HarvardX PH125.9x Capstone - Used Car Data

Aditya Shah

Overview/Executive summary

The goal of the project is to train models using machine learning algorithms to predict if a particular car would be sold or unsold based on certain variables or predictors.

A dataset containing the records of nearly 8000 cars in the US and their sold status is used to train the models. For each car the dataset includes details (variables) such as its brand, selling price, transmission type, location, etc.

Variable/s not necessary for this project are removed to reduce the size of the dataset. The variables in the dataset are then made ready for data analysis and model training.

The suitability of the variables as predictors for the models is considered through data exploration, visualization and analysis leading to a shortlist of variables to use.

The shortlisted variables are used to train logistic regression models individually and collectively. A classification tree model is then trained using the variable/s from the best performing logistic regression model.

The dataset (version 1) was obtained here: https://www.kaggle.com/datasets/shubham1kumar/usedcar-data/versions/1 (https://www.kaggle.com/datasets/shubham1kumar/usedcar-data/versions/1)

Introduction

This project is part of the PH125.9x Capstone course of the HarvardX Data Science Professional Certificate program where learners are tasked with training models using machine learning algorithms on a dataset of their own choosing.

This report documents the training of models to predict if a particular car would be sold or not based on certain variables and is based on a dataset of used cars sales obtained from Kaggle.com.

This report is split into three sections:

Section 1 covers the initial exploration and manipulation of the dataset along with detailed analysis of the variables to assess their suitability to train the models.

Section 2 covers the training of the models and provides an overview of the accuracy results.

Section 3 contains the concluding remarks.

1. Data Exploration and Analysis

1.1 Initial exploration and manipulation

The usedcardata dataset is loaded:

```
data <- read_csv("UsedCarData.csv")
```

A quick look at the dataset shows that there are 7906 records with 18 variables containing details such as the car's brand, selling price, fuel type, location, etc:

```
glimpse(data)
```

```
## Rows: 7,906
## Columns: 18
## $ Sales_ID            <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16,~
## $ name                <chr> "Maruti", "Skoda", "Honda", "Hyundai", "Maruti", "~
## $ year                <dbl> 2014, 2014, 2006, 2010, 2007, 2017, 2007, 2001, 20~
## $ selling_price       <dbl> 450000, 370000, 158000, 225000, 130000, 440000, 96~
## $ km_driven           <dbl> 145500, 120000, 140000, 127000, 120000, 45000, 175~
## $ Region              <chr> "East", "East", "Central", "Central", "East", "Eas~
## $ `State or Province` <chr> "District of Columbia", "New York", "Illinois", "I~
## $ City                <chr> "Washington", "New York City", "Chicago", "Chicago~
## $ fuel                <chr> "Diesel", "Diesel", "Petrol", "Diesel", "Petrol", ~
## $ seller_type         <chr> "Individual", "Individual", "Individual", "Individ~
## $ transmission        <chr> "Manual", "Manual", "Manual", "Manual", "Manual", ~
## $ owner               <chr> "First_Owner", "Second_Owner", "Third_Owner", "Fir~
## $ mileage             <dbl> 23.40, 21.14, 17.70, 23.00, 16.10, 20.14, 17.30, 1~
## $ engine              <dbl> 1248, 1498, 1497, 1396, 1298, 1197, 1061, 796, 136~
## $ max_power           <dbl> 74.00, 103.52, 78.00, 90.00, 88.20, 81.86, 57.50, ~
## $ torque              <chr> "190Nm@ 2000rpm", "250Nm@ 1500-2500rpm", "12.7@ 2,~
## $ seats               <dbl> 5, 5, 5, 5, 5, 5, 5, 4, 5, 5, 5, 5, 5, 5, 5, 7, 5,~
## $ sold                <chr> "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", ~
```

Missing values

There are no missing values in the dataset:

```
cbind(
  lapply(
    lapply(data, is.na)
    , sum)
)
```

```
##                   [,1]
## Sales_ID          0
## name              0
## year              0
## selling_price     0
## km_driven         0
## Region            0
## State or Province 0
## City              0
## fuel              0
## seller_type       0
## transmission      0
## owner             0
## mileage           0
## engine            0
## max_power         0
## torque            0
## seats             0
## sold              0
```

Engine torque

One of the variables contains details regarding the engine torque. Whilst the engine size and power may influence the decision making process of a potential buyer, it is unlikely that an average car buyer may consider the engine torque or base their purchase decision on it. Consequently, this variable won't be considered as a potential predictor for our model and this data is removed from the dataset to reduce the size of the dataset:

```
data_2 <- select(data, -torque)
```

Converting categorical data to factor

Variables containing categorical data (sales ID, brand name, region, state or province, city, fuel type, seller type, transmission type, owner type and number of seats) are converted to factor type to aid analysis and model training later:

```
data_3 <- mutate(data_2,
                     Sales_ID = factor(Sales_ID),
                     name = factor(name),
                     Region = factor(Region),
                     `State or Province` = factor(`State or Province`),
                     City = factor(City),
                     fuel = factor(fuel),
                     seller_type = factor(seller_type),
                     transmission = factor(transmission),
                     owner = factor(owner),
                     seats = factor(seats))
```

Categorical data is qualitative data that can be classified into groups or categories and can only take on certain values. Examples include gender, hair color, education level, etc.

