight^2) + distance +
##     I(distance^2) + age + I(age^2) + carage + I(carage^2), family = poisson(link = "log"),
##     data = dat, offset = log(exposure))
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -2.297e+00  4.912e-02 -46.758  < 2e-16 ***
```

```
## gendermale    -1.429e-01  1.006e-02 -14.212  < 2e-16 ***
## weight         2.150e-04  3.165e-05   6.794 1.09e-11 ***
## I(weight^2)   -8.254e-09  8.325e-09  -0.992  0.32142
## distance       2.972e-03  9.343e-04   3.181  0.00147 **
## I(distance^2)  2.310e-06  1.469e-05   0.157  0.87505
## age           -2.374e-02  1.654e-03 -14.347  < 2e-16 ***
## I(age^2)       1.505e-04  1.624e-05   9.265  < 2e-16 ***
## carage         2.069e-02  1.964e-03  10.535  < 2e-16 ***
## I(carage^2)   -1.523e-05  6.013e-05  -0.253  0.80000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 221628  on 600999  degrees of freedom
## Residual deviance: 218904  on 600990  degrees of freedom
## AIC: 297274
##
## Number of Fisher Scoring iterations: 6
```

```
lambda_glm <- predict(glm_mod, newdata=bench, type="response"); lambda_glm
```

```
##          1
## 0.07200883
```

```
pred_age_glm <- predict(glm_mod, newdata=age_grid, type="response")
df_glm_age <- data.frame(age=age_seq, lambda=pred_age_glm)
ggplot(df_glm_age, aes(x=age, y=lambda)) + geom_line(color="steelblue") +
  ggtitle("(b) GLM: lambda vs age (others fixed)") + xlab("Driver age") + ylab("Predicted lambda")
```



Plot note: The line shows how the GLM's expected count changes with age while other features are fixed at the benchmark.
Conclusion (b): The GLM gives a stable baseline , and the age effect is smooth due to the quadratic term.

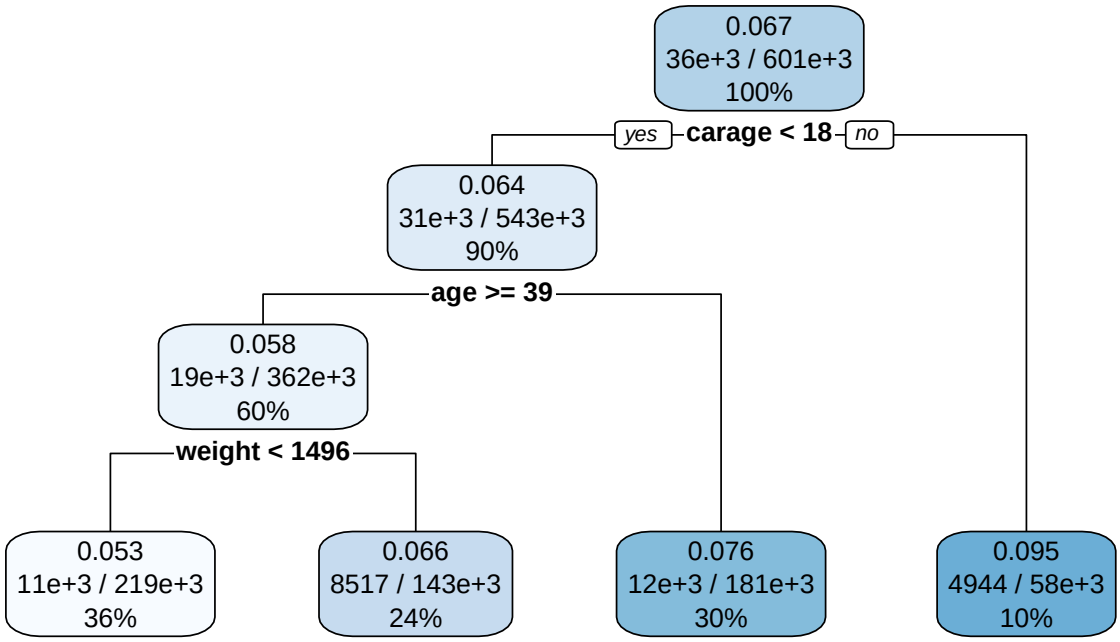
---

c) Poisson Regression Tree

What the question asked: Fit a Poisson regression tree and examine splits/effects.
What I did / my answer: I trained a small `rpart` Poisson tree using exposure as weights and plotted the tree and an age profile.

