15.072 Advanced Analytics Edge Fall 2025

DELIVERABLE 1

Authors

Hindy Rossignol, Riya Parikh,

Mrugank Pednekar, Ioannis Panagiotopoulos

Question 1

Question1. (A)

We first get an insight into the data by looking at our base model including all the features, looking at the corelation matrix, OOS R^2, and identifying which features are significant, highly corelated, etc.

```
library(tidyverse)
library(dplyr)

train <- read.csv("laptop_train.csv")
test  <- read.csv("laptop_test.csv")

train$Company  <- factor(train$Company)
train$TypeName <- factor(train$TypeName)
train$GPU      <- factor(train$GPU)

# These should stay numeric
num_vars <- c("Screen","Memory","Weight","Rating","Price")
train[num_vars] <- lapply(train[num_vars], as.numeric)

# correlation matrix
cor(train[, num_vars], use = "complete.obs")
```

```
##              Screen     Memory      Weight      Rating      Price
## Screen   1.00000000 0.09940796  0.81150314 -0.03097351 -0.1306660
## Memory   0.09940796 1.00000000  0.29239722  0.02431015  0.7224164
## Weight   0.81150314 0.29239722  1.00000000 -0.01924711  0.1167789
## Rating  -0.03097351 0.02431015 -0.01924711  1.00000000 -0.0290045
## Price   -0.13066597 0.72241635  0.11677891 -0.02900450  1.0000000
```

```
# Base Model
model1 <- lm(Price ~ InventoryID + Screen + Memory + Weight + Rating + Company + TypeName + GPU, data =
summary(model1)
```

```
##
## Call:
## lm(formula = Price ~ InventoryID + Screen + Memory + Weight +
##     Rating + Company + TypeName + GPU, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -975.88 -188.34  -35.98  161.98 1523.80
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)        1421.95846  289.06648   4.919 1.10e-06 ***
## InventoryID           0.06703    0.08057   0.832  0.40572
## Screen             -120.38632   21.81619  -5.518 4.94e-08 ***
## Memory               87.25133    4.40165  19.822  < 2e-16 ***
```

```
## Weight             270.53619   49.45684   5.470 6.41e-08 ***
## Rating             -11.66999    4.61720  -2.528  0.01172 *
## CompanyDell         77.66116   44.30153   1.753  0.08007 .
## CompanyHP          147.04460   51.77867   2.840  0.00465 **
## CompanyLenovo       -1.75199   61.33453  -0.029  0.97722
## TypeNameNotebook   -81.93259   59.12440  -1.386  0.16629
## TypeNameUltrabook  405.48355   76.00576   5.335 1.32e-07 ***
## GPUIntel           164.06650   39.70629   4.132 4.06e-05 ***
## GPUNvidia          217.76478   46.46989   4.686 3.39e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 337.7 on 652 degrees of freedom
## Multiple R-squared:  0.6601, Adjusted R-squared:  0.6538
## F-statistic: 105.5 on 12 and 652 DF,  p-value: < 2.2e-16
```

```r
library(olsrr)
ols_step_backward_p(model1, p_val = 0.05, progress = TRUE)
```

```
## Backward Elimination Method
## ---------------------------
##
## Candidate Terms:
##
## 1. InventoryID
## 2. Screen
## 3. Memory
## 4. Weight
## 5. Rating
## 6. Company
## 7. TypeName
## 8. GPU
##
##
## Variables Removed:
##
## => InventoryID
##
## No more variables to be removed.

##
##
##                                   Stepwise Summary
## -----------------------------------------------------------------------------
## Step    Variable         AIC         SBC         SBIC        R2       Adj. R2
## -----------------------------------------------------------------------------
##  0      Full Model     9645.396    9708.393    7750.566    0.66009    0.65383
##  1      InventoryID    9644.102    9702.599    7749.215    0.65973    0.65399
## -----------------------------------------------------------------------------
##
## Final Model Output
## ------------------
##
##                               Model Summary
## --------------------------------------------------------------------
```

```
## R                           0.812       RMSE                 334.527
## R-Squared                   0.660       MSE               111908.165
## Adj. R-Squared              0.654       Coef. Var             32.882
## Pred R-Squared              0.644       AIC                 9644.102
## MAE                       247.405       SBC                 9702.599
## -----------------------------------------------------------------
##  RMSE: Root Mean Square Error
##  MSE: Mean Square Error
##  MAE: Mean Absolute Error
##  AIC: Akaike Information Criteria
##  SBC: Schwarz Bayesian Criteria
##
##                                    ANOVA
## ---------------------------------------------------------------------------
##                      Sum of
##                     Squares          DF     Mean Square       F          Sig.
## ---------------------------------------------------------------------------
## Regression    144284374.791          11    13116761.345    115.095    0.0000
## Residual       74418929.460         653      113964.670
## Total         218703304.251         664
## ---------------------------------------------------------------------------
##
##                                         Parameter Estimates
## ----------------------------------------------------------------------------------------------------
##              model        Beta    Std. Error    Std. Beta       t         Sig        lower       upper
## ----------------------------------------------------------------------------------------------------
##        (Intercept)    1434.247       288.621                  4.969     0.000      867.510    2000.984
##             Screen    -120.932        21.801       -0.249    -5.547     0.000     -163.741     -78.123
##             Memory      87.373         4.398        0.574    19.866     0.000       78.736      96.009
##             Weight     271.099        49.441        0.287     5.483     0.000      174.017     368.180
##             Rating     -11.821         4.613       -0.059    -2.563     0.011      -20.878      -2.763
##        CompanyDell      86.725        42.931        0.069     2.020     0.044        2.426     171.025
##          CompanyHP     170.395        43.502        0.131     3.917     0.000       84.974     255.816
##      CompanyLenovo      35.328        42.129        0.028     0.839     0.402      -47.396     118.052
##   TypeNameNotebook     -76.295        58.721       -0.061    -1.299     0.194     -191.600      39.009
## TypeNameUltrabook     416.394        74.848        0.265     5.563     0.000      269.421     563.366
##           GPUIntel     165.430        39.663        0.144     4.171     0.000       87.547     243.312
##          GPUNvidia     218.139        46.457        0.173     4.696     0.000      126.916     309.361
## ----------------------------------------------------------------------------------------------------
```

```r
# out-of-sample R² for base model
pred_test <- predict(model1, newdata = test)
sse <- sum((test$Price - pred_test)^2)
sst <- sum((test$Price - mean(test$Price))^2)
R2_out <- 1 - sse/sst
R2_out
```

```
## [1] 0.5506981
```

Looking at the correlation matrix, I observed that Screen and Weight are highly correlated (0.81). This high collinearity inflates variances of coefficient estimates and makes interpretation unstable. Indeed, in the full regression output, both Screen and Weight appeared significant, but their strong correlation makes it difficult to separate their individual contributions. To avoid redundancy and multicollinearity, I decided to drop Weight and retain Screen, which has a closer correlation to our dependent variable (price), low p-value (statistically significant), and which is more directly interpretable as a feature consumers see when buying laptops. Next, I examined the coefficient for InventoryID. This variable is simply an internal stock identifier and has no managerial meaning for pricing. Its coefficient was very small (0.067), with a p-value of 0.406, confirming it was not statistically significant. Including InventoryID risks overfitting without adding explanatory value. Therefore, I chose to exclude InventoryID from the second model.

```r
# 2nd Model to compare with
model2 <- lm(Price ~ Screen + Memory + Rating + Company + TypeName + GPU, data = train)
summary(model2)
```

```
##
## Call:
## lm(formula = Price ~ Screen + Memory + Rating + Company + TypeName +
##     GPU, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1040.73  -194.38   -36.05   166.39  1559.87
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        823.027    272.079   3.025  0.00258 **
## Screen             -36.598     15.791  -2.318  0.02077 *
## Memory              90.206      4.464  20.208  < 2e-16 ***
## Rating             -12.815      4.710  -2.721  0.00669 **
## CompanyDell        124.905     43.294   2.885  0.00404 **
## CompanyHP          175.313     44.449   3.944 8.87e-05 ***
## CompanyLenovo       60.894     42.790   1.423  0.15519
## TypeNameNotebook  -234.455     52.273  -4.485 8.60e-06 ***
## TypeNameUltrabook  203.684     65.418   3.114  0.00193 **
## GPUIntel           163.638     40.534   4.037 6.05e-05 ***
## GPUNvidia          227.598     47.445   4.797 2.00e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 345 on 654 degrees of freedom
## Multiple R-squared:  0.6441, Adjusted R-squared:  0.6386
## F-statistic: 118.3 on 10 and 654 DF,  p-value: < 2.2e-16
```

```r
# out-of-sample R² for model2
pred_test <- predict(model2, newdata = test)
sse <- sum((test$Price - pred_test)^2)
sst <- sum((test$Price - mean(test$Price))^2)
R2_out <- 1 - sse/sst
R2_out
```

```
## [1] 0.5216836
```

Here we notice that our OOS R^2 value and Multiple R-squared values have dropped. Although removing InventoryID and Weight reduces the OOS R^2 of our model (from 0.5506981 to 0.5216836), InventoryID was removed for the above mentioned reasons as it was not significant and also has no managerial insight or impact on the model. However, removing Weight was a tradeoff between predictive power and interpretability given that it is highly corelated with Screen. If our emphasis was purely on predictive power, one could make a case to include it in the model.

```r
# 3rd Model to compare with
model3 <- lm(Price ~ Screen + Memory + Company + TypeName + GPU, data = train)
summary(model3)
```

```
##
## Call:
## lm(formula = Price ~ Screen + Memory + Company + TypeName + GPU,
##     data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1094.55  -205.66   -29.65   159.11  1543.68
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        716.852    270.578   2.649  0.00826 **
## Screen             -34.580     15.850  -2.182  0.02949 *
## Memory              90.349      4.485  20.144  < 2e-16 ***
## CompanyDell        127.121     43.497   2.923  0.00359 **
## CompanyHP          177.134     44.660   3.966 8.11e-05 ***
## CompanyLenovo       64.252     42.981   1.495  0.13542
## TypeNameNotebook  -228.566     52.483  -4.355 1.54e-05 ***
## TypeNameUltrabook  209.510     65.702   3.189  0.00150 **
## GPUIntel           162.281     40.728   3.985 7.52e-05 ***
## GPUNvidia          223.443     47.652   4.689 3.34e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 346.7 on 655 degrees of freedom
## Multiple R-squared:   0.64,  Adjusted R-squared:  0.6351
## F-statistic: 129.4 on 9 and 655 DF,  p-value: < 2.2e-16
```

```r
# out-of-sample R² for model3
pred_test <- predict(model3, newdata = test)
sse <- sum((test$Price - pred_test)^2)
sst <- sum((test$Price - mean(test$Price))^2)
R2_out <- 1 - sse/sst
R2_out
```

```
## [1] 0.5225497
```

> Here I noticed that in the 3rd model, removing rating improves our OOS R^2 but worsens our Multiple R-squared. But I still chose to include it as it remains statistically significant in our model and also provides managerial impact and is intuitively a factor that consumers consider when thinking about price.
>
> After considering the models, I decided to go ahead with Model2 (i.e dropping InventoryID and Weight, but not dropping Rating) as this configuration provides much more interpretability at a slightly low prediction power. It's important to know the requirements of our client so we can decide the tradeoff between predictive power and explainability/managerial sense.

```r
model2 <- lm(Price ~ Screen + Memory + Rating + Company + TypeName + GPU, data = train)
summary(model2)
```

```
##
## Call:
```

```
## lm(formula = Price ~ Screen + Memory + Rating + Company + TypeName +
##     GPU, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1040.73  -194.38   -36.05   166.39  1559.87
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        823.027    272.079   3.025  0.00258 **
## Screen             -36.598     15.791  -2.318  0.02077 *
## Memory              90.206      4.464  20.208  < 2e-16 ***
## Rating             -12.815      4.710  -2.721  0.00669 **
## CompanyDell        124.905     43.294   2.885  0.00404 **
## CompanyHP          175.313     44.449   3.944 8.87e-05 ***
## CompanyLenovo       60.894     42.790   1.423  0.15519
## TypeNameNotebook  -234.455     52.273  -4.485 8.60e-06 ***
## TypeNameUltrabook  203.684     65.418   3.114  0.00193 **
## GPUIntel           163.638     40.534   4.037 6.05e-05 ***
## GPUNvidia          227.598     47.445   4.797 2.00e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 345 on 654 degrees of freedom
## Multiple R-squared:  0.6441, Adjusted R-squared:  0.6386
## F-statistic: 118.3 on 10 and 654 DF,  p-value: < 2.2e-16
```

(B)

In the final model, the effects of Screen, Memory, Company, Rating, Ultrabook type, and GPU are managerially sensible, as they align with expectations that screen size, RAM, premium designs, brand value, GPUs have a strong impact on laptop prices. However, the negative effect of Screen size and Rating seems to be conterintuitive. So it is imperative to investigate those features further in depth and look carefully at the feature data provided. But the other variables included do follow expectations based on their estimates and their impact on price.

(C)

Based on the model, HP has the highest effect on laptop price adding 175.313 more than Asus holding other factors constant, commanding the largest premium over the other brands. Even though Lenovo's coefficients are the lowest among the listed coefficients, one can argue that Asus has the smallest effect, since it is the baseline and all other manufacturers have higher estimated coefficients. While Lenovo's effect is positive (60.894), it is not statistically significant, which suggests Lenovo laptops are priced similarly to Asus on average.

(D)

```r
pred_test <- predict(model2, newdata = test)
sse <- sum((test$Price - pred_test)^2)
sst <- sum((test$Price - mean(test$Price))^2)
R2_out <- 1 - sse/sst
R2_out  # out-of-sample R²
```

```
## [1] 0.5216836
```

Interpretation: This means that when predicting laptop prices on unseen test data,the model explains about 52.16836% of the variation in prices compared to simply predicting the mean price for all laptops.

Formally, out-of-sample R^2 is defined as $R^2 = 1$ - (SSE/SST), where SSE is the sum of squared prediction errors on the test set and SST is the total sum of squared deviations of the test set prices from their mean. An out-of-sample R^2 of 0.5216836 indicates that the model explains 52.16836% of the variation in laptop prices in the test data, compared to a baseline model that always predicts the average price.