Non-categorical data is numerical quantitative data that can be measured on a continuous scale and can take on a range of values. Examples include age, height, annual sales, etc. Not all numerical data is non-categorical. For instance, in this dataset, although the number of seats are measured numerically, it is categorical as it can only take on certain values (a car cannot have 2.4 seats for example).

Renaming variables

Some variables (sales ID, brand name, region, state or province and city) are renamed:

```
data_4 <- rename(data_3,
                      id = Sales_ID,
                      brand = name,
                      region = Region,
                      state_province = `State or Province`,
                      city = City)
```

Converting sold to 0, 1

The sold status of a car is recorded as “Y” if sold and “N” if unsold. This is replaced with 1 and 0 respectively to aid the model training process later.

```
data_5 <- data_4

data_5$sold <- ifelse(data_4$sold == "Y", 1, 0) %>% factor()
```

1.2 Summary statistics and visualization

The statistical data of the variables being considered as potential predictors for training the models are summarized and visualized in sections 1.2.1 and 1.2.2, and key observations are noted in section 1.2.3.

1.2.1 Summary statistics

The dataset contains 7906 records and 17 variables (1 record ID, 1 outcome and 15 potential predictors):

```
dim(data_5)
```

```
## [1] 7906   17
```

Summary statistics of the dataset’s variables:

```
summary(data_5)[,-1]
```

```
##       brand           year      selling_price        km_driven
##  Maruti  :2367   Min.   :1994   Min.   :   29999   Min.   :      1
##  Hyundai :1360   1st Qu.:2012   1st Qu.:  270000   1st Qu.:  35000
##  Mahindra: 758   Median :2015   Median :  450000   Median :  60000
##  Tata    : 719   Mean   :2014   Mean   :  649814   Mean   :  69189
##  Honda   : 466   3rd Qu.:2017   3rd Qu.:  690000   3rd Qu.:  95425
##  Toyota  : 452   Max.   :2020   Max.   :10000000   Max.   :2360457
##  (Other) :1784
##      region        state_province            city          fuel
##  Central:2376   California: 901   New York City: 196   CNG   :  52
##  East   :1939   Texas     : 499   Los Angeles  : 191   Diesel:4299
##  South  :1615   New York  : 491   Seattle      :  92   LPG   :  35
##  West   :1976   Illinois  : 489   Chicago      :  88   Petrol:3520
##                 Florida   : 425   Boston       :  75
##                 Ohio      : 321   Washington   :  68
##                 (Other)   :4780   (Other)      :7196
##            seller_type      transmission                owner
##  Dealer          :1107   Automatic:1041   First_Owner       :5215
##  Individual      :6563   Manual   :6865   Fourth_Above_Owner: 160
##  Trustmark_Dealer: 236                    Second_Owner      :2016
##                                           Test_Drive_Car    :   5
##                                           Third_Owner       : 510
##
##
##     mileage          engine       max_power          seats      sold
##  Min.   : 0.00   Min.   : 624   Min.   : 32.80   5      :6254   0:5906
##  1st Qu.:16.78   1st Qu.:1197   1st Qu.: 68.05   7      :1120   1:2000
##  Median :19.30   Median :1248   Median : 82.00   8      : 235
##  Mean   :19.42   Mean   :1459   Mean   : 91.59   4      : 133
##  3rd Qu.:22.32   3rd Qu.:1582   3rd Qu.:102.00   9      :  80
##  Max.   :42.00   Max.   :3604   Max.   :400.00   6      :  62
##                                                  (Other):  22
```

Number of brands:

```
n_distinct(data_5$brand)
```

```
## [1] 31
```

Number of states or provinces:

```
n_distinct(data_5$state_province)
```

```
## [1] 49
```

Number of cities:

```
n_distinct(data_5$city)
```

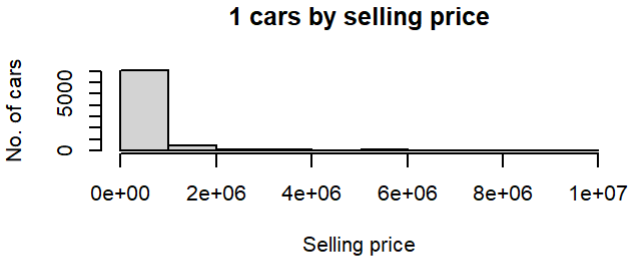
```
## [1] 1310
```

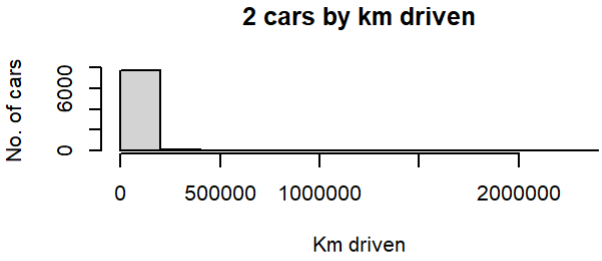
1.2.2 Visualizing the data

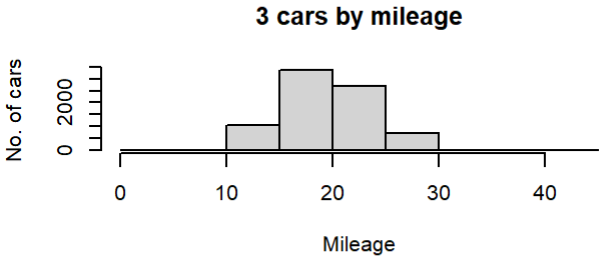
1.2.2.1 Non-categorical data with histograms