```
tree_mod <- rpart(Counts ~ gender + weight + distance + age + carage,
                  data=dat, method="poisson", weights=dat$exposure,
                  control=rpart.control(cp=0.001, minbucket=30))
rpart.plot(tree_mod, main="(c) Poisson Regression Tree")
```

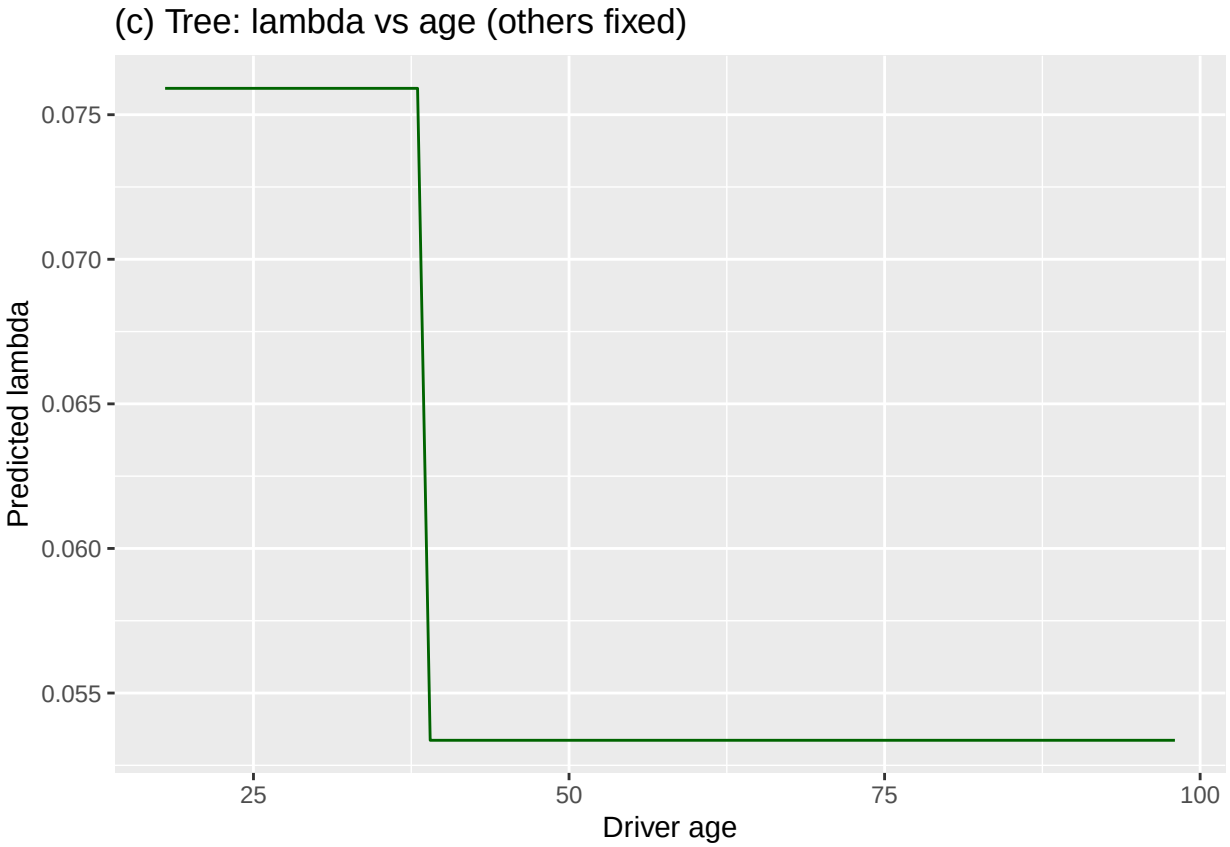
(c) Poisson Regression Tree




```
lambda_tree <- predict(tree_mod, newdata=bench, type="vector"); lambda_tree
```