(E)

```r
newlap <- data.frame(
  InventoryID = 950,
  Screen = 15.6,
  Memory = 6,
  Weight = 3,
  Rating = 8,
  Company = factor("Asus", levels = levels(train$Company)),
  TypeName = factor("Ultrabook", levels = levels(train$TypeName)),
  GPU = factor("Intel", levels = levels(train$GPU))
)

# Prediction with prediction interval (includes error variance)
pred <- predict(model2, newdata = newlap, interval = "prediction", level = 0.95)
pred
```

```
##        fit      lwr      upr
## 1 1058.133 371.3338 1744.931
```

```r
p <- predict(model2, newdata = newlap, se.fit = TRUE)
sigma2 <- summary(model2)$sigma^2
se_pred <- sqrt(p$se.fit^2 + sigma2)
df <- model2$df.residual

t_stat <- (1100 - p$fit) / se_pred
prob_gt_1100 <- 1 - pt(t_stat, df = df)
prob_gt_1100
```

```
##         1
## 0.4523782
```

For this laptop, the model predicts an average price of 1,058.133 euros, with a 95% prediction interval ranging from 371.3338 euros to 1,744.931 euros. The probability that the actual price exceeds €1,100 is 45.23782%. This calculation is based on the assumptions that regression errors are normally distributed (so the t-distribution approximation for the probability is valid), homoscedastic, and independent, and that the model is correctly specified.

(F)

```r
model_mem_gpu <- lm(Price ~ Screen + Memory + Rating + TypeName + Company +
                      GPU + Memory:GPU, data = train)
summary(model_mem_gpu)
```

```
##
## Call:
## lm(formula = Price ~ Screen + Memory + Rating + TypeName + Company +
##     GPU + Memory:GPU, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1086.71  -194.71   -30.79   158.38  1569.31
```

```
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        1044.932    277.494   3.766 0.000181 ***
## Screen              -38.107     15.765  -2.417 0.015915 *
## Memory               60.243      9.964   6.046  2.5e-09 ***
## Rating              -13.055      4.675  -2.792 0.005384 **
## TypeNameNotebook   -211.650     54.205  -3.905 0.000104 ***
## TypeNameUltrabook   213.135     69.884   3.050 0.002382 **
## CompanyDell         121.588     42.982   2.829 0.004816 **
## CompanyHP           165.627     44.206   3.747 0.000195 ***
## CompanyLenovo        61.037     42.503   1.436 0.151466
## GPUIntel            -70.059     90.786  -0.772 0.440573
## GPUNvidia           -78.966    101.655  -0.777 0.437557
## Memory:GPUIntel      32.854     11.941   2.751 0.006102 **
## Memory:GPUNvidia     39.956     11.632   3.435 0.000631 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 342.4 on 652 degrees of freedom
## Multiple R-squared:  0.6505, Adjusted R-squared:  0.6441
## F-statistic: 101.1 on 12 and 652 DF,  p-value: < 2.2e-16
```

```r
pred_test <- predict(model_mem_gpu, newdata = test)
sse <- sum((test$Price - pred_test)^2)
sst <- sum((test$Price - mean(test$Price))^2)
R2_out <- 1 - sse/sst
R2_out  # out-of-sample R²
```

```
## [1] 0.5180113
```

The interaction terms show that the effect of Memory depends on the GPU type installed. For AMD laptops, each extra unit of memory adds about 60.243 euros to price. For Intel and Nvidia laptops, the memory premium is much higher, about 93.097 euros and 100.199 euros per memory unit, respectively. This suggests that additional RAM is valued more when paired with stronger GPUs. Compared to the model in part (a), this specification highlights brand-technology complementarities, and shows that consumers value memory more when paired with GPUs, though overall model fit (adjusted R^2) is slightly lower. When I added the Memory × GPU interaction, the adjusted R^2 increased slightly (0.639 to 0.644), but the out-of-sample R^2 decreased marginally (0.522 to 0.518) which suggests no predictive gain, but the model provides richer interpretation: the price premium for additional RAM depends on GPU type.

(G)

```r
train$Company  <- factor(train$Company,  levels = c("Asus","Dell","HP","Lenovo"))
train$TypeName <- factor(train$TypeName, levels = c("Gaming","Notebook","Ultrabook"))
train$GPU      <- factor(train$GPU,      levels = c("AMD","Intel","Nvidia"))

# Model with GPU × Company interaction
model_gpu_company <- lm(
  Price ~ Screen + Memory + Rating + TypeName + GPU + Company + GPU:Company,
  data = train
)
summary(model_gpu_company)
```

```
##
```

```
## Call:
## lm(formula = Price ~ Screen + Memory + Rating + TypeName + GPU +
##     Company + GPU:Company, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1162.7  -201.0   -16.2   176.6  1515.5
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)             1101.353    293.294   3.755 0.000189 ***
## Screen                   -36.089     15.640  -2.307 0.021343 *
## Memory                    88.693      4.388  20.212  < 2e-16 ***
## Rating                   -12.318      4.623  -2.665 0.007901 **
## TypeNameNotebook        -163.473     53.046  -3.082 0.002145 **
## TypeNameUltrabook        295.321     66.385   4.449 1.02e-05 ***
## GPUIntel                -270.926    141.486  -1.915 0.055950 .
## GPUNvidia                -56.560    135.229  -0.418 0.675900
## CompanyDell             -253.432    138.346  -1.832 0.067429 .
## CompanyHP               -186.109    143.655  -1.296 0.195600
## CompanyLenovo           -297.136    151.692  -1.959 0.050563 .
## GPUIntel:CompanyDell     394.962    151.783   2.602 0.009476 **
## GPUNvidia:CompanyDell    517.452    153.830   3.364 0.000814 ***
## GPUIntel:CompanyHP       472.646    155.538   3.039 0.002471 **
## GPUNvidia:CompanyHP      178.070    162.639   1.095 0.273978
## GPUIntel:CompanyLenovo   490.746    163.354   3.004 0.002766 **
## GPUNvidia:CompanyLenovo  242.211    162.478   1.491 0.136519
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 337.5 on 648 degrees of freedom
## Multiple R-squared:  0.6625, Adjusted R-squared:  0.6541
## F-statistic: 79.49 on 16 and 648 DF,  p-value: < 2.2e-16
```

```r
pred_test <- predict(model_gpu_company, newdata = test)
sse <- sum((test$Price - pred_test)^2)
sst <- sum((test$Price - mean(test$Price))^2)
R2_out <- 1 - sse/sst
R2_out  # out-of-sample R²
```

```
## [1] 0.5331728
```

> The GPU * Company interaction terms show that the price impact of GPUs varies by manufacturer. For Asus (baseline), Intel and Nvidia GPUs do not add value, but for Dell, HP, and Lenovo, Intel GPUs command strong positive premiums of about 394-491 euros. Nvidia GPUs also add large premiums at Dell but not at HP or Lenovo. Compared to the model in part (a), this specification provides richer insight into brand-specific GPU pricing and slightly improves both adjusted R^2 (0.639 to 0.654) and out-of-sample R^2 (0.522 to 0.533).

Problem 2

Hindy Rossignol, Riya Parikh, Mrugank Pednekar, Ioannis Panagiotopoulos

2025-09-11

R setup

```r
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

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

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

```r
library(fitdistrplus)
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
## Loading required package: survival
```

Problem 2

Part a

```r
# loading df
df_insurance <- read_csv("/Users/riyaparikh_computeracct/Downloads/MIT/15.072_AdvancedAnalyticsEdge/deliverable1-analyticsedge-mit/Data/insurance.csv", show_col_types = FALSE)
```

Part b

```
mean_charges = mean(df_insurance$charges)
sd_charges = sd(df_insurance$charges)
```

Mean = 13270.42, SD = 12110.01

Part c

```
probability_between_values = pnorm(14000, mean_charges, sd_charges) - pnorm(8000, mean_c
harges, sd_charges)
```

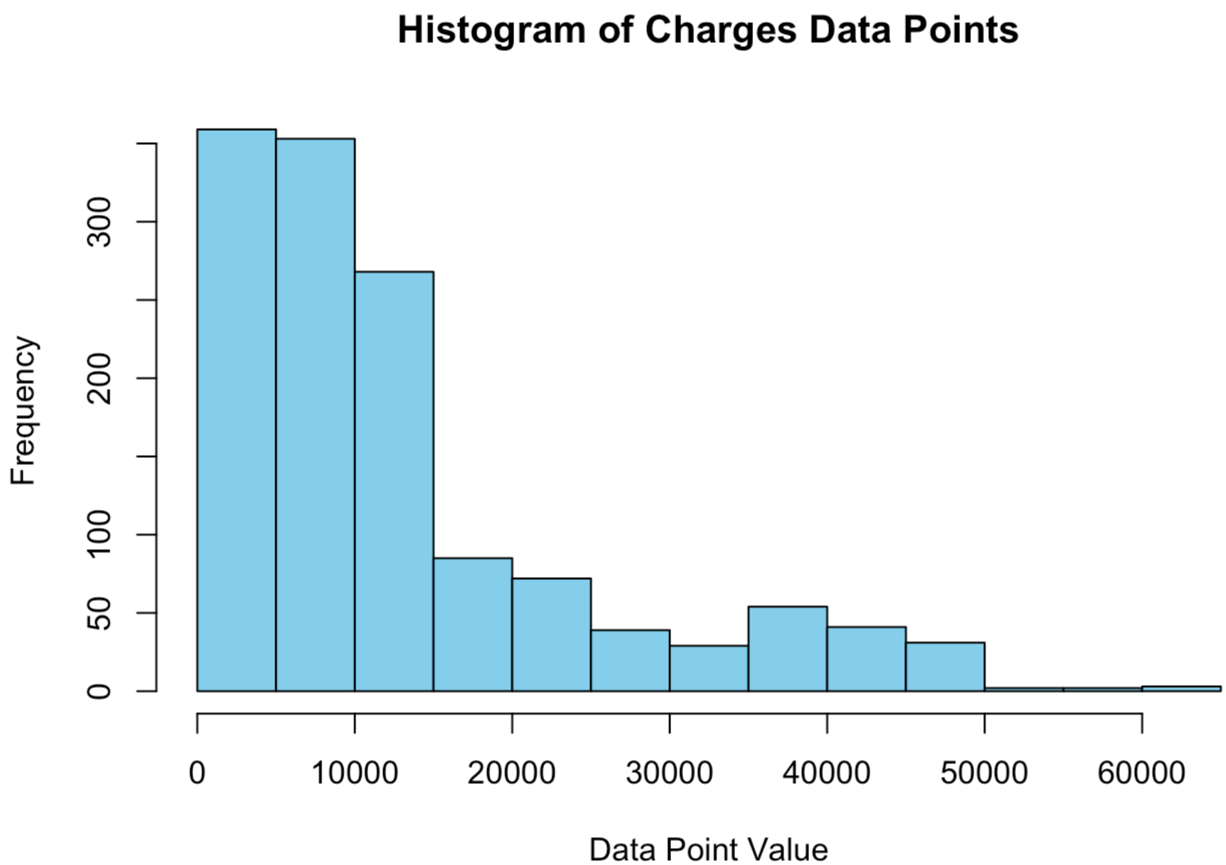
The probability of being between $8000 and $14000 based on the Normal distribution is 0.19.

Part d

```
count_in_range <- sum(df_insurance$charges >= 8000 & df_insurance$charges <= 14000)
total_vals = length(df_insurance$charges)
true_prop_between_values = count_in_range/total_vals
median(df_insurance$charges)
```

```
## [1] 9382.033
```

```
hist(df_insurance$charges, main = "Histogram of Charges Data Points", xlab = "Data Point
Value", col = "skyblue")
```



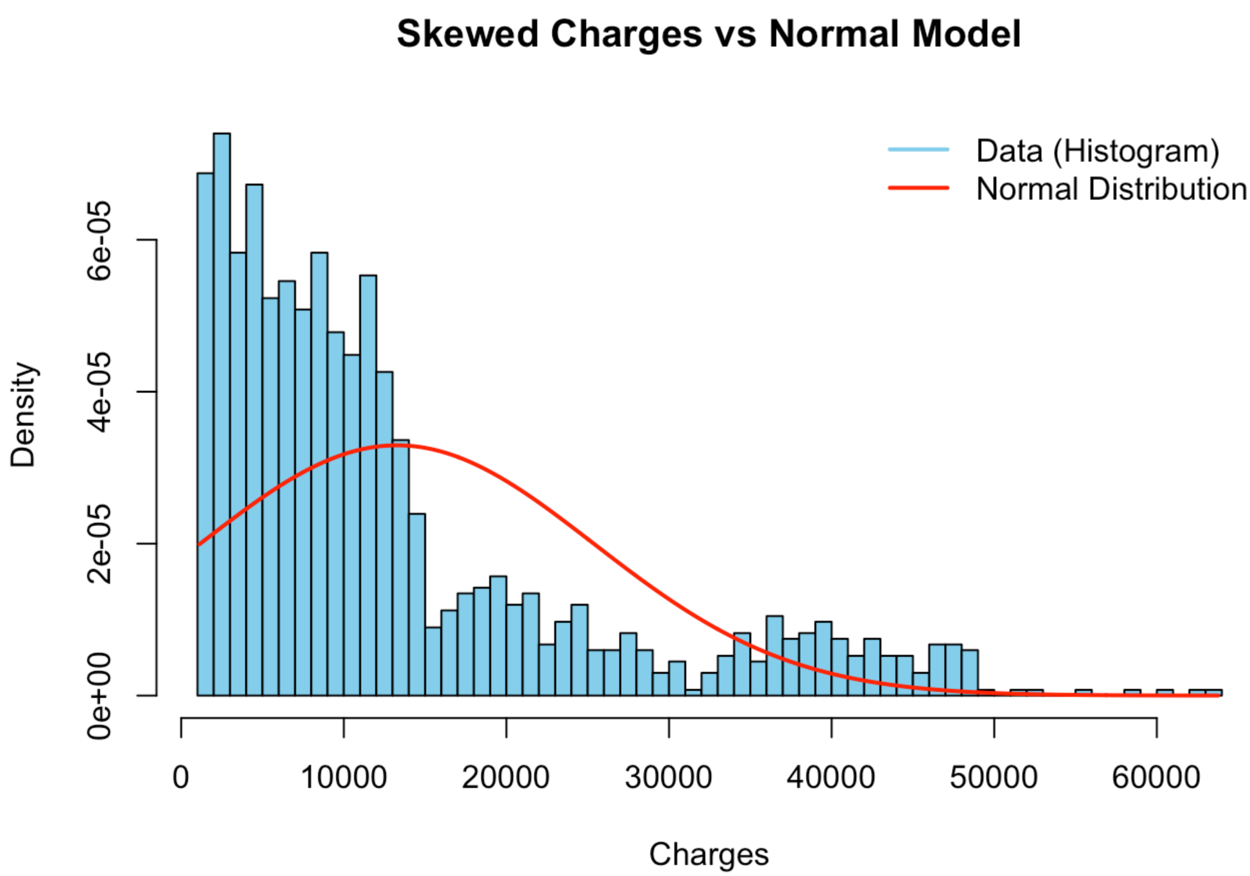

