Problem 4

Hindy Rossignol, Riya Parikh, Mrugank Pednekar, Ioannis Panagiotopoulos

2025-09-11

R setup

1. Load Data

```r
# 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.
```

```r
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)
```

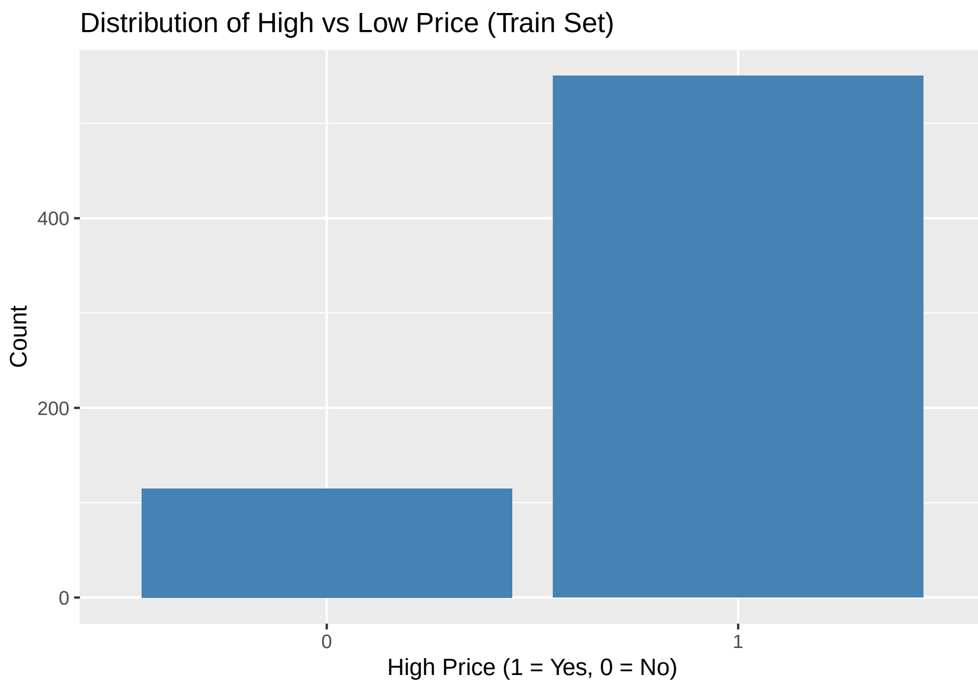
```
##
##   0   1
##  53 227
```

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

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

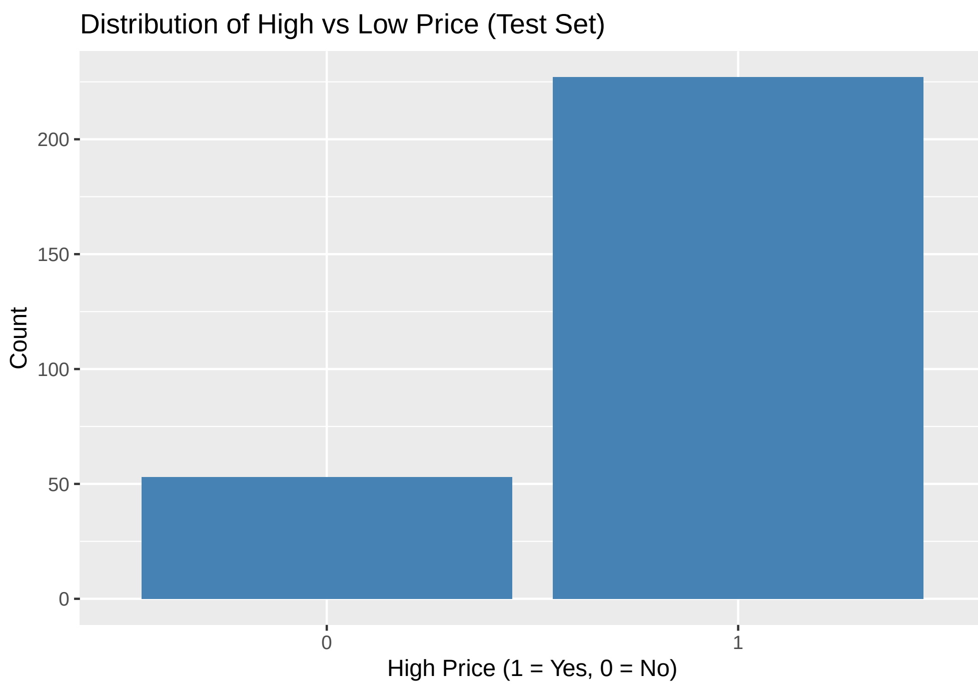
The following plot shows the count of laptops in the training set that are classified as high price (1) or low price (0). The bars are colored steel blue for better visualization.

```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")
```