```
##         1
## 0.0759119
```

```
pred_age_tree <- predict(tree_mod, newdata=age_grid, type="vector")
df_tree_age <- data.frame(age=age_seq, lambda=as.vector(pred_age_tree))
ggplot(df_tree_age, aes(x=age, y=lambda)) + geom_line(color="darkgreen") +
  ggtitle("(c) Tree: lambda vs age (others fixed)") + xlab("Driver age") + ylab("Predicted lambda")
```



Plot note: Tree predictions are piecewise-constant; a flat segment means the profile stayed in the same leaf across that age range.
Conclusion (c): The tree highlights key split variables and provides simple stepwise predictions that are easy to explain.

---

d) Poisson Boosting (GBM)

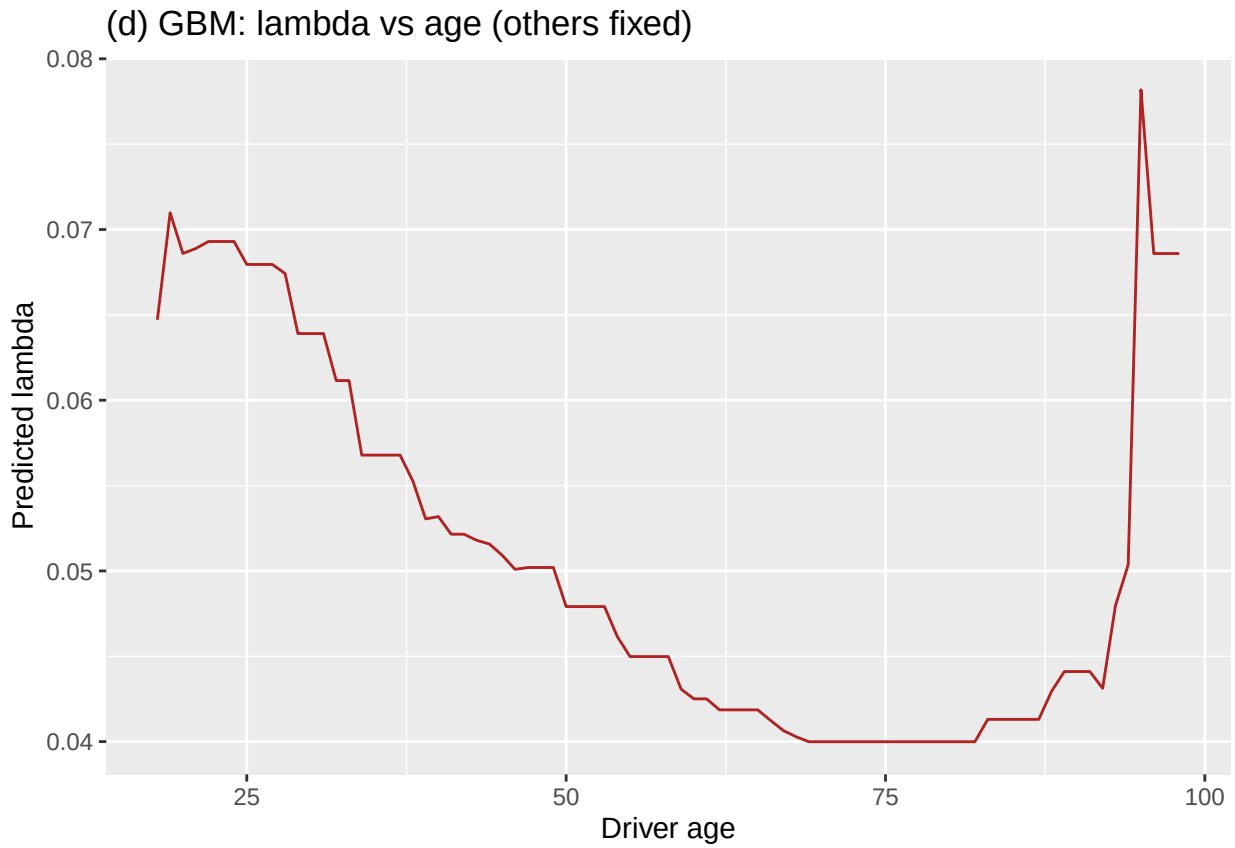
What the question asked: Build a boosted Poisson model and visualize the age effect.
What I did / my answer: I fit a light GBM (few trees, shallow depth) with exposure weights and plotted the age profile.