```
# to see overlapping stats plots - ours vs normal model
# calculate values for comparison
count_in_range <- sum(df_insurance$charges >= 8000 & df_insurance$charges <= 14000)
total_vals <- length(df_insurance$charges)
true_prop_between_values <- count_in_range / total_vals
print(true_prop_between_values)
```

```
## [1] 0.2825112
```

```
median(df_insurance$charges)
```

```
## [1] 9382.033
```

```r
# histogram of charges (empirical distribution)
hist(df_insurance$charges,
     breaks = 50,                    # more detail
     freq = FALSE,                   # scale histogram to density
     main = "Skewed Charges vs Normal Model",
     xlab = "Charges",
     col = "skyblue")

# overlay fitted normal distribution curve
x_vals <- seq(min(df_insurance$charges), max(df_insurance$charges), length.out = 1000)
y_norm <- dnorm(x_vals, mean = mean_charges, sd = sd_charges)

lines(x_vals, y_norm, col = "red", lwd = 2)

# add legend
legend("topright", legend = c("Data (Histogram)", "Normal Distribution"),
       col = c("skyblue", "red"), lwd = 2, bty = "n")
```



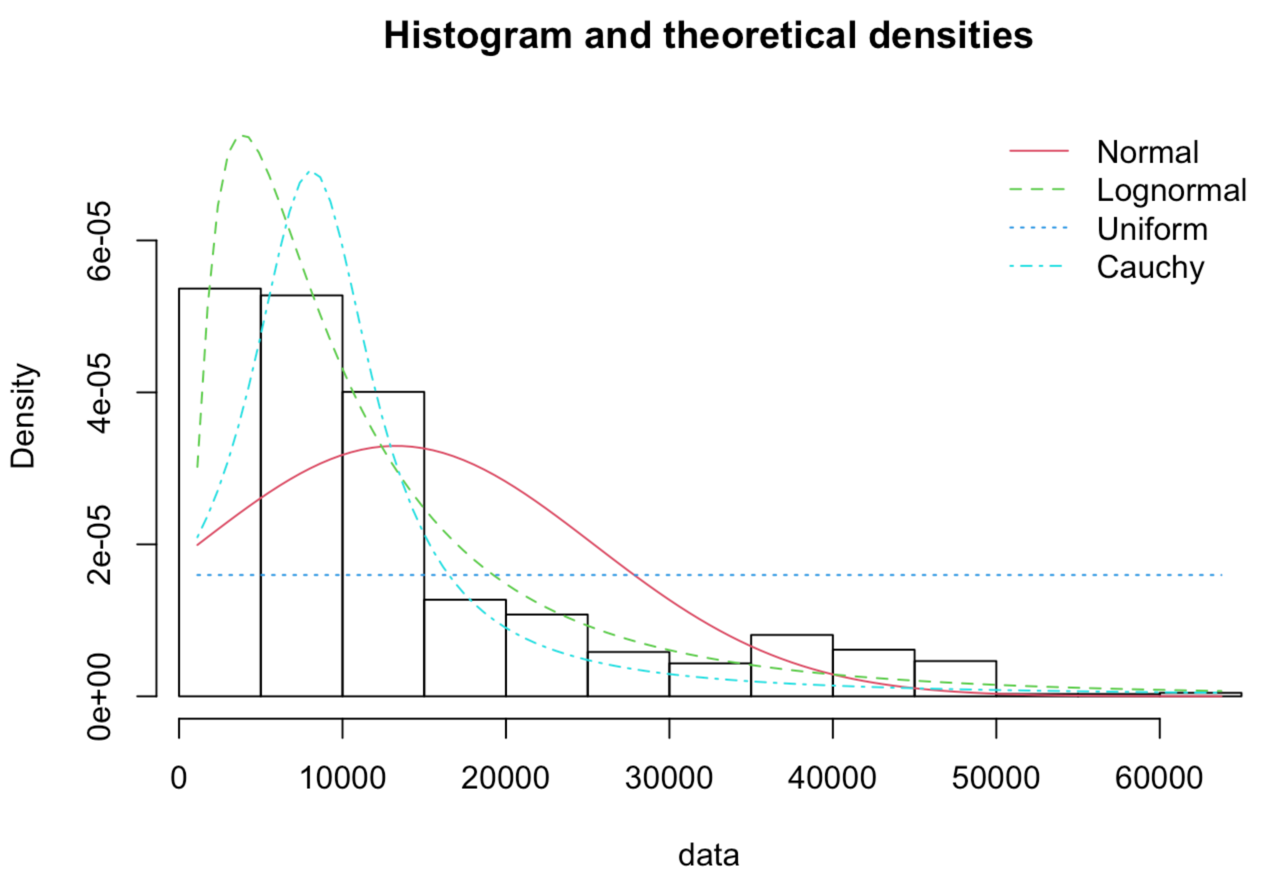
Actual proportion of charges between $8000 and $14000 is 0.28. This is not the same as the probability we got when assuming the charges follow a Normal distribution in part c. Our initial assumption is that the charges do not follow a Normal distribution. The histogram we plotted of charges data points shows that the distribution is right skewed towards higher charges. We also see that the median charge is lower than the mean charge ($9382.033 vs $13270.42).

Part e

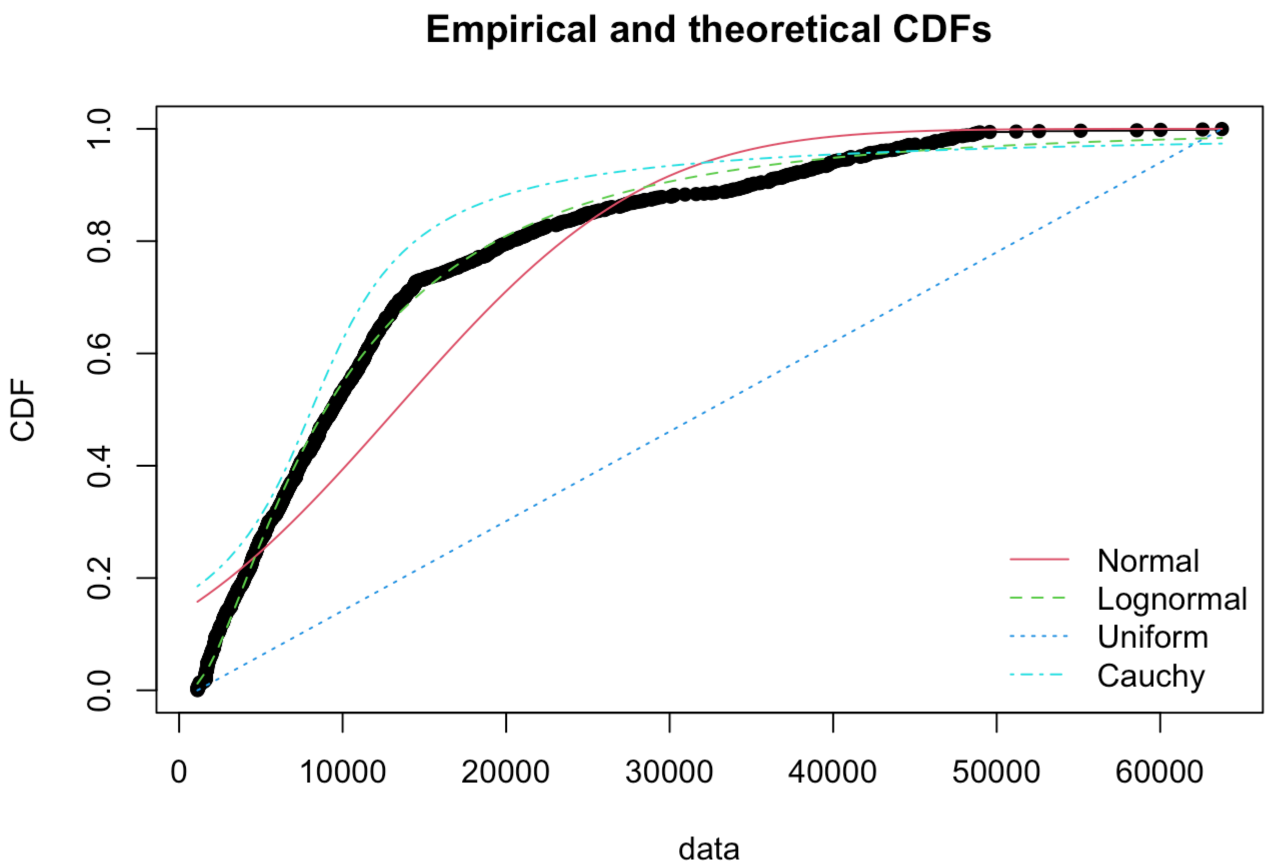
```
fit_norm <- fitdist(df_insurance$charges, "norm")
fit_lognorm <- fitdist(df_insurance$charges, "lnorm")
fit_unif <- fitdist(df_insurance$charges, "unif")
fit_cauchy <- fitdist(df_insurance$charges, "cauchy")

gofstat(list(fit_norm, fit_lognorm, fit_unif, fit_cauchy))
```

```
## Goodness-of-fit statistics
##                              1-mle-norm 2-mle-lnorm  3-mle-unif 4-mle-cauchy
## Kolmogorov-Smirnov statistic   0.188462   0.0365844   0.5147556     0.185312
## Cramer-von Mises statistic    14.829729   0.3973136 155.5524079     9.501239
## Anderson-Darling statistic    85.138872   3.9424972         Inf    65.856208
##
## Goodness-of-fit criteria
##                                1-mle-norm 2-mle-lnorm 3-mle-unif 4-mle-cauchy
## Akaike's Information Criterion   28959.26    27923.58   29561.21     28716.76
## Bayesian Information Criterion   28969.66    27933.98   29571.61     28727.16
```

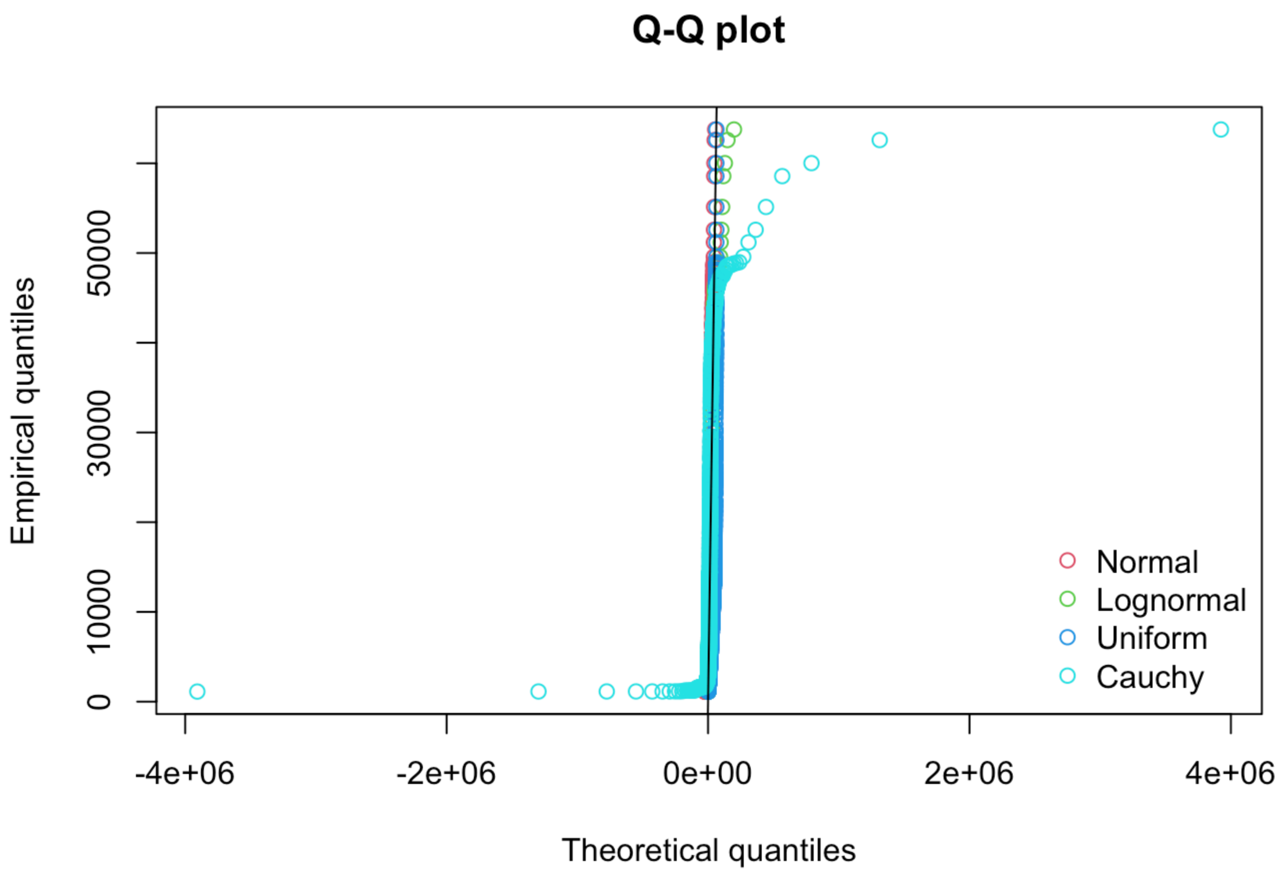
```
plot.legend <- c("Normal", "Lognormal", "Uniform", "Cauchy")