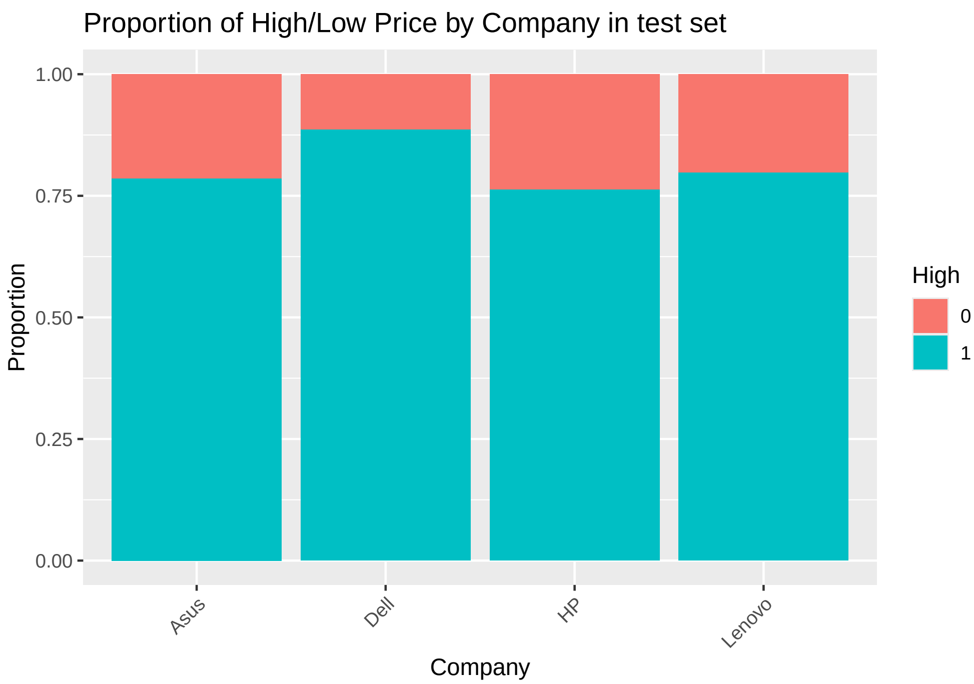
From the above plots, we can see that the classes are inbalanced, with more high price laptops (1) than low price laptops (0). The following plot shows the proportion of high price (1) and low price (0) laptops for each company in the training set. Each bar is stacked and filled by the 'high' variable: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

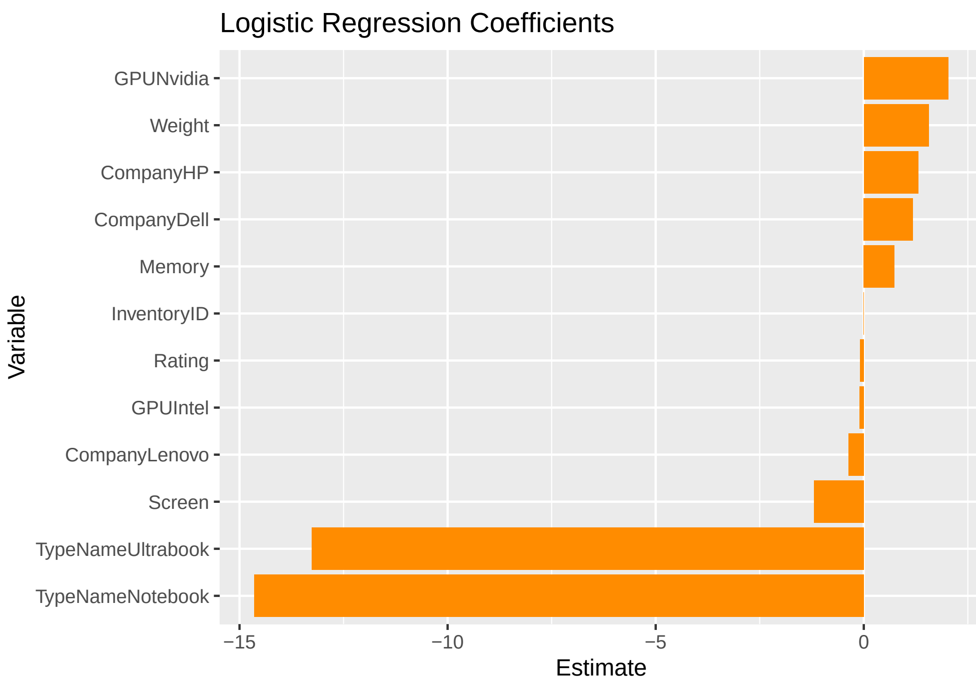
Baseline/reference categories:
- Company baseline = Asus.
- Type baseline = Gaming.
- GPU baseline = AMD.