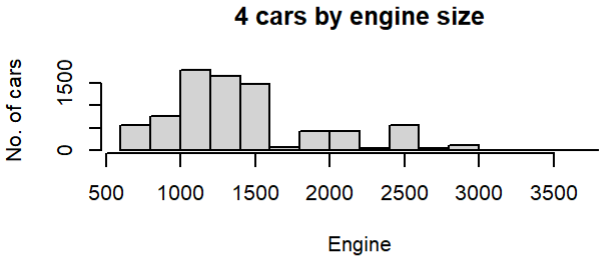
Selling price, distance driven, mileage, engine size and max engine power:

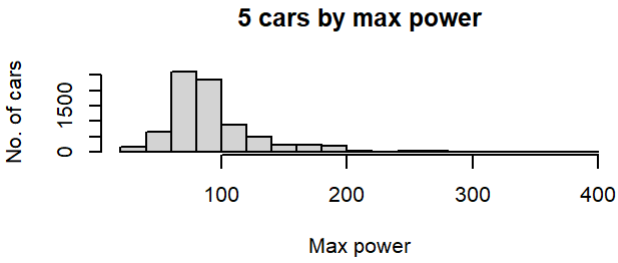
```
par(mfrow=c(3,2))

hist(data_5$selling_price, main = "1 cars by selling price", xlab = "Selling price", ylab =
"No. of cars")
hist(data_5$km_driven, main = "2 cars by km driven", xlab = "Km driven", ylab = "No. of cars"
)
hist(data_5$mileage, main = "3 cars by mileage", xlab = "Mileage", ylab = "No. of cars")
hist(data_5$engine, main = "4 cars by engine size", xlab = "Engine", ylab = "No. of cars")
hist(data_5$max_power, main = "5 cars by max power", xlab = "Max power", ylab = "No. of cars"
)
```











Skewness of engine size and max engine power:

```
skewness(data_5$engine)
```

```
## [1] 1.134928
```

```
skewness((data_5$max_power))
```

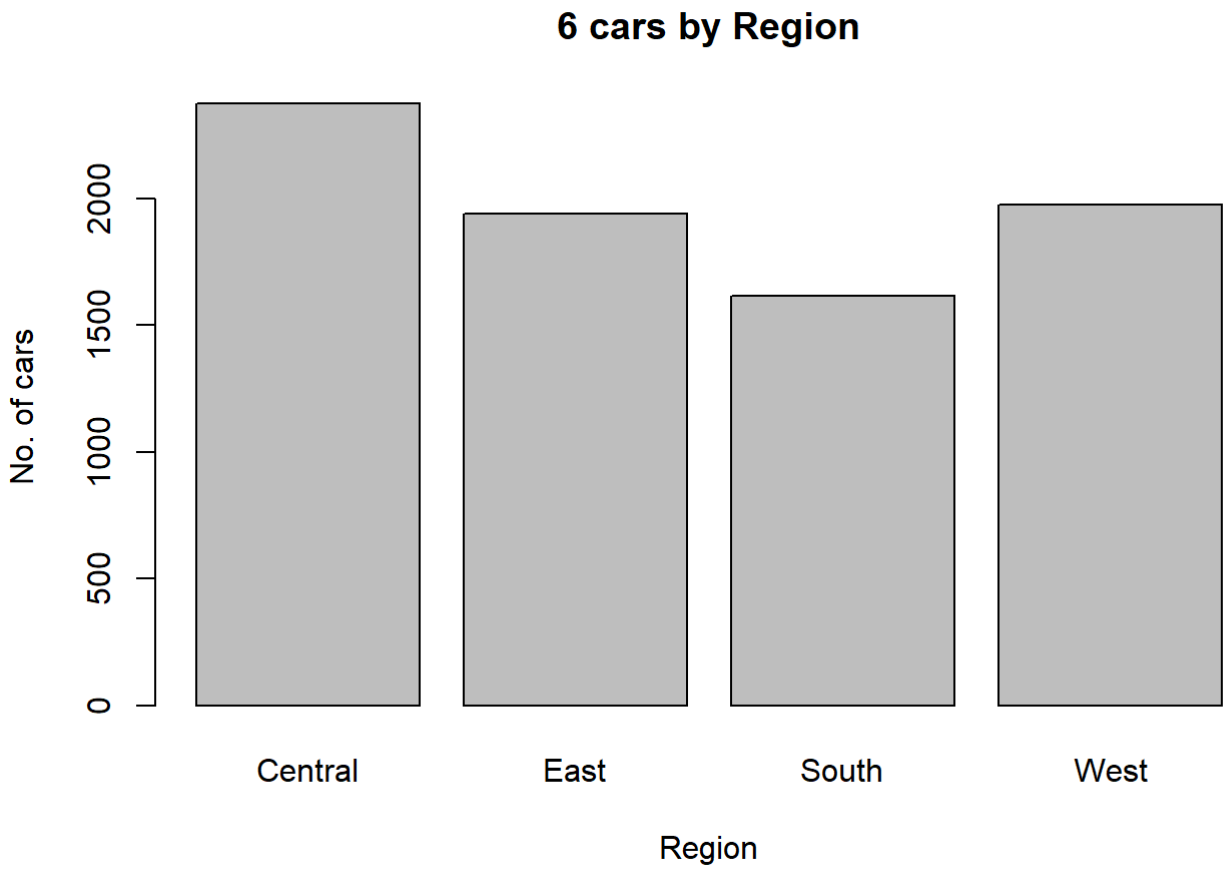
```
## [1] 1.639051
```

Skewness measures the asymmetry of a distribution where the skewness can be positive (the tail is on the right), negative (the tail is on the left) or zero (symmetrical). In a normal distribution (i.e. symmetrical) the mean and the median are equal.