denscomp(list(fit_norm, fit_lognorm, fit_unif, fit_cauchy), legendtext = plot.legend)
```



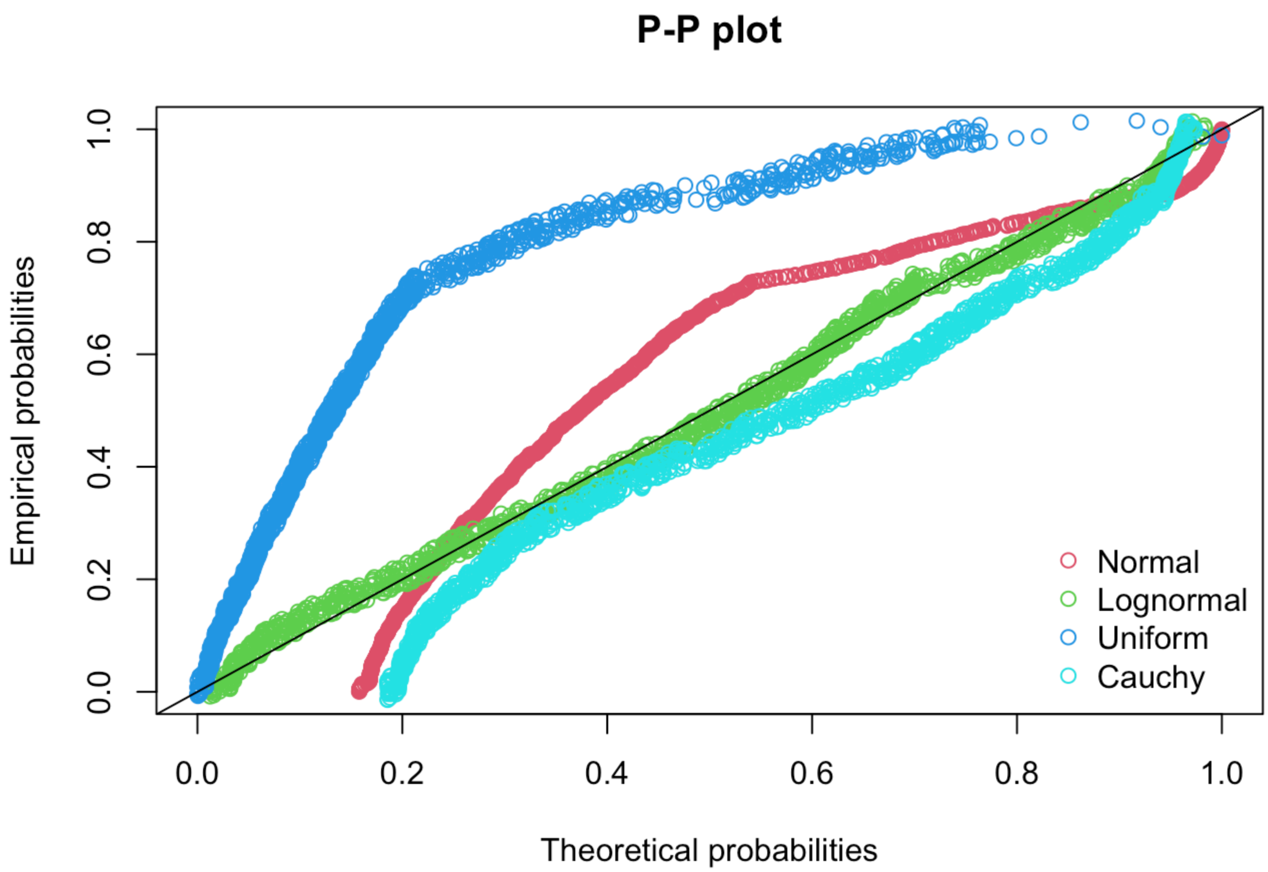
```
cdfcomp(list(fit_norm, fit_lognorm, fit_unif, fit_cauchy), legendtext = plot.legend)
```




```
qqcomp(list(fit_norm, fit_lognorm, fit_unif, fit_cauchy), legendtext = plot.legend)
```




```
ppcomp(list(fit_norm, fit_lognorm, fit_unif, fit_cauchy), legendtext = plot.legend)
```




```
quantile(df_insurance$charges, probs = c(0.25, 0.5, 0.75))
```

```
##       25%       50%       75%
##  4740.287  9382.033 16639.913
```

```
qlnorm(c(0.25, 0.5, 0.75),
       meanlog = fit_lognorm$estimate["meanlog"],
       sdlog = fit_lognorm$estimate["sdlog"])
```

```
## [1]  4811.090  8943.289 16624.595
```

Based on both statistical metrics and visual diagnostics, the lognormal distribution is the best fit for the insurance charges data. It consistently outperforms alternatives across goodness-of-fit statistics, with a notably lower KS statistic indicating a closer match between empirical and theoretical distributions. The quantile comparison confirms that lognormal quantiles align tightly with the actual data, especially in the right tail.

To support this conclusion, we examine four key plots:

- Histogram with Density Curves: The histogram reveals a pronounced right skew in the data. Among the overlaid theoretical curves, the lognormal density (green dashed line) hugs the histogram most closely, especially in the peak and tail regions. The Normal and uniform curves fail to capture the asymmetry, while the Cauchy curve overestimates the tail.

- Empirical vs. Theoretical CDFs: The lognormal CDF tracks the empirical CDF almost perfectly across the entire range. The Normal CDF diverges in the upper tail, underestimating high charges. Uniform and Cauchy distributions show poor alignment, with noticeable deviations throughout.
- Q-Q Plot: The lognormal quantiles (green triangles) lie closest to the diagonal, indicating strong agreement with the empirical quantiles. The Normal distribution shows curvature, especially in the tails, while the Cauchy and uniform quantiles deviate substantially, suggesting poor fit.
- P-P Plot: Lognormal points cluster tightly around the diagonal, confirming that the predicted probabilities match the observed proportions well. Normal distribution points begin to drift in the upper range, and both uniform and Cauchy show systematic deviations.

Taken together, these plots reinforce the statistical conclusion: the lognormal distribution best captures the shape, spread, and tail behavior of the insurance charges data. It accommodates the skewness and heavy upper tail, which the normal distribution fails to model adequately.

Part f

```
mod <-  lm(charges ~ age + factor(sex) + bmi + children + factor(smoker) + factor(regio
n) , data = df_insurance)
summary_mod = summary(mod)
summary_mod
```

```
## 
## Call:
## lm(formula = charges ~ age + factor(sex) + bmi + children + factor(smoker) + 
##     factor(region), data = df_insurance)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -11304.9  -2848.1   -982.1   1393.9  29992.8 
## 
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)    
## (Intercept)             -11938.5      987.8 -12.086  < 2e-16 ***
## age                        256.9       11.9  21.587  < 2e-16 ***
## factor(sex)male           -131.3      332.9  -0.394 0.693348    
## bmi                        339.2       28.6  11.860  < 2e-16 ***
## children                   475.5      137.8   3.451 0.000577 ***
## factor(smoker)yes        23848.5      413.1  57.723  < 2e-16 ***
## factor(region)northwest   -353.0      476.3  -0.741 0.458769    
## factor(region)southeast  -1035.0      478.7  -2.162 0.030782 *  
## factor(region)southwest   -960.0      477.9  -2.009 0.044765 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 6062 on 1329 degrees of freedom
## Multiple R-squared:  0.7509, Adjusted R-squared:  0.7494 
## F-statistic: 500.8 on 8 and 1329 DF,  p-value: < 2.2e-16
```

Part g

It is important to note that for the following interpretations, the change per unit that is being described only holds true if all other factors are held constant.

Age: for every increase of 1 yr in age, the charges are predicted to increase by $256.9, holding other variables constant. this variable is statistically significant as the pvalue is <0.05 at 2e-16.

Factor(sex)male: being a male, the charges are predicted to be lower by $131.3 compared to their equivalent female counterparts, holding other variables constant. this variable is not statistically significant as the pvalue is >0.05 at 0.693348.

BMI: for every increase of 1 unit in bmi, the charges are predicted to increase by $339.2, holding other variables constant. this variable is statistically significant as the pvalue is <0.05 at 2e-16.

Children: for every additional child the customer has, the charges are predicted to increase by $475.5, holding other variables constant. this variable is statistically significant as the pvalue is <0.05 at 0.000577.

Factor(smoker)yes: being a smoker, the charges are predicted to be higher by $23848.5 compared to their nonsmoking counterparts, holding other variables constant. this variable is statistically significant as the pvalue is <0.05 at 2e-16.

Factor(region)northwest: if a customer lives in the northwest, their charges are predicted to be lower by $353.0 holding all other variables constant. this variable is not statistically significant as the pvalue is <0.05 at 0.458769.

Factor(region)southeast: if a customer lives in the southeast, their charges are predicted to be lower by $1035.0 holding all other variables constant. this variable is statistically significant as the pvalue is <0.05 at 0.030782.

Factor(region)southwest: if a customer lives in the southwest, their charges are predicted to be lower by $960.0 holding all other variables constant. this variable is statistically significant as the pvalue is <0.05 at 0.044765.

Part h

```
mod2 <-  lm(charges ~ factor(sex) + bmi + children + factor(smoker) + factor(region) , d
ata = df_insurance)
summary_mod2 = summary(mod2)
summary_mod2
```

```
##
## Call:
## lm(formula = charges ~ factor(sex) + bmi + children + factor(smoker) +
##     factor(region), data = df_insurance)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -15013  -4646   -945   3652  32122
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -3953.0     1064.1  -3.715 0.000212 ***
## factor(sex)male            -310.9      386.7  -0.804 0.421590
## bmi                         411.3       33.0  12.464  < 2e-16 ***
## children                    597.5      160.0   3.735 0.000196 ***
## factor(smoker)yes         23658.9      479.9  49.303  < 2e-16 ***
## factor(region)northwest    -392.4      553.3  -0.709 0.478359
## factor(region)southeast   -1410.3      555.7  -2.538 0.011274 *
## factor(region)southwest   -1031.9      555.2  -1.859 0.063312 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7043 on 1330 degrees of freedom
## Multiple R-squared:  0.6636, Adjusted R-squared:  0.6618
## F-statistic: 374.8 on 7 and 1330 DF,  p-value: < 2.2e-16
```

The first model had an r^2 of .751 meaning that 75.1% of the variation in charges is accounted for by the model built on all the factors. the adj r^2 is .749. The second model (built without age) has an r^2 of .664 meaning that 66.4% of the variation in charges is accounted for by the model built on all the factors without age. the adj r^2 is .662. When age is not considered as an independent variable, the model is forced to "spread" the influence of age across other variables like BMI, smoking status, and region. Younger individuals are likely to be overcharged: without age, the model can't distinguish between a healthy 25-year-old and a 55-year-old with similar BMI and smoking status. Older individuals are likely to be undercharged: older individuals typically incur higher medical costs, which the model without age can't fully capture.

Part i

I would recommend applying a log transformation to the charges variable, and then redoing the analysis using the transformed variable to assess linear relationships and model assumptions more effectively.

Problem_3

Hindy Rossignol, Riya Parikh, Mrugank Pednekar, Ioannis Panagiotopoulos

2025-09-11

```r
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.4     v tibble    3.3.0
## v purrr     1.1.0     v tidyr     1.3.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts
↪  to become errors
```

```r
# Load the mtcars dataset
data(mtcars)
# Display basic information about the dataset
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```r
summary(mtcars)
```

```
##       mpg             cyl             disp             hp
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0
##       drat             wt             qsec             vs
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000
```

```
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000
##        am              gear            carb
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
##  Median :0.0000   Median :4.000   Median :2.000
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

Sanity Check for our Models

To make sure our models are correct we will use a function to visualize their equation explicity.

```r
# This is our function to visualize the equations
eq_print <- function(mod, digits = 4, mult_sign = " * ") {
  b <- coef(mod)
  fmt <- function(x) formatC(x, digits = digits, format = "f")
  parts <- Map(function(name, val) {
    if (name == "(Intercept)") return(fmt(val))
    nm <- gsub(":", mult_sign, name, fixed = TRUE)
    paste0(ifelse(val >= 0, " + ", " - "), fmt(abs(val)), " ", nm)
  }, names(b), b)
  intercept <- parts[[which(names(b) == "(Intercept)")[1]]]
  others <- parts[names(b) != "(Intercept)"]
  paste0("y_hat = ", intercept, paste0(others, collapse = ""))
}
```

Part a) Fit Model 1 and show summary

```r
# Model 1: mpg ~ hp + wt + hp*wt
model1 <- lm(mpg ~ hp + wt + hp*wt, data = mtcars)

summary(model1)
```

```
##
## Call:
## lm(formula = mpg ~ hp + wt + hp * wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0632 -1.6491 -0.7362  1.4211  4.5513
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 49.80842    3.60516  13.816 5.01e-14 ***
## hp          -0.12010    0.02470  -4.863 4.04e-05 ***
## wt          -8.21662    1.26971  -6.471 5.20e-07 ***
## hp:wt        0.02785    0.00742   3.753 0.000811 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.153 on 28 degrees of freedom
## Multiple R-squared:  0.8848, Adjusted R-squared:  0.8724
## F-statistic: 71.66 on 3 and 28 DF,  p-value: 2.981e-13
```

```
cat(sprintf("Model %d's equation: %s\n", 1, eq_print(model1)))
```

```
## Model 1's equation: y_hat = 49.8084 - 0.1201 hp - 8.2166 wt + 0.0278 hp * wt
```

Part b) Interpretation of coefficients in Model 1

The interpretation of the coefficients in Model 1:

- **hp coefficient (-0.12010)**: This is the effect of horsepower when weight is equal to zero. While a car cannot realistically weigh zero, this coefficient combines with the interaction term to determine the true effect of hp at different weights. Formally, the marginal effect of horsepower on mpg is:

$$\frac{\partial mpg}{\partial hp} = -0.12010 + 0.02785 \cdot wt$$

  This means that for each additional unit of horsepower, mpg decreases by about 0.12 miles per gallon, but this negative effect becomes less severe as weight increases.

- **wt coefficient (-8.21662)**: This is the effect of weight when horsepower is equal to zero. Again, this is not realistic in practice, but together with the interaction term, it determines how the slope changes at different hp levels. Formally, the marginal effect of weight on mpg is:

$$\frac{\partial mpg}{\partial wt} = -8.21662 + 0.02785 \cdot hp$$

  This means that for each additional unit of weight (1000 lbs), mpg decreases by about 8.22 miles per gallon, but this negative effect becomes less severe as horsepower increases.