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.

```
# 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 neither, so practical impact is limited.

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:

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

$$Z \approx 22.9$$

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**.

```
# 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
```

```
# 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

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

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

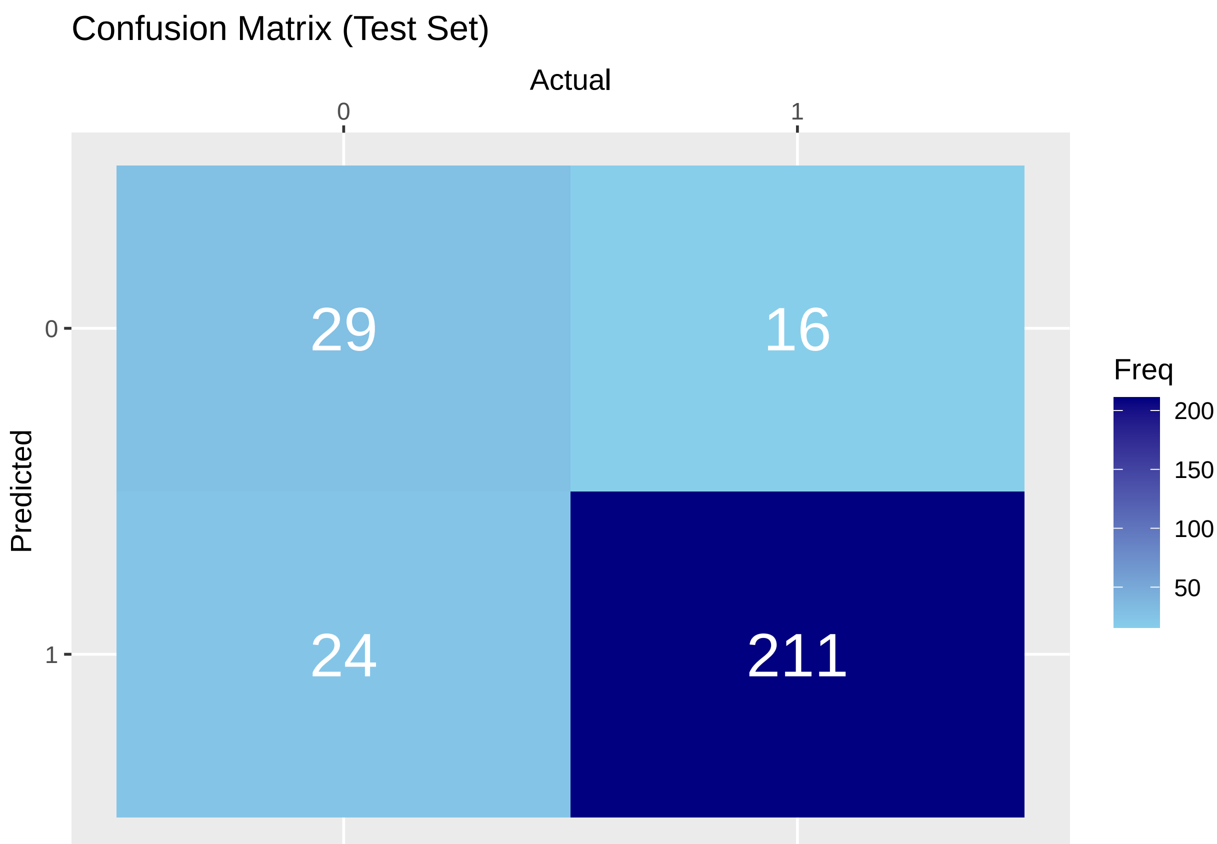
Interpretation

- 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