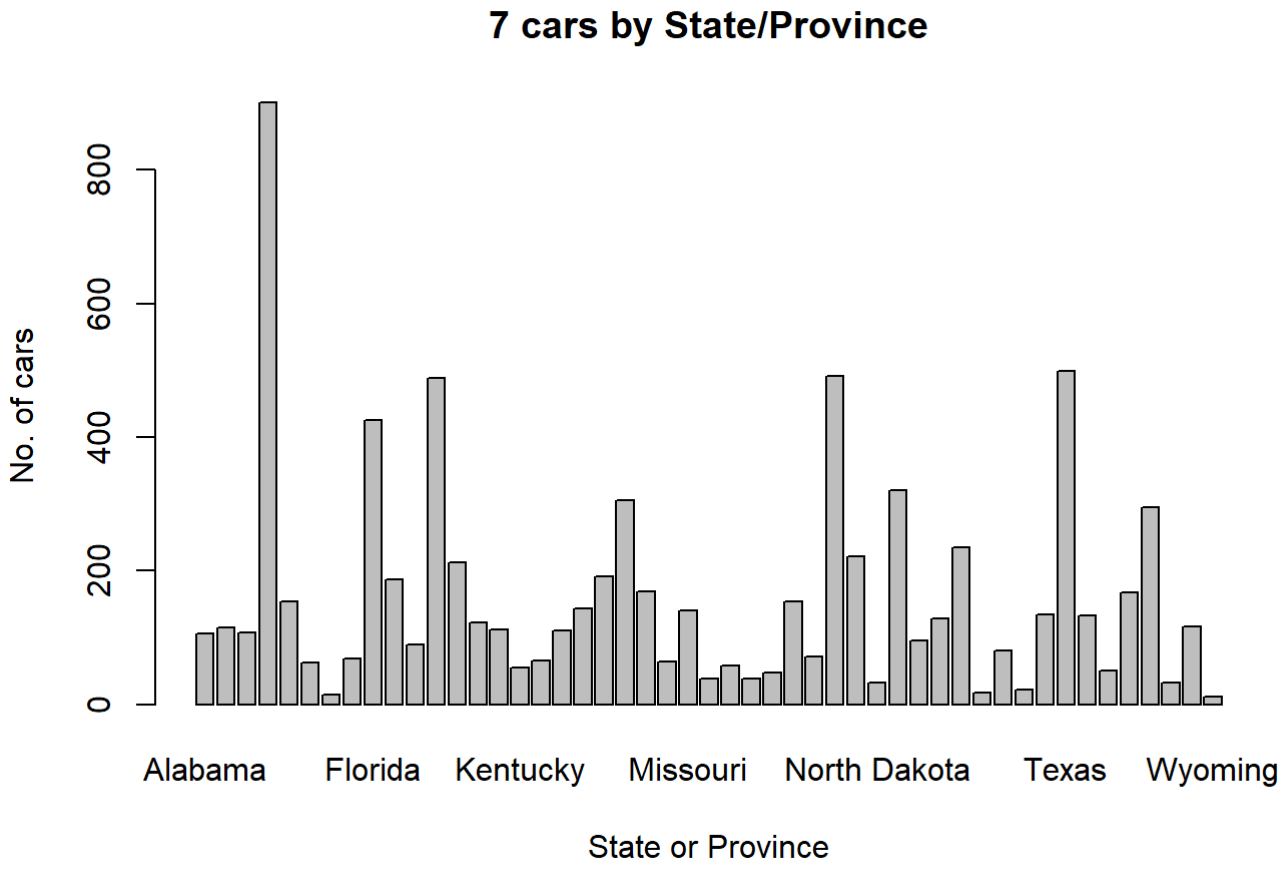
1.2.2.2 Categorical data with bar plots

Region, state or province, city and brand:

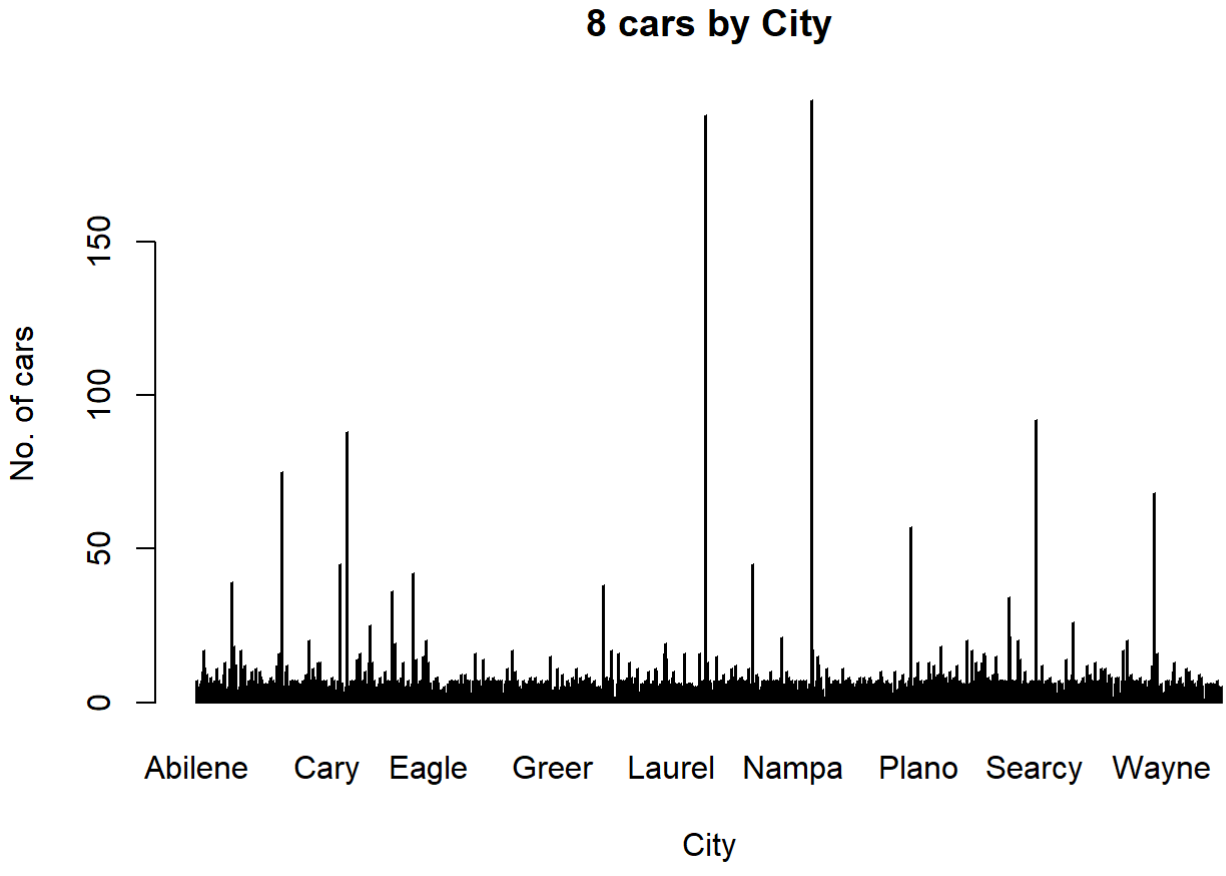
```
par(mfrow=c(1,1))

plot(data_5$region, main = "6 cars by Region", xlab = "Region", ylab = "No. of cars")
```



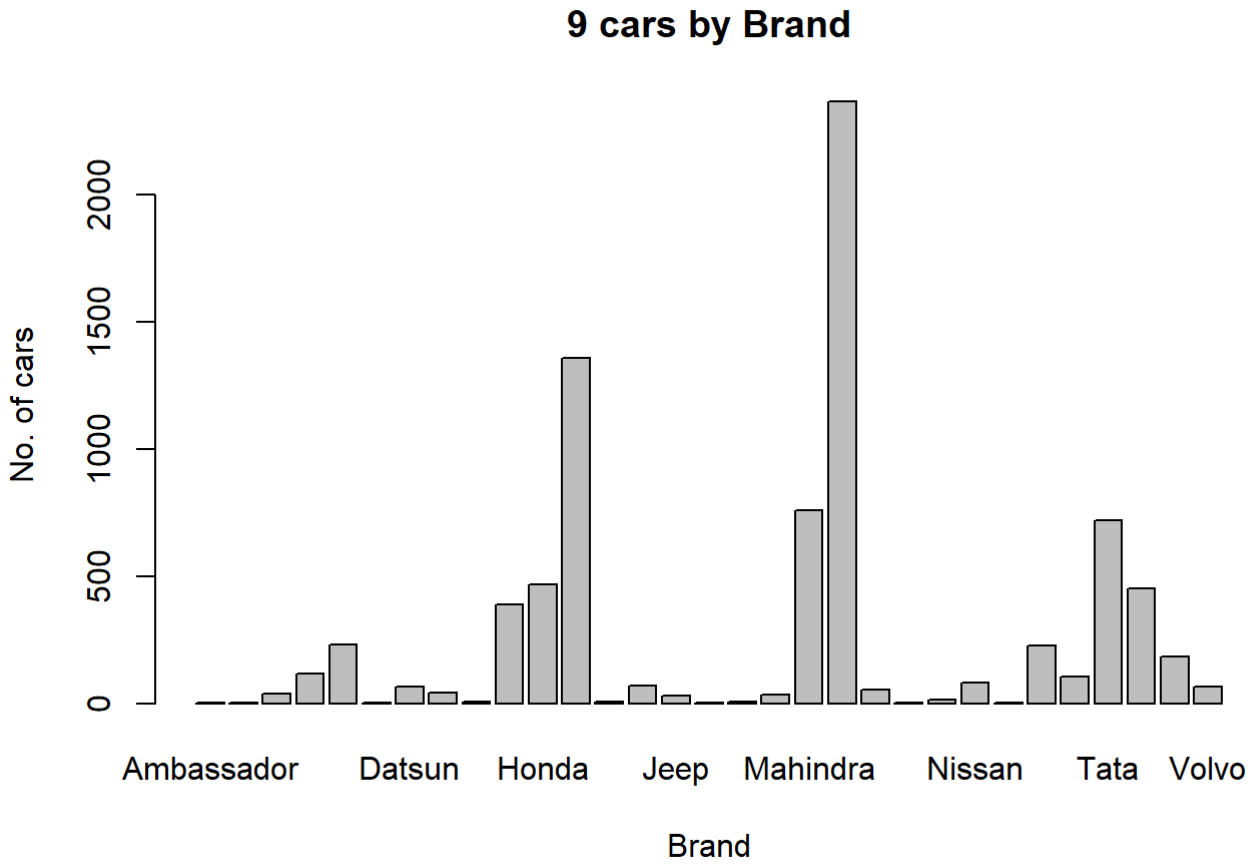

```
plot(data_5$state_province, main = "7 cars by State/Province", xlab = "State or Province", yl
ab = "No. of cars")
```




```
plot(data_5$city, main = "8 cars by City", xlab = "City", ylab = "No. of cars")
```