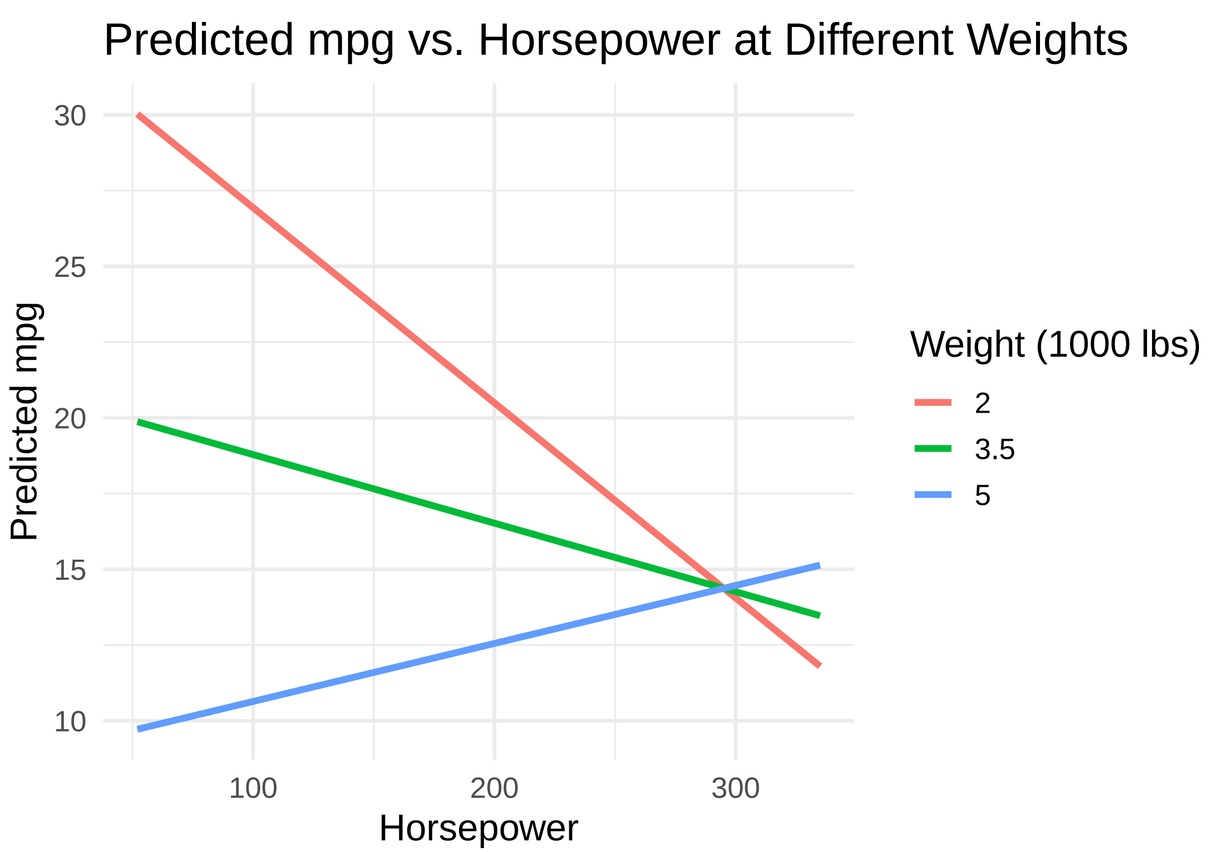
- **hp wt interaction coefficient (0.02785)**: The positive interaction means that the effect of horsepower depends on weight (and vice versa). Specifically:
  - For heavier cars, the negative effect of horsepower on mpg becomes *less severe*.
  - For more powerful cars, the negative effect of weight on mpg becomes *less severe*.

In other words, extra horsepower is more harmful for light cars than for heavy cars, while extra weight is more harmful for low-power cars than for high-power ones. To better understand this, we can plot the following graph:

```
# Visualising the interaction effect

# Create a grid of values for predictions
newdata <- expand.grid(
  hp = seq(min(mtcars$hp), max(mtcars$hp), length.out = 100),
  wt = c(2, 3.5, 5) # choose three representative weights (1000 lbs units)
)

# Add predictions
```

```r
newdata$mpg_pred <- predict(model1, newdata)

# Plot
ggplot(newdata, aes(x = hp, y = mpg_pred, colour = factor(wt))) +
  geom_line(size = 1.2) +
  labs(
    title = "Predicted mpg vs. Horsepower at Different Weights",
    x = "Horsepower",
    y = "Predicted mpg",
    colour = "Weight (1000 lbs)"
  ) +
  theme_minimal(base_size = 14)
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



As you can see in the above, the effect of increasing horsepower for the lightest car (in red) is clearly leading to a bigger decrease in the predicted miles per gallon (mpg) than it is for a car slightly heavier (in green). Additionally, for a car much heavier, we can see that increasing power can even lead to an increase in the predicted mpg!

Part c) Model 2 with mean-centered variables

```r
# Create mean-centered variables
mtcars$hp_centered <- mtcars$hp - mean(mtcars$hp)
mtcars$wt_centered <- mtcars$wt - mean(mtcars$wt)

# Model 2: mpg ~ hp_centered + wt_centered + hp_centered*wt_centered
model2 <- lm(mpg ~ hp_centered + wt_centered + hp_centered*wt_centered, data = mtcars)

# Summary of stats of model2
summary(model2)
```

```
##
## Call:
## lm(formula = mpg ~ hp_centered + wt_centered + hp_centered *
##     wt_centered, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0632 -1.6491 -0.7362  1.4211  4.5513
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             18.898400   0.495703  38.124  < 2e-16 ***
## hp_centered             -0.030508   0.007503  -4.066 0.000352 ***
## wt_centered             -4.131649   0.529558  -7.802 1.69e-08 ***
## hp_centered:wt_centered  0.027848   0.007420   3.753 0.000811 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.153 on 28 degrees of freedom
## Multiple R-squared:  0.8848, Adjusted R-squared:  0.8724
## F-statistic: 71.66 on 3 and 28 DF,  p-value: 2.981e-13
```

```r
cat(sprintf("Model %d's equation: %s\n", 2, eq_print(model2)))
```

```
## Model 2's equation: y_hat = 18.8984 - 0.0305 hp_centered - 4.1316 wt_centered + 0.0278
↪  hp_centered * wt_centered
```

Comparison of coefficients between Model 1 and Model 2:

The underlying objective in centering the values taken by hp and wt around their mean is that, now, the hp coefficient represents the effect of horsepower on mpg when weight is at its average value, which is a much more realistic model than model1. Similarly, the wt coefficient represents the effect of weight on mpg when horsepower is at its mean value.

- The **interaction coefficient** remains exactly the same (0.02784815) in both models. This is a key property of mean-centering: it preserves the interaction coefficient (see question e).
- The **main effect coefficients** change significantly:
  - hp coefficient: -0.12010 (Model 1) vs -0.03051 (Model 2)
  - wt coefficient: -8.21662 (Model 1) vs -4.13165 (Model 2)

- The **intercept** changes from 49.80842 (Model 1) to 18.89840 (Model 2)

Key insight: The main effect coefficients in Model 2 represent the marginal effects at the mean values of the other variable. For example, the hp coefficient in Model 2 (-0.03051) is the effect of horsepower when weight equals its mean (3.21725), which can be verified by calculating the marginal effect from Model 1: $-0.12010 + 0.02785 \times 3.21725 = -0.03051$.

R-squared Analysis for Models 1 and 2: Both models have identical R-squared values (≈ 0.8848) and adjusted R-squared values (≈ 0.8724), which confirms that mean-centering does not change the model's explanatory power. This is expected because: - Mean-centering is a linear transformation that preserves the linear relationships - The interaction coefficient remains unchanged - The model fit is mathematically equivalent, just with different coefficient interpretations

Part d) Model 3 with only interaction term

```r
# Model 3: mpg ~ hp_centered:wt_centered (only interaction, no main effects)
model3 <- lm(mpg ~ hp_centered:wt_centered, data = mtcars)
summary(model3)
```

```
##
## Call:
## lm(formula = mpg ~ hp_centered:wt_centered, data = mtcars)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.3892  -4.0066  -0.3153   2.8009  12.7062
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)             19.41692    1.39189   13.95 1.19e-14 ***
## hp_centered:wt_centered  0.01574    0.02072    0.76    0.453
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.068 on 30 degrees of freedom
## Multiple R-squared:  0.01887,    Adjusted R-squared:  -0.01383
## F-statistic: 0.5771 on 1 and 30 DF,  p-value: 0.4534
```

```r
cat(sprintf("Model %d's equation: %s\n", 3, eq_print(model3)))
```

```
## Model 3's equation: y_hat = 19.4169 + 0.0157 hp_centered * wt_centered
```

Comparison of coefficients across all three models:

Model 3 results

- **Intercept:** 19.41692

- **hp_centered wt_centered:** 0.01574

```r
# Create a comparison table of R-squared values
r_squared_comparison <- data.frame(
  Model = c("Model 1", "Model 2", "Model 3"),
  R_squared = c(summary(model1)$r.squared, summary(model2)$r.squared,
  ↪  summary(model3)$r.squared),
  Adj_R_squared = c(summary(model1)$adj.r.squared, summary(model2)$adj.r.squared,
  ↪  summary(model3)$adj.r.squared),
  F_statistic = c(summary(model1)$fstatistic[1], summary(model2)$fstatistic[1],
  ↪  summary(model3)$fstatistic[1]),
  P_value = c(pf(summary(model1)$fstatistic[1], summary(model1)$fstatistic[2],
  ↪  summary(model1)$fstatistic[3], lower.tail = FALSE),
              pf(summary(model2)$fstatistic[1], summary(model2)$fstatistic[2],
              ↪  summary(model2)$fstatistic[3], lower.tail = FALSE),
              pf(summary(model3)$fstatistic[1], summary(model3)$fstatistic[2],
              ↪  summary(model3)$fstatistic[3], lower.tail = FALSE))
)

print(r_squared_comparison)
```

R-squared Comparison Across All Models

```
##     Model  R_squared Adj_R_squared F_statistic      P_value
## 1 Model 1 0.88476371    0.87241697  71.6596724 2.981394e-13
## 2 Model 2 0.88476371    0.87241697  71.6596724 2.981394e-13
## 3 Model 3 0.01887209   -0.01383218   0.5770528 4.533982e-01
```

Key R-squared Observations:

1. **Models 1 and 2 are mathematically equivalent::** Both achieve $R^2 \approx 0.8848$ and adjusted $R^2 \approx 0.8848$, confirming that mean-centering preserves model fit. They produce identical predictions despite different coefficient interpretations.

2. **Model 3 shows dramatic decline:** R^2 drops to ≈ 0.0189 (only 1.89% of variance explained), demonstrating the critical importance of including main effects.

3. **Model 2 coefficients represent marginal effects at means:** The hp coefficient in Model 2 exactly equals the marginal effect of hp when wt equals its mean, calculated from Model 1.

4. **Statistical significance:**
   - Models 1 and 2: Highly significant ($p < 0.001$)
   - Model 3: Not statistically significant ($p \approx 0.454$)

5. **Practical interpretation:** The interaction alone explains almost nothing about mpg variation, while the full model (with main effects) explains nearly 88% of the variance in mpg.

6. **Main effects are crucial:** Removing them causes an 86+ percentage point drop in R^2, demonstrating that individual effects of hp and wt are far more important than their interaction alone.

Comparison with Models 1 and 2

- **Interaction coefficient:** The interaction term changes significantly across models:
  - Model 1: 0.02784815
  - Model 2: 0.02784815 (identical to Model 1)
  - Model 3: 0.01573639 (significantly different!)

  This shows that **removing main effects DOES impact the interaction coefficient**, contrary to what one might expect. This is because the interaction term in Model 3 is forced to capture both the true interaction effect AND compensate for the missing main effects.

- **Main effects:**
  - Models 1 and 2 include main effects for hp and wt. In Model 1 these are defined at unrealistic baselines (hp = 0, wt = 0), while in Model 2 they are defined at realistic baselines (mean hp, mean wt).

  - Model 3 excludes the main effects entirely, meaning the model assumes hp and wt only affect mpg through their joint interaction. This is an oversimplification that costs the R-squared value to drop drastically, barely reaching 2% (i.e. 2% of the variations in mpg can be explained by the joint interaction of horsepower and weight).

- **Intercept:**
  - Model 1: 49.80842 (predicted mpg when hp = 0 and wt = 0, not meaningful).
  - Model 2: 18.89840 (predicted mpg at mean hp and mean wt, interpretable).
  - Model 3: 19.41692 (close to Model 2, but adjusted upward because the main effects are missing).

Interpretation Model 3 preserves the interaction effect but ignores the independent contributions of horsepower and weight. While its intercept represents predicted mpg at average hp and wt (similar to Model 2), excluding main effects makes it a weaker and less realistic specification. Importantly, the interaction coefficient in Model 3 (0.01574) is different from Models 1 and 2 (0.02785), indicating that the interaction term is absorbing some of the effect that should be attributed to the main effects.

Part e) Mathematical explanation

See the following:

Model 1:

$$mpg = \beta_0 + \beta_1 h_p + \beta_2 w_t + \beta_3 (h_p \cdot w_t) + \epsilon$$

Model 2:
Let $(h_{pc} = hp - \bar{h_p})$ and $(w_{tc} = wt - \bar{w_t})$ be the mean-centered variables.

$$mpg = \beta_0 + \beta_1 h_{pc} + \beta_2 w_{tc} + \beta_3 (h_{pc} w_{tc}) + \epsilon$$

$$mpg = \beta_0 + \beta_1 (hp - \bar{h_p}) + \beta_2 (wt - \bar{w_t}) + \beta_3 ((hp - \bar{h_p})(wt - \bar{w_t})) + \epsilon$$

$$mpg = \beta_0 + \beta_1 h_p - \beta_1 \bar{h_p} + \beta_2 w_t - \beta_2 \bar{w}_t + \beta_3 h_p w_t - \beta_3 h_p \bar{w}_t - \beta_3 \bar{h_p} w_t + \beta_3 \bar{h_p} \bar{w}_t + \epsilon$$

$$mpg = h_p(\beta_1 - \beta_3 \bar{w}_t) + w_t(\beta_2 - \beta_3 \bar{h_p}) + \beta_0 - \beta_1 \bar{h_p} - \beta_2 \bar{w}_t + \beta_3 \bar{h_p} \bar{w}_t + \epsilon$$

(2)

Let's match coefficients of model 1 with model2 . Let's set betas in (1) to alphas instead and keep the current notation for equation (2).

$$\begin{aligned} \alpha_0 = \beta_0 - \beta_1 \bar{h_p} - \beta_2 \bar{w}_t + \beta_3 \bar{h_p} \bar{w}_t + \epsilon \\ \alpha_1 = \beta_1 - \beta_3 \bar{w}_t \\ \alpha_2 = \beta_2 - \beta_3 \bar{h_p} \\ \alpha_3 = \beta_3 \end{aligned}$$

Then we solve the above for the betas in terms of alphas:

$$\begin{aligned} \beta_0 = \alpha_0 + \alpha_1 \bar{h_p} + \alpha_2 \bar{w}_t + \alpha_3 \bar{h_p} \bar{w}_t - \epsilon \\ \beta_1 = \alpha_1 + \alpha_3 \bar{w}_t \\ \beta_2 = \alpha_2 + \alpha_3 \bar{h_p} \\ \beta_3 = \alpha_3 \end{aligned}$$

By identification we see that the coefficient of the interaction term remains the exact same from model 1 to model 2. $\alpha_3 = \beta_3$ with α_3 and β_3 being the exact same value. For the rest of the coefficients we simply observe that there is a change of basis, which is why we get the same R^2 value.

Model 3:
(3)

$$mpg = \beta_0 + \beta_3 h_p c w_t c + \epsilon$$

$$mpg = \beta_0 + \beta_3(h_p - \bar{h_p})(w_t - \bar{w}_t) + \epsilon$$

$$mpg = \beta_0 + (\beta_3 h_p - \beta_3 \bar{h_p})(w_t - \bar{w}_t) + \epsilon$$

$$mpg = \beta_0 + \beta_3 h_p w_t - \beta_3 h_p \bar{w}_t - \beta_3 \bar{h_p} w_t + \beta_3 \bar{h_p} \bar{w}_t + \epsilon$$

As we can see, the model 3 only contains 2 free parameters β_0 and β_3, lowering the overall flexibility and complexity of the model and decreasing R^2 value drastically.

Problem 4

Hindy Rossignol, Riya Parikh, Mrugank Pednekar, Ioannis Panagiotopoulos

2025-09-11

R setup

1. Load Data