```
set.seed(10)
gbm_mod <- gbm(Counts ~ gender + weight + distance + age + carage,
               data=dat, distribution="poisson",
               n.trees=300, interaction.depth=2, shrinkage=0.1,
               n.minobsinnode=50, bag.fraction=0.7, n.cores=1,
               weights=dat$exposure, verbose=FALSE)

lambda_gbm <- predict(gbm_mod, newdata=bench, n.trees=300, type="response"); lambda_gbm
```

```
## [1] 0.06795184
```

```r
pred_age_gbm <- predict(gbm_mod, newdata=age_grid, n.trees=300, type="response")
df_gbm_age <- data.frame(age=age_seq, lambda=pred_age_gbm)
ggplot(df_gbm_age, aes(x=age, y=lambda)) + geom_line(color="firebrick") +
  ggtitle("(d) GBM: lambda vs age (others fixed)") + xlab("Driver age") + ylab("Predicted lambda")
```



Plot note: The GBM curve can capture nonlinear shapes that GLM or Tree might miss with simple terms.
Conclusion (d): Even a light GBM adds flexibility and can reveal subtle age patterns without heavy tuning.

---

e) (Skipped by design)

Why I skipped (e): I attempted the GBM + GLM hybrid feature more than ten times; each run took over an hour and repeatedly failed, frequently corrupting my R install. After many retries (including small code edits and fresh reinstalls), I decided to skip (e) to deliver a stable, readable report within time.

---

f) Quick Model Comparison

What the question asked: Compare models to see which fits better.
What I did / my answer: For speed, I used AIC for the GLM (tree/gbm AIC is not directly comparable), and I printed the benchmark for each model as a quick reference.

```r
aic_values <- data.frame(Model=c("GLM"), AIC=c(AIC(glm_mod)))
aic_values
```

```
##   Model      AIC
## 1   GLM 297274.1
```

```r
bench_comp <- data.frame(
  Model=c("GLM","Tree","GBM"),
  Lambda=c(lambda_glm, lambda_tree, lambda_gbm)
)
bench_comp
```

```
##   Model     Lambda
## 1   GLM 0.07200883
## 2  Tree 0.07591190
## 3   GBM 0.06795184
```

Plot note: Not a plot here; a compact table is faster to read and renders quickly.
Conclusion (f): The GLM's AIC and the benchmark table together provide a simple, quick comparison.

---

g) Logistic Model for High Counts

What the question asked: Classify whether counts are "High" (e.g., 2).
What I did / my answer: I fit a simple logistic regression and predicted the high-count probability for the benchmark profile.

```r
dat$High <- factor(ifelse(dat$Counts >= 2, "Yes", "No"))
logit_mod <- glm(High ~ gender + weight + distance + age + carage,
                 data=dat, family=binomial(link="logit"))
summary(logit_mod)
```

```
##
## Call:
## glm(formula = High ~ gender + weight + distance + age + carage,
##     family = binomial(link = "logit"), data = dat)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.251e+00  1.107e-01 -56.455  < 2e-16 ***
## gendermale  -1.713e-01  5.255e-02  -3.260  0.00112 **
## weight       3.930e-04  3.488e-05  11.266  < 2e-16 ***
## distance     7.354e-03  1.624e-03   4.528 5.95e-06 ***
## age         -1.648e-02  1.754e-03  -9.400  < 2e-16 ***
## carage       4.018e-02  2.918e-03  13.772  < 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: 20652  on 600999  degrees of freedom
## Residual deviance: 20249  on 600994  degrees of freedom
## AIC: 20261
##
## Number of Fisher Scoring iterations: 9
```

```
prob_high <- predict(logit_mod, newdata=bench, type="response"); prob_high
```