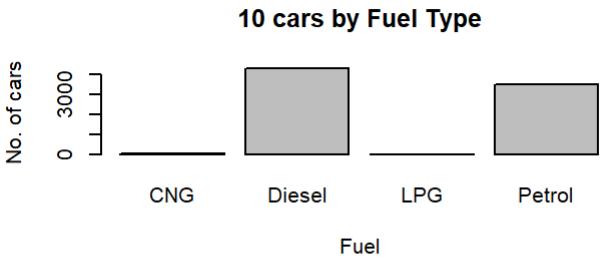


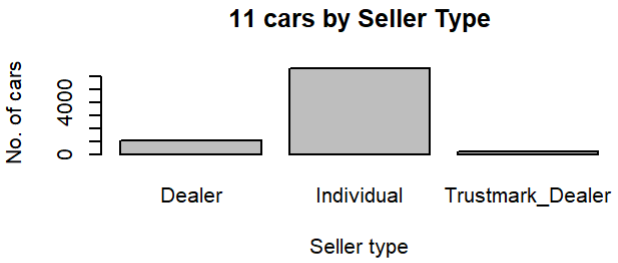
```
plot(data_5$brand, main = "9 cars by Brand", xlab = "Brand", ylab = "No. of cars")
```

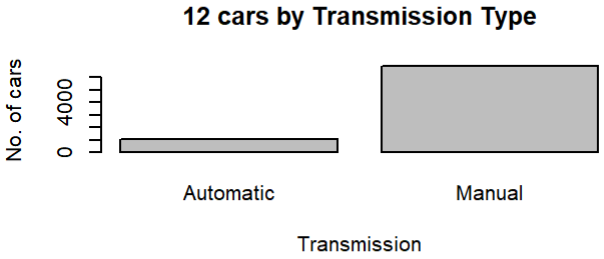


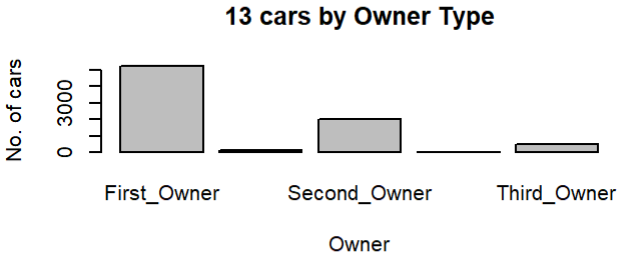
Fuel type, seller type, transmission type, owner type, number of seats and year:

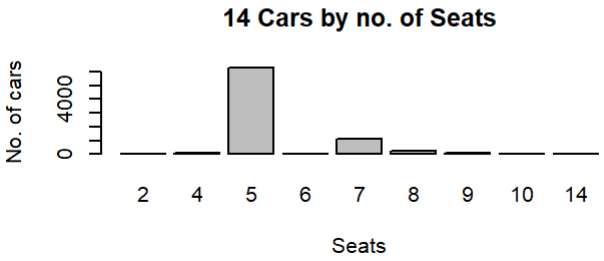
```
par(mfrow=c(3,2))

plot(data_5$fuel, main = "10 cars by Fuel Type", xlab = "Fuel", ylab = "No. of cars")
plot(data_5$seller_type, main = "11 cars by Seller Type", xlab = "Seller type", ylab = "No. o
f cars")
plot(data_5$transmission, main = "12 cars by Transmission Type", xlab = "Transmission" , ylab
= "No. of cars")
plot(data_5$owner, main = "13 cars by Owner Type", xlab = "Owner" , ylab = "No. of cars")
plot(data_5$seats, main = "14 Cars by no. of Seats", xlab = "Seats" , ylab = "No. of cars")
hist(data_5$year,main = "15 Cars by Year", xlab = "Year" , ylab = "No. of cars")
```

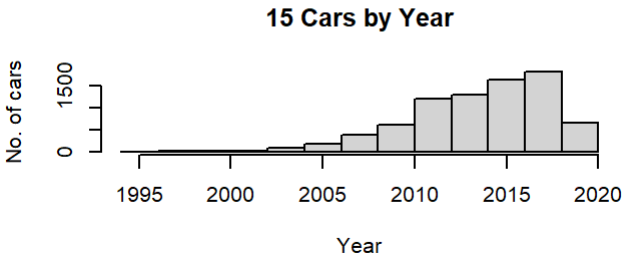












1.2.3 Observations (from 1.2.1 and 1.2.2):

Brand: The cars in the dataset are not equally distributed among the 31 brands represented in the dataset. Almost half of the cars (47%) are either Maruti or Hyundai. 77% of the cars in the dataset are represented by just 6 brands.

Year: The oldest car in the dataset is from 1994 and the majority are from 2012 or later.

Selling price: The cheapest car is just under 30,000 dollars and the most expensive car is 10,000,000 dollars. The median average price is 450,000 dollars. Based on the positive skew of the histogram, the majority of the cars seem to be concentrated at the lower end of the pricing range.