```
# NOTE: Data files must be in a 'Data' subdirectory relative to this R Markdown file
# Expected structure:
#   - prob4.Rmd
#   - Data/
#     |- laptop_train.csv
#     |- laptop_test.csv
setwd(dirname(rstudioapi::getSourceEditorContext()$path))

# Read CSV files using relative paths
df_train <- read_csv("Data/laptop_train.csv")
```

```
## Rows: 665 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): Company, TypeName, GPU
## dbl (6): InventoryID, Screen, Memory, Weight, Rating, Price
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
df_test <- read_csv("Data/laptop_test.csv")
```

```
## Rows: 280 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): Company, TypeName, GPU
## dbl (6): InventoryID, Screen, Memory, Weight, Rating, Price
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

2. Data Exploration

```r
head(df_train)
```

```
## # A tibble: 6 x 9
##   InventoryID Company TypeName GPU    Screen Memory Weight Rating Price
##         <dbl> <chr>   <chr>    <chr>   <dbl>  <dbl>  <dbl>  <dbl> <dbl>
## 1           6 Asus    Gaming   Nvidia   17.3     16   2.9        1 2122
## 2          15 Asus    Gaming   Nvidia   17.3     16   2.73       3 2050.
## 3          17 Asus    Gaming   Nvidia   15.6     16   2.5        4 1799
## 4          18 Asus    Gaming   Nvidia   17.3     16   4         10  998
## 5          38 Asus    Gaming   Nvidia   15.6      8   2.3        6 1649
## 6          40 Asus    Gaming   Nvidia   17.3      8   3          9 1168
```

```r
head(df_test)
```

```
## # A tibble: 6 x 9
##   InventoryID Company TypeName GPU    Screen Memory Weight Rating Price
##         <dbl> <chr>   <chr>    <chr>   <dbl>  <dbl>  <dbl>  <dbl> <dbl>
## 1           1 Asus    Gaming   Nvidia   15.6     16   2.2        8 1449
## 2          29 Asus    Gaming   AMD      15.6      8   2.45      10  699
## 3          30 Asus    Gaming   Nvidia   17.3      8   3          7  938
## 4          35 Asus    Gaming   Nvidia   17.3      8   3          5 1039
## 5          56 Asus    Notebook Intel    15.6      4   2.37       6  399.
## 6          62 Asus    Notebook Intel    15.6      4   2          5  559
```

```r
# We want to know the dimensions of our dataset.
dim(df_train) # there are 9 features and 665 observations
```

```
## [1] 665   9
```

```r
dim(df_test)  # there are 9 features and 280 observations
```

```
## [1] 280   9
```

```r
str(df_train) # structure (variable types, first few entries)
```

```
## spc_tbl_ [665 x 9] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ InventoryID: num [1:665] 6 15 17 18 38 40 41 45 46 47 ...
##  $ Company    : chr [1:665] "Asus" "Asus" "Asus" "Asus" ...
##  $ TypeName   : chr [1:665] "Gaming" "Gaming" "Gaming" "Gaming" ...
##  $ GPU        : chr [1:665] "Nvidia" "Nvidia" "Nvidia" "Nvidia" ...
##  $ Screen     : num [1:665] 17.3 17.3 15.6 17.3 15.6 17.3 15.6 15.6 15.6 15.6 ...
##  $ Memory     : num [1:665] 16 16 16 16 8 8 8 8 8 16 ...
##  $ Weight     : num [1:665] 2.9 2.73 2.5 4 2.3 ...
##  $ Rating     : num [1:665] 1 3 4 10 6 9 4 6 8 4 ...
##  $ Price      : num [1:665] 2122 2050 1799 998 1649 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   InventoryID = col_double(),
##   ..   Company = col_character(),
```

```
##   ..   TypeName = col_character(),
##   ..   GPU = col_character(),
##   ..   Screen = col_double(),
##   ..   Memory = col_double(),
##   ..   Weight = col_double(),
##   ..   Rating = col_double(),
##   ..   Price = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```r
str(df_test)  # structure (variable types, first few entries)
```

```
## spc_tbl_ [280 x 9] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ InventoryID: num [1:280] 1 29 30 35 56 62 143 146 158 160 ...
##  $ Company    : chr [1:280] "Asus" "Asus" "Asus" "Asus" ...
##  $ TypeName   : chr [1:280] "Gaming" "Gaming" "Gaming" "Gaming" ...
##  $ GPU        : chr [1:280] "Nvidia" "AMD" "Nvidia" "Nvidia" ...
##  $ Screen     : num [1:280] 15.6 15.6 17.3 17.3 15.6 15.6 15.6 15.6 15.6 15.6 ...
##  $ Memory     : num [1:280] 16 8 8 8 4 4 16 16 8 8 ...
##  $ Weight     : num [1:280] 2.2 2.45 3 3 2.37 2 3.49 2.62 2.62 2.62 ...
##  $ Rating     : num [1:280] 8 10 7 5 6 5 3 1 7 4 ...
##  $ Price      : num [1:280] 1449 699 938 1039 399 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   InventoryID = col_double(),
##   ..   Company = col_character(),
##   ..   TypeName = col_character(),
##   ..   GPU = col_character(),
##   ..   Screen = col_double(),
##   ..   Memory = col_double(),
##   ..   Weight = col_double(),
##   ..   Rating = col_double(),
##   ..   Price = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```r
summary(df_train) # summary statistics by column
```

```
##   InventoryID      Company            TypeName             GPU
##  Min.   :  2.0   Length:665         Length:665         Length:665
##  1st Qu.:226.0   Class :character   Class :character   Class :character
##  Median :461.0   Mode  :character   Mode  :character   Mode  :character
##  Mean   :461.9
##  3rd Qu.:693.0
##  Max.   :945.0
##      Screen          Memory            Weight          Rating
##  Min.   :12.50   Min.   : 4.000   Min.   :0.91   Min.   : 1.000
##  1st Qu.:14.00   1st Qu.: 4.000   1st Qu.:1.70   1st Qu.: 3.000
##  Median :15.60   Median : 8.000   Median :2.06   Median : 5.000
##  Mean   :15.21   Mean   : 7.829   Mean   :2.09   Mean   : 5.402
##  3rd Qu.:15.60   3rd Qu.: 8.000   3rd Qu.:2.30   3rd Qu.: 8.000
##  Max.   :17.30   Max.   :16.000   Max.   :4.60   Max.   :10.000
##      Price
```

```
##  Min.   : 224
##  1st Qu.: 589
##  Median : 899
##  Mean   :1027
##  3rd Qu.:1280
##  Max.   :3154
```

```r
summary(df_test)  # summary statistics by column
```

```
##   InventoryID      Company            TypeName              GPU
##  Min.   :  1.0   Length:280         Length:280         Length:280
##  1st Qu.:254.0   Class :character   Class :character   Class :character
##  Median :491.5   Mode  :character   Mode  :character   Mode  :character
##  Mean   :499.3
##  3rd Qu.:750.8
##  Max.   :944.0
##      Screen          Memory           Weight          Rating
##  Min.   :12.50   Min.   : 4.000   Min.   :0.990   Min.   : 1.000
##  1st Qu.:14.00   1st Qu.: 4.000   1st Qu.:1.700   1st Qu.: 3.000
##  Median :15.60   Median : 8.000   Median :2.040   Median : 5.000
##  Mean   :15.24   Mean   : 7.486   Mean   :2.053   Mean   : 5.354
##  3rd Qu.:15.60   3rd Qu.: 8.000   3rd Qu.:2.300   3rd Qu.: 8.000
##  Max.   :17.30   Max.   :16.000   Max.   :4.600   Max.   :10.000
##      Price
##  Min.   : 274.9
##  1st Qu.: 598.7
##  Median : 897.5
##  Mean   : 998.5
##  3rd Qu.:1268.0
##  Max.   :2999.0
```

```r
colSums(is.na(df_train))   # number of NAs per column
```

```
## InventoryID     Company    TypeName         GPU      Screen      Memory
##           0           0           0           0           0           0
##      Weight      Rating       Price
##           0           0           0
```

```r
anyNA(df_train)            # check if dataset has any missing values
```

```
## [1] FALSE
```

3. Create Binary Target Variable

We are creating a new binary column high for both the test and train dataframes which is 1 if the price is 500 Euros or higher, and 0 otherwise If a laptop's price is $\geq$ 500 Euros, it gets high = 1 (expensive). If it's $\leq$ 499.99 Euros, it gets high = 0 (not expensive).

```r
# Add binary column 'high' to training and test datasets
df_train <- df_train %>%
  mutate(high = ifelse(Price >= 500, 1, 0))
```

```r
df_test <- df_test %>%
  mutate(high = ifelse(Price >= 500, 1, 0))

# Check the distribution of the new binary variable in training set
table(df_train$high)
```

```
##
##   0   1
## 115 550
```

```r
prop.table(table(df_train$high))
```

```
##
##         0         1
## 0.1729323 0.8270677
```

```r
# Check distribution in test set
table(df_test$high)
```

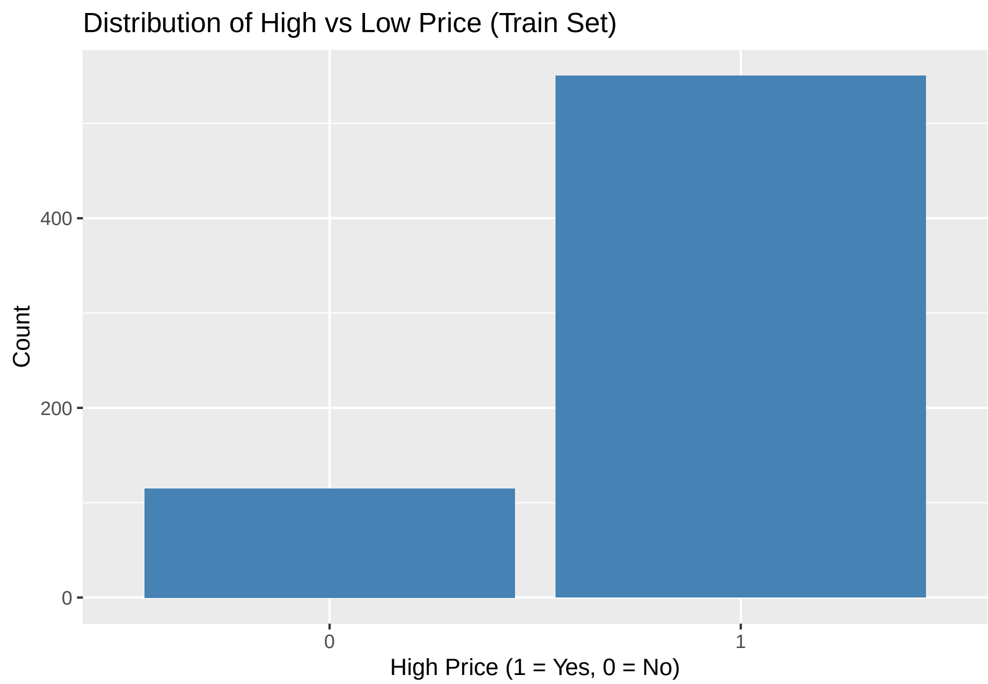
```
##
##   0   1
##  53 227
```

```r
prop.table(table(df_test$high))
```

```
##
##         0         1
## 0.1892857 0.8107143
```