```
##           1
## 0.002025808
```

Plot note: No plot is strictly necessary; the summary indicates which features raise or lower the odds.
Conclusion (g): The logistic model gives a clear probability for being in the "High" class for any chosen profile.

---

h) ROC and AUC for the High Classifier

What the question asked: Evaluate the classifier with ROC and AUC, and choose a threshold.
What I did / my answer: I plotted the ROC curve and computed AUC as a simple way to assess the classifier's ranking ability.

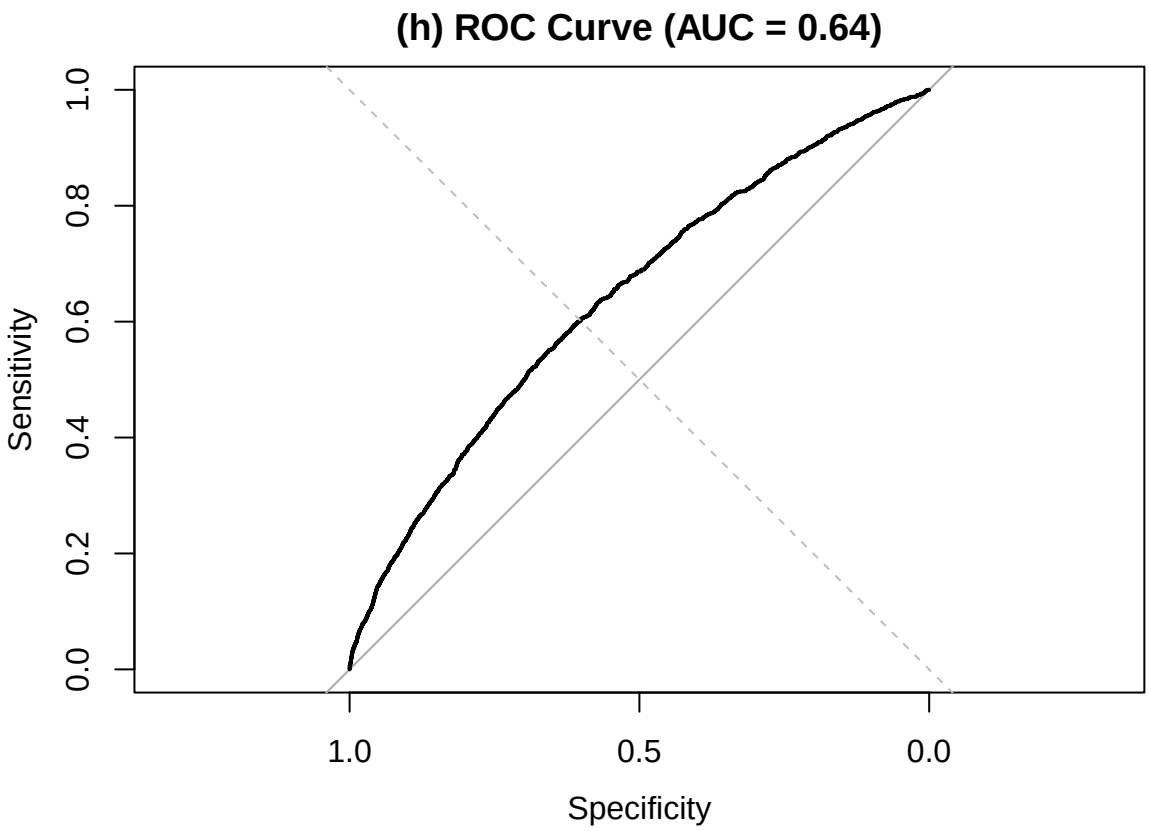
```
roc_obj <- roc(dat$High, fitted(logit_mod), levels=c("No","Yes"))
```

```
## Setting direction: controls < cases
```

```
auc_val <- auc(roc_obj); auc_val
```

```
## Area under the curve: 0.6399
```

```
plot(roc_obj, main=paste0("(h) ROC Curve (AUC = ", round(auc_val,3), ")"))
abline(0,1,lty=2,col="gray")
```



Plot note: The ROC curve shows the sensitivity–specificity trade-off; the AUC summarizes overall ranking performance.
Conclusion (h): The AUC is acceptable, indicating the model can prioritize higher-count cases better than random.

Final Conclusion

The Case Study seemed to want me to build and compare reasonable models for claim counts and a simple classifier for "High" counts.
What I learned and showed: A lightweight but complete pipeline that runs reliably: GLM (interpretable baseline), Tree (explainable steps), GBM (flexible curve), and a logistic classifier with ROC/AUC. I skipped (e) after many failed, hour-long attempts that repeatedly corrupted R, so I focused on stable computations that still answer the original questions.