Distance driven: The shortest distance driven is just 1 km and the longest distance driven is over 2.3 million km. The median average distance driven is 60,000 km. Based on the positive skew of the histogram, the majority of the cars seem to be concentrated at the lower end of the range of distance driven.

Region: 30% of the cars are in the central region - the region with the highest proportion of cars in the dataset. The other three regions have a fairly similar number of cars in the dataset.

State/Province: The cars in the dataset are not equally distributed across all the states. Just over 11% of cars are in California - the state with the highest proportion of cars in the dataset. Other states with a large proportion of cars include Texas, New York, Illinois and Florida.

City: New York City and Los Angeles combined account for nearly 5% of all the cars in the dataset. The cities that are larger and more populous seem to account for more cars in the dataset than smaller less populous cities which may not be very surprising.

Fuel type: Nearly 99% of the cars run on either diesel or petrol.

Seller type: Majority of the cars (83%) are being sold by individual sellers.

Transmission: Majority of the cars (87%) have a manual transmission.

Owner: Majority of the cars (91%) are being sold by their first or second owners.

Mileage: The lowest mileage seems to be 0. This seems highly unlikely as even the most fuel inefficient car would have a mileage figure of over 0 so this will be investigated further later as it is likely that 0 represents a missing value rather than the reported mileage. The median average mileage is 19.3. The shape of the histogram seems to suggest that the mileage of the cars in the dataset is normally distributed.

Engine size: The smallest engine size is 624cc while the largest is 3604cc. The median average size is 1248cc. Based on the positive skew of the histogram (skewness = 1.13), the majority of the cars in the dataset seem to have relatively small sized engines.

Engine power: The lowest power output is 32.8hp while the highest output is 400hp. The median average output is 82hp. Based on the positive skew of the histogram (skewness = 1.63), the majority of the cars in the dataset seem to have engines that produce less than 100hp.

Seats: Majority of the cars (79%) have five seats.

Sold: Majority of the cars (75%) are unsold.

1.3 Further exploration and manipulation

As already observed in 1.2.3, some cars in the dataset report to have a fuel efficiency of 0. Further exploration reveals that there are 17 cars in the dataset with 0 fuel efficiency:

```
data_5 %>%
  filter(mileage == 0)
```

```
## # A tibble: 17 x 17
##    id    brand   year selling_price km_driven region state_province city   fuel
##    <fct> <fct>  <dbl>         <dbl>     <dbl> <fct>  <fct>          <fct>  <fct>
##  1 645   Tata    2009        135000     28900 East   New York       New Y~ Petr~
##  2 786   Hyund~  2009        120000     90000 East   New York       New Y~ Petr~
##  3 1650  Hyund~  2008        105000    128000 East   New York       North~ Petr~
##  4 1677  Merce~  2011       1700000    110000 East   Pennsylvania   Altoo~ Dies~
##  5 2138  Land    2013       1650000     64788 Centr~ Texas          Victo~ Dies~
##  6 2367  Hyund~  2010        110000     80000 West   Oregon         Pendl~ Petr~
##  7 2726  Hyund~  2013        184000     15000 Centr~ Illinois       Elk G~ Petr~
##  8 4528  Merce~  2011       1700000    110000 West   California     Stock~ Dies~
##  9 5277  Hyund~  2008        175000     40000 West   California     Temec~ Petr~
## 10 5844  Volks~  2014        574000     28080 Centr~ North Dakota   Mandan Petr~
## 11 5847  Volks~  2014        575000     28100 Centr~ North Dakota   Mandan Petr~
## 12 5901  Mahin~  2020        679000      5000 Centr~ Illinois       Fores~ Dies~
## 13 6535  Hyund~  2010        150000    110000 South  North Carolina Conco~ Petr~
## 14 6630  Mahin~  2019        722000     80000 South  Florida        Brade~ Dies~
## 15 6825  Hyund~  2011        150000     40000 Centr~ Missouri       Unive~ Petr~
## 16 7003  Hyund~  2010        110000     80000 Centr~ Indiana        Valpa~ Petr~
## 17 7338  Merce~  2017       3300000     60000 West   Idaho          Eagle  Dies~
## # ... with 8 more variables: seller_type <fct>, transmission <fct>,
## #   owner <fct>, mileage <dbl>, engine <dbl>, max_power <dbl>, seats <fct>,
## #   sold <fct>
```

As noted previously, this suggests missing values or unavailable data. Before proceeding with any further exploration or model training, it is important to either remove entries of cars with 0 mileage or replace them with an appropriate substitute.

As the distribution of the mileage data is fairly normal and symmetrical (skewness = -0.14), we can be confident that the median mileage value will serve as a suitable approximation of the mileage of the cars with 0 mileage.

Consequently, cars with 0 mileage in the dataset are replaced with the median mileage of all the other cars:

```
skewness(data_5$mileage)
```

```
## [1] -0.1423365
```

```
mileage_median <- data_5 %>%
  filter(mileage != 0) %>%
  select(mileage) %>%
  as.matrix() %>%
  colMedians(.)

usedcardata <- data_5

usedcardata$mileage <- replace(data_5$mileage, data_5$mileage==0, mileage_median)

summary(usedcardata$mileage)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    9.00   16.78   19.33   19.46   22.32   42.00
```

1.3.1 Exploring sold:unsold ratio across all the variables

As observed in 1.2.3, overall 75% of the cars in the dataset are unsold. The purpose of this project is to train models that can predict if a given car is likely to be sold or unsold based on the variable/s used to train the models. But first, it is necessary to identify the variable/s best suited for training the models.