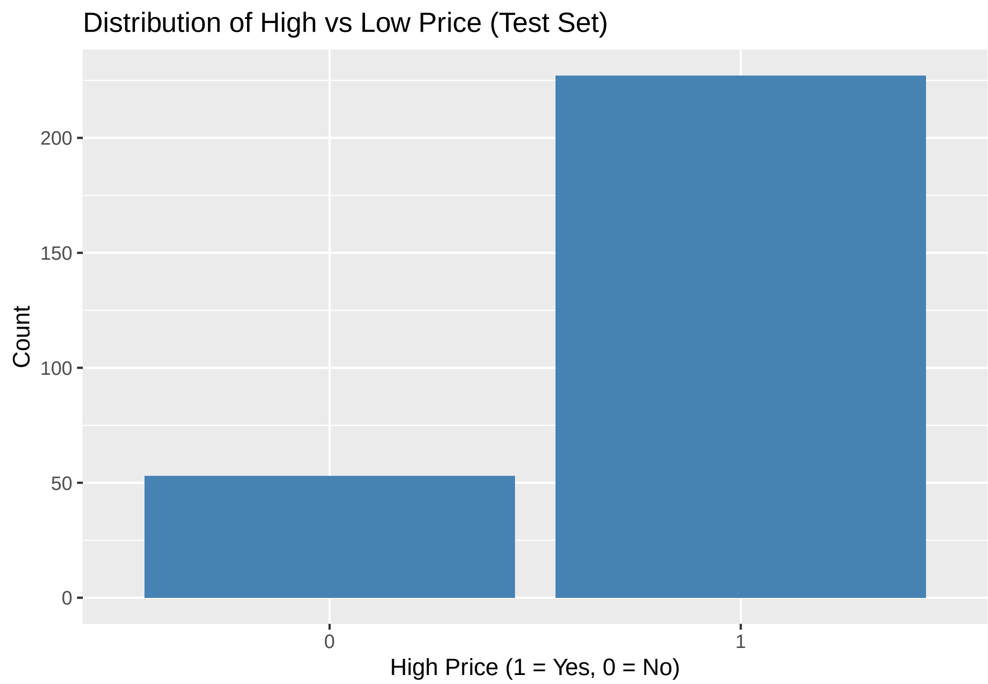
The following plot shows the count of laptops in the training set that are classified as high price (1) or low price (0). The bars are colored steel blue for better visualization.

```r
# Bar plot of high/low price distribution in training set
ggplot(df_train, aes(factor(high))) +
  geom_bar(fill = "steelblue") +
  labs(title = "Distribution of High vs Low Price (Train Set)",
       x = "High Price (1 = Yes, 0 = No)", y = "Count")
```



We do the same for the test set.

```
# Bar plot of high/low price distribution in test set
ggplot(df_test, aes(factor(high))) +
  geom_bar(fill = "steelblue") +
  labs(title = "Distribution of High vs Low Price (Test Set)",
       x = "High Price (1 = Yes, 0 = No)", y = "Count")
```

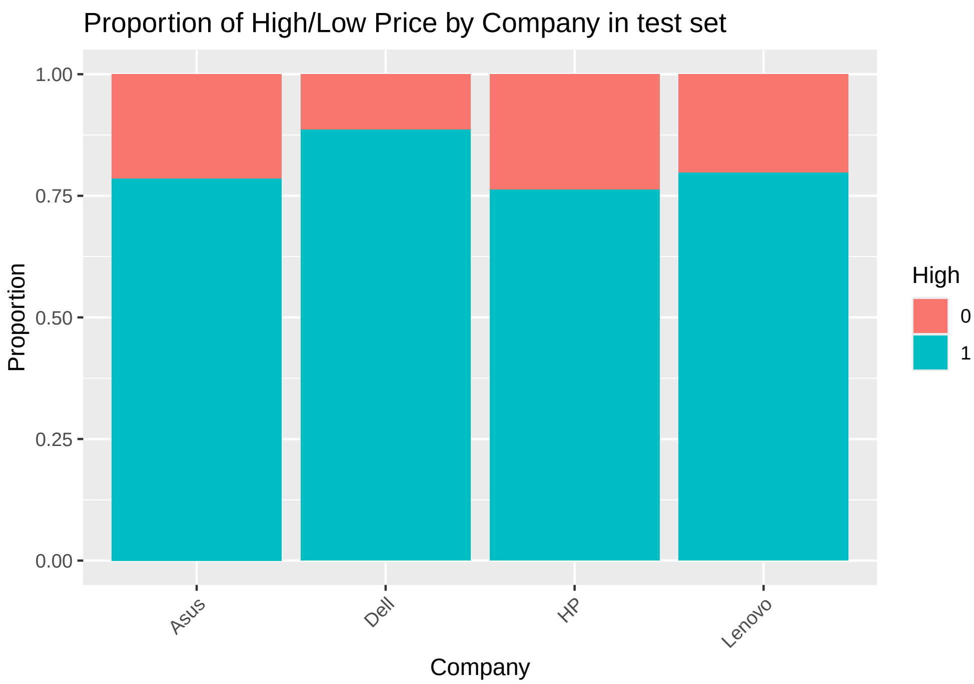


From the above plots, we can see that the classes are inbalanced, with more high price laptops (1) than low price laptops (0). The following plot shows the proportion of high price (1) and low price (0) laptops for each company in the training set. Each bar is stacked and filled by the 'high' variable: different colors for high (1) and low (0).

```
# Bar plot: Company vs High/Low Price in training set

ggplot(df_train, aes(x = Company, fill = factor(high))) +
  geom_bar(position = "fill") +
  labs(title = "Proportion of High/Low Price by Company in train set",
        y = "Proportion", fill = "High") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

We are doing the same for the test set.

```
# Bar plot: Company vs High/Low in test set
ggplot(df_test, aes(x = Company, fill = factor(high))) +
  geom_bar(position = "fill") +
  labs(title = "Proportion of High/Low Price by Company in test set",
        y = "Proportion", fill = "High") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Based on the above plots, we can see whether some manufacturers (Dell, HP, Lenovo, Asus) are more likely to sell high-priced laptops. It just gives us an intuition before we build our model.

[Optional] Make the dataset balanced

Here we will use caret package to upsample the minority class (low price laptops) in the training set to make the classes balanced. Since it is not asked, we will just present it here and continue with the original unbalanced dataset for modeling.

```r
# Make copies of the original datasets
df_train_orig <- df_train
df_test_orig  <- df_test

# Set seed for reproducibility
set.seed(123)

# Convert 'high' to factor for classification
df_train_orig$high <- factor(df_train_orig$high, levels = c(0,1), labels = c("low","high"))
df_test_orig$high  <- factor(df_test_orig$high,  levels = c(0,1), labels = c("low","high"))

# Upsample the minority class (low price laptops)
df_train_orig_balanced <- upSample(x = subset(df_train_orig, select = -high),
                                   y = df_train_orig$high,
                                   yname = "high")

# Convert 'high' back to 0/1 for easier analysis
df_train_orig_balanced$high <- ifelse(df_train_orig_balanced$high == "high", 1, 0)

# Check the distribution of the new binary variable in balanced training set
table(df_train_orig_balanced$high)
```

```
##
##   0   1
## 550 550
```

```r
prop.table(table(df_train_orig_balanced$high))
```

```
##
##   0   1
## 0.5 0.5
```

Problem 4

Question (a) Build Model

```r
# Build logistic regression model
# Dependent variable: 'high' (binary outcome: 1 = high price, 0 = low price)
# Independent variables: InventoryID, Company, TypeName, GPU, Screen, Memory,
# Weight, Rating
# We include all predictors in the model without performing variable selection
# Build logistic regression model
logistic_model <- glm(high ~ InventoryID + Company + TypeName + GPU + Screen +
                     Memory + Weight + Rating,
                     data = df_train,
                     family = "binomial")

# Show model summary
# This outputs coefficient estimates, significance levels,
# and model fit statistics
summary(logistic_model)
```

```
##
## Call:
## glm(formula = high ~ InventoryID + Company + TypeName + GPU +
##     Screen + Memory + Weight + Rating, family = "binomial", data = df_train)
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        2.647e+01  8.091e+02   0.033 0.973901
## InventoryID        5.577e-04  9.390e-04   0.594 0.552571
## CompanyDell        1.184e+00  5.108e-01   2.318 0.020473 *
## CompanyHP          1.309e+00  5.531e-01   2.367 0.017912 *
## CompanyLenovo     -3.677e-01  6.915e-01  -0.532 0.594867
## TypeNameNotebook  -1.464e+01  8.091e+02  -0.018 0.985562
## TypeNameUltrabook -1.326e+01  8.091e+02  -0.016 0.986925
## GPUIntel          -1.077e-01  3.544e-01  -0.304 0.761111
## GPUNvidia          2.031e+00  6.074e-01   3.344 0.000826 ***
## Screen            -1.198e+00  3.121e-01  -3.839 0.000124 ***
## Memory             7.337e-01  9.354e-02   7.843 4.39e-15 ***
## Weight             1.560e+00  9.069e-01   1.720 0.085373 .
## Rating            -9.092e-02  4.924e-02  -1.846 0.064823 .
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 612.47  on 664  degrees of freedom
## Residual deviance: 334.40  on 652  degrees of freedom
## AIC: 360.4
##
## Number of Fisher Scoring iterations: 18
```

Question (b) Which variables are significant in predicting the probability of a price's being high?

```r
# Which variables are significant in predicting the probability of a
# price's being high? Variables with p-values less than 0.05 are considered
# statistically significant
significant_vars <- summary(logistic_model)$coefficients
significant_vars[significant_vars[,4] < 0.05, ]
```

```
##               Estimate Std. Error   z value     Pr(>|z|)
## CompanyDell  1.1838020 0.51079418  2.317571 2.047263e-02
## CompanyHP    1.3094383 0.55310260  2.367442 1.791153e-02
## GPUNvidia    2.0311573 0.60738742  3.344089 8.255339e-04
## Screen      -1.1981472 0.31212375 -3.838693 1.236912e-04
## Memory       0.7336737 0.09354129  7.843313 4.388113e-15
```

Interpretation of Results The p-value is the probability of observing a sample with results as extreme as, or more extreme than, the observed data, assuming the null hypothesis is true. A lower p-value indicates stronger evidence against the null hypothesis. In this analysis, we use a threshold of 0.05, meaning variables with p-values below this level are considered statistically significant.

Significant predictors ($p < 0.05$):

- CompanyDell ($p = 0.020$) → Dell laptops are more likely to be high-priced vs. Asus.
- CompanyHP ($p = 0.018$) → HP laptops are also more likely to be high-priced.
- GPUNvidia ($p < 0.001$) → Nvidia GPUs strongly increase the likelihood of high-priced laptops.
- Screen ($p < 0.001$) → Larger screen size reduces the probability of being high-priced.
- Memory ($p < 0.001$) → More RAM strongly increases the likelihood of being high-priced.

Not significant predictors ($p \geq 0.05$):

- Weight
- Rating
- InventoryID

- CompanyLenovo
- TypeName
- GPUIntel

Interpretation:

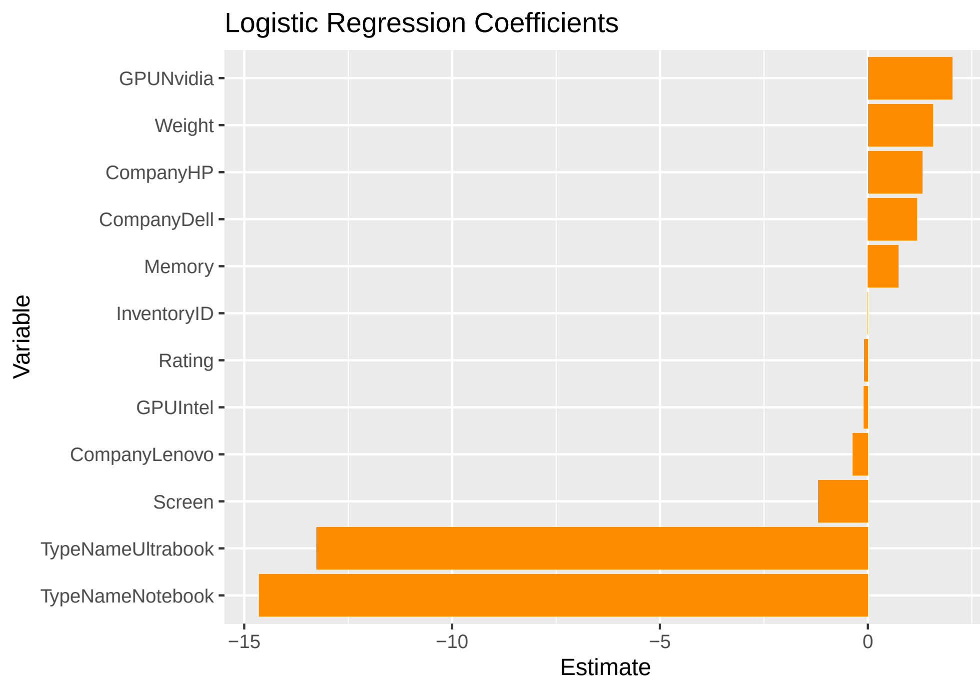
Practical factors influencing price:
- More RAM (+), Nvidia GPU (+), Dell/HP branding (+) increase odds of being expensive.
- Larger screens (-) reduce odds (perhaps because gaming/ultrabooks with smaller but more powerful components are pricey).

Baseline/reference categories:
- Company baseline = Asus.
- Type baseline = Gaming.
- GPU baseline = AMD.

So, coefficients are relative to these baselines.

Visualize Coefficients (optional) Visualize model coefficients (excluding intercept). This horizontal bar plot shows the estimated coefficients from the logistic regression model (excluding the intercept). Each bar represents a variable, and the length and direction indicate the effect size and sign.

```r
# Bars are colored dark orange for better visibility.
coefs_df <- as.data.frame(summary(logistic_model)$coefficients)
coefs_df$Variable <- rownames(coefs_df)
coefs_df <- coefs_df[coefs_df$Variable != "(Intercept)", ]

ggplot(coefs_df, aes(x = reorder(Variable, Estimate), y = Estimate)) +
  geom_bar(stat = "identity", fill = "darkorange") +
  coord_flip() +
  labs(title = "Logistic Regression Coefficients", x = "Variable", y = "Estimate")
```



Question (c) Comparison with the linear regression model

Logistic Regression Model We give a short summary of the logistic regression model here:

- Significant predictors ($p < 0.05$) included: **Company (Dell, HP), GPU (Nvidia), Screen size, and Memory**.
- Results suggest that **Dell/HP branding, more RAM, and Nvidia GPUs** increase the odds of a laptop being high-priced, while **larger screen sizes reduce** those odds.
- Baseline categories are: **Asus (Company), Gaming (TypeName), AMD (GPU)**. Coefficients for other categories are interpreted relative to these baselines.

Linear Regression Model The linear regression model directly predicts **laptop price** as a continuous outcome.

Key results:

- **Screen size (-36.60, p = 0.021):** Larger screens are associated with slightly lower prices.
- **Memory (+90.21, p < 0.001):** Each additional GB of RAM increases price on average by about €90.
- **Rating (-12.82, p = 0.0067):** Higher ratings are associated with slightly lower prices.
- **CompanyDell (+124.91, p = 0.004) and CompanyHP (+175.31, p < 0.001):** Dell and HP laptops are priced higher compared to Asus.

- **TypeNameNotebook (-234.46, $p < 0.001$):** Notebooks are significantly cheaper than Gaming laptops.
- **TypeNameUltrabook (+203.68, $p = 0.0019$):** Ultrabooks are significantly more expensive than Gaming laptops.
- **GPUIntel (+163.64, $p < 0.001$) and GPUNvidia (+227.60, $p < 0.001$):** Both Intel and Nvidia GPUs increase price compared to AMD GPUs.

Comparison and Insights

- Logistic regression provides a binary view (is the laptop high-priced or not), while linear regression quantifies *how much* each factor contributes to price in Euros.
- Both models consistently highlight **Memory** and **GPU type** as important price drivers.
- **Company (Dell, HP)** is significant in both models, reinforcing the strong effect of branding on price.
- Linear regression suggests **Ultrabook type** is also strong price determinant, which is insignificant in the logistic model.
- The negative effect of **Screen size** appears in both models, reinforcing that larger screens may not always mean higher-priced laptops (likely due to mid-range notebooks with larger but less powerful builds).

Question (d) Interpretation of Significant Variables

Variable	Effect on Probability of Being High-Priced	Possible Explanation
CompanyDell	Increases → Dell laptops are more likely to be high-priced compared to Asus.	Dell often offers premium business and gaming models at higher prices.
CompanyHP	Increases → HP laptops are more likely to be high-priced compared to Asus.	HP has strong enterprise and premium product lines.
GPUNvidia	Increases → Laptops with Nvidia GPUs are more likely to be high-priced compared to AMD.	Nvidia GPUs are powerful and common in high-end gaming/professional laptops.
Screen	Decreases → Larger screen size reduces the likelihood of being high-priced.	Counterintuitive, but many premium laptops (e.g., gaming/ultrabooks) favor performance/portability over large displays.
Memory	Increases → More RAM increases the likelihood of being high-priced.	Higher RAM is directly linked to better performance.

Question (e) Comparison of coefficient signs between models

When comparing the logistic regression (high-priced vs. low-priced) with the linear regression (continuous price), most predictors show consistent directions of effect, while a few differ.

Same sign in both models:

- **Memory:** Positive in both → more RAM increases the likelihood of being high-priced and also raises the price level in Euros.
- **GPUNvidia:** Positive in both → Nvidia GPUs are associated with higher odds of being expensive and increase absolute price.
- **CompanyDell / CompanyHP:** Positive in both → Dell and HP branding raises both the odds of being high-priced and the absolute price compared to Asus.
- **Screen:** Negative in both → larger screens reduce odds of being high-priced and are associated with lower prices.

Different signs between models:

- **GPUIntel:** Negative in logistic regression (though not significant) but positive in linear regression → Intel GPUs are linked with slightly lower odds of being in the high-price group, yet contribute positively to price as a continuous outcome.
- **TypeNameUltrabook:** Negative in logistic regression (not significant) but strongly positive in linear regression → Ultrabooks are not clearly more likely to cross the 500 price threshold, but when they do, their absolute prices are much higher.
- **TypeNameNotebook:** Negative in both, but only significant in linear regression.

Key takeaway

- The **consistent predictors across both models** are **Memory, Nvidia GPUs, Weight, and Screen size**, all showing the same directional effect.
- The **main divergences** are for **Intel GPUs and Ultrabook type**, where logistic and linear models disagree. This suggests that these features influence *absolute pricing levels* but may not cleanly separate laptops into high- vs. low-price categories.

Question (f) Predicting for a Specific Laptop

We want to predict the probability that a given laptop is **high-priced** ($\geq$ 500 Euros).
The logistic regression model gives this probability using the logistic function:

$$P(\text{high} = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k)}}$$

Given Laptop Characteristics

- InventoryID = 4096
- Company = Lenovo
- TypeName = Ultrabook
- GPU = Intel

- Screen = 8
- Memory = 8
- Weight = 4.2
- Rating = 7

Step 1: Extract Coefficients From the logistic regression model:

- (Intercept) = 26.47131
- InventoryID = 0.0005577
- CompanyLenovo = -0.3677
- TypeNameUltrabook = -13.25963
- GPUIntel = -0.1077
- Screen = -1.1981
- Memory = 0.7337
- Weight = 1.5602
- Rating = -0.0909

(Other coefficients are not included since they do not apply to this laptop's profile.)

Step 2: Compute the Logit (Z)

$$\begin{aligned} Z = \beta_0 + \beta_1(\text{InventoryID}) + \beta_{\text{Lenovo}} + \beta_{\text{Ultrabook}} + \beta_{\text{Intel}} \\ + \beta_{\text{Screen}} \cdot \text{Screen} + \beta_{\text{Memory}} \cdot \text{Memory} + \beta_{\text{Weight}} \cdot \text{Weight} + \beta_{\text{Rating}} \cdot \text{Rating}. \end{aligned}$$

Substituting the values:

$$\begin{gathered} Z = 26.47131 + 0.0005577 \times 4096 - 0.3677 - 13.25963 - 0.1077 \\ - 1.1981 \times 8 + 0.7337 \times 8 + 1.5602 \times 4.2 - 0.0909 \times 7. \end{gathered}$$

$$Z \approx 22.9$$

Step 3: Apply Logistic Function

$$P(\text{high} = 1) = \frac{1}{1 + e^{-Z}} = \frac{1}{1 + e^{-22.9}} \approx 1$$

Final Prediction

- **Manual calculation:** Probability $\approx$ **1.0000**
- **Using `predict()` in R:** Probability $\approx$ **1.0000**

Explanation of Implementation

1. Created a new observation (`new_laptop`) with the given laptop's features.
2. Extracted coefficients from the fitted logistic regression model.
3. Calculated the logit (Z) by plugging in the observation's values.
4. Applied the logistic function to transform Z into a probability.
5. Validated with R's `predict()` function, which confirmed the manual result.

The model predicts with near certainty that this Lenovo Ultrabook would be **high-priced**.

```r
# Create a new observation
new_laptop <- data.frame(
  InventoryID = 4096,
  Company = "Lenovo",
  TypeName = "Ultrabook",
  GPU = "Intel",
  Screen = 8,
  Memory = 8,
  Weight = 4.2,
  Rating = 7
)

# Show the equation for probability:
# pr(high = 1) = 1 / (1 + exp(-(B0 + B1*InventoryID + ... + B7*Rating)))
coefs <- coef(logistic_model)
coefs
```

```
##       (Intercept)       InventoryID       CompanyDell         CompanyHP
##      2.647131e+01      5.576561e-04      1.183802e+00      1.309438e+00
##     CompanyLenovo  TypeNameNotebook TypeNameUltrabook          GPUIntel
##     -3.677291e-01     -1.464189e+01     -1.325963e+01     -1.077326e-01
##         GPUNvidia            Screen            Memory            Weight
##      2.031157e+00     -1.198147e+00      7.336737e-01      1.560218e+00
##            Rating
##     -9.092252e-02
```

```r
# Calculate logit (Z) manually
Z <- coefs["(Intercept)"] +
     coefs["InventoryID"] * new_laptop$InventoryID +
     coefs[paste0("Company", new_laptop$Company)] +
     coefs[paste0("TypeName", new_laptop$TypeName)] +
     coefs[paste0("GPU", new_laptop$GPU)] +
     coefs["Screen"] * new_laptop$Screen +
     coefs["Memory"] * new_laptop$Memory +
     coefs["Weight"] * new_laptop$Weight +
     coefs["Rating"] * new_laptop$Rating

# Calculate probability
prob <- 1 / (1 + exp(-Z))
print(paste("Predicted probability (manual):", round(prob, 4)))
```

```
## [1] "Predicted probability (manual): 1"
```

```r
# Or use predict()
prob_predict <- predict(logistic_model, newdata = new_laptop, type = "response")
print(paste("Predicted probability (predict):", round(prob_predict, 4)))
```

```
## [1] "Predicted probability (predict): 1"
```

Question (g) Model Evaluation on Test Set

```r
# Generate predictions on test set
test_predictions <- predict(logistic_model, newdata = df_test, type = "response")

# Convert probabilities to binary predictions using 0.5 cutoff
predicted_classes <- ifelse(test_predictions >= 0.5, 1, 0)

# Calculate accuracy
accuracy <- mean(predicted_classes == df_test$high)

# Create confusion matrix
conf_matrix <- table(Predicted = predicted_classes, Actual = df_test$high)

# Display results
print("Confusion Matrix:")
```

```
## [1] "Confusion Matrix:"
```

```r
print(conf_matrix)
```

```
##          Actual
## Predicted   0   1
##         0  29  16
##         1  24 211
```

```r
print(paste("Accuracy:", round(accuracy * 100, 2), "%"))
```

```
## [1] "Accuracy: 85.71 %"
```

We applied the logistic regression model to the **test dataset** and evaluated performance using a **0.5 probability cutoff**.

- **True Negatives (TN):** 29
- **False Negatives (FN):** 16
- **False Positives (FP):** 24
- **True Positives (TP):** 211

Accuracy

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

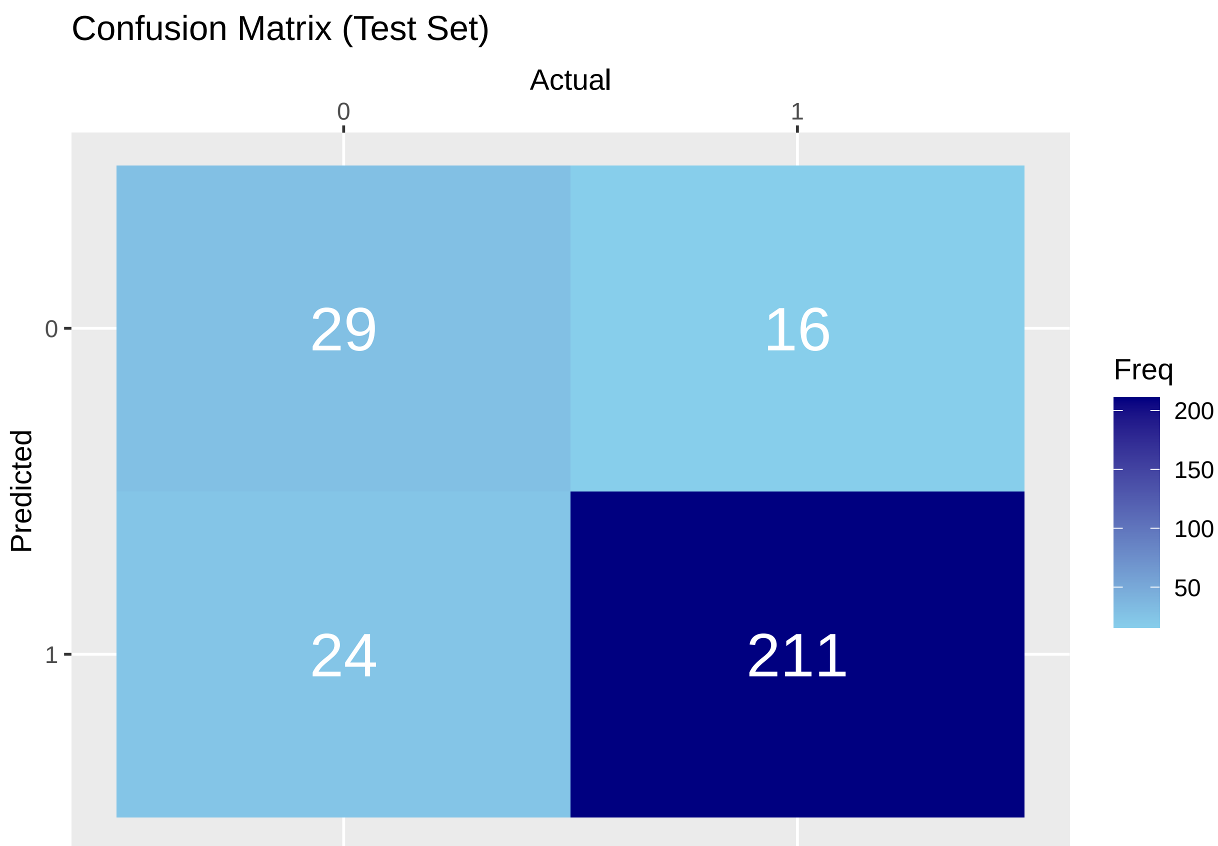
$$\text{Accuracy} = \frac{211 + 29}{211 + 29 + 24 + 16} = \frac{240}{280} \approx 85.71\%$$

Interpretation

- The model achieves an **accuracy of 85.71%** on the test dataset.
- Most high-priced laptops (class = 1) were correctly identified (211 true positives), showing the model is strong at predicting expensive laptops.
- Some errors occur in predicting low-priced laptops, as seen in the 24 false positives and 16 false negatives.

Visualization The following heatmap provides a visual representation of the confusion matrix for the test set predictions. This heatmap shows the confusion matrix for the test set predictions. The fill color (from sky blue to navy) indicates the number of laptops in each cell (Predicted vs Actual).

```r
# Visualize confusion matrix as heatmap
cm_df <- as.data.frame(conf_matrix)
colnames(cm_df) <- c("Predicted", "Actual", "Freq")
cm_df$Predicted <- factor(cm_df$Predicted, levels = c(1, 0))
cm_df$Actual <- factor(cm_df$Actual, levels = c(0, 1))

ggplot(cm_df, aes(x = Actual, y = Predicted, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq), color = "white", size = 8) +
  scale_fill_gradient(low = "skyblue", high = "navy") +
  labs(
    title = "Confusion Matrix (Test Set)",
    x = "Actual",
    y = "Predicted"
  ) +
  scale_x_discrete(position = "top") # Put Actual labels on top to match table
```


Confusion Matrix (Test Set)
Actual
0
1
Predicted
0
1
29
16
24
211
Freq
200
150
100
50