For the categorical variables, their suitability for training the models is determined by assessing the ratio of sold:unsold cars while controlling for the different categories. This will help us see if the proportion of sold to unsold cars in the categories within the variables is similar to the overall proportion or not which in turn will help identify which variables can help determine if a car is more likely to be sold than unsold. The overall proportion for the dataset is represented in the plots with a vertical dashed line to assist with visually comparing it against the proportions for the individual categories.

For the non-categorical variables, their suitability is assessed by analysing the distribution of their data and the summary statistics while controlling for their sold or unsold status. This is done by using a combination of violin and box plots. This will help us see if, for a given variable, the distribution of data and the summary statistics for cars that are sold and unsold are similar or not.

A violin plot shows the density distribution of data and is very useful for comparing the distribution of data categorized into different groups.

A box plot is used to display the distribution of a dataset using their quartiles and shows a five number summary that consists of the minimum/maximum values, the 1st and 3rd quartiles and the median. Box plots can be useful when comparing the data distribution between groups.

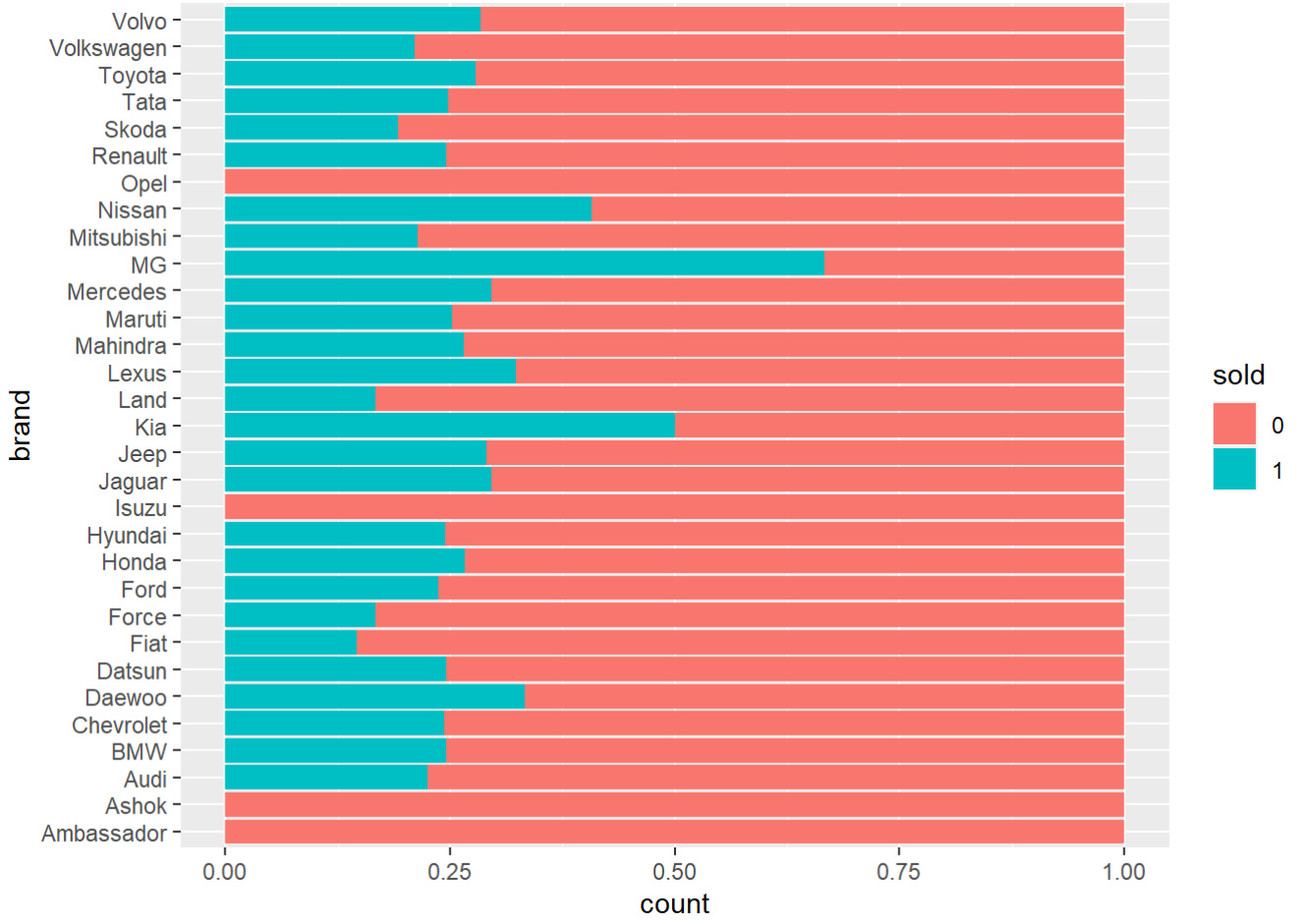
1.3.1.1 Brand

Overall there seems to be a lot of variability in the proportion of sold cars across the different brands.

However, for the top 6 brands that together represent 77% of the cars in the dataset, the proportion of sold cars is very similar to the overall 25% proportion for the dataset. It seems unlikely that we can use a car's brand to predict its sold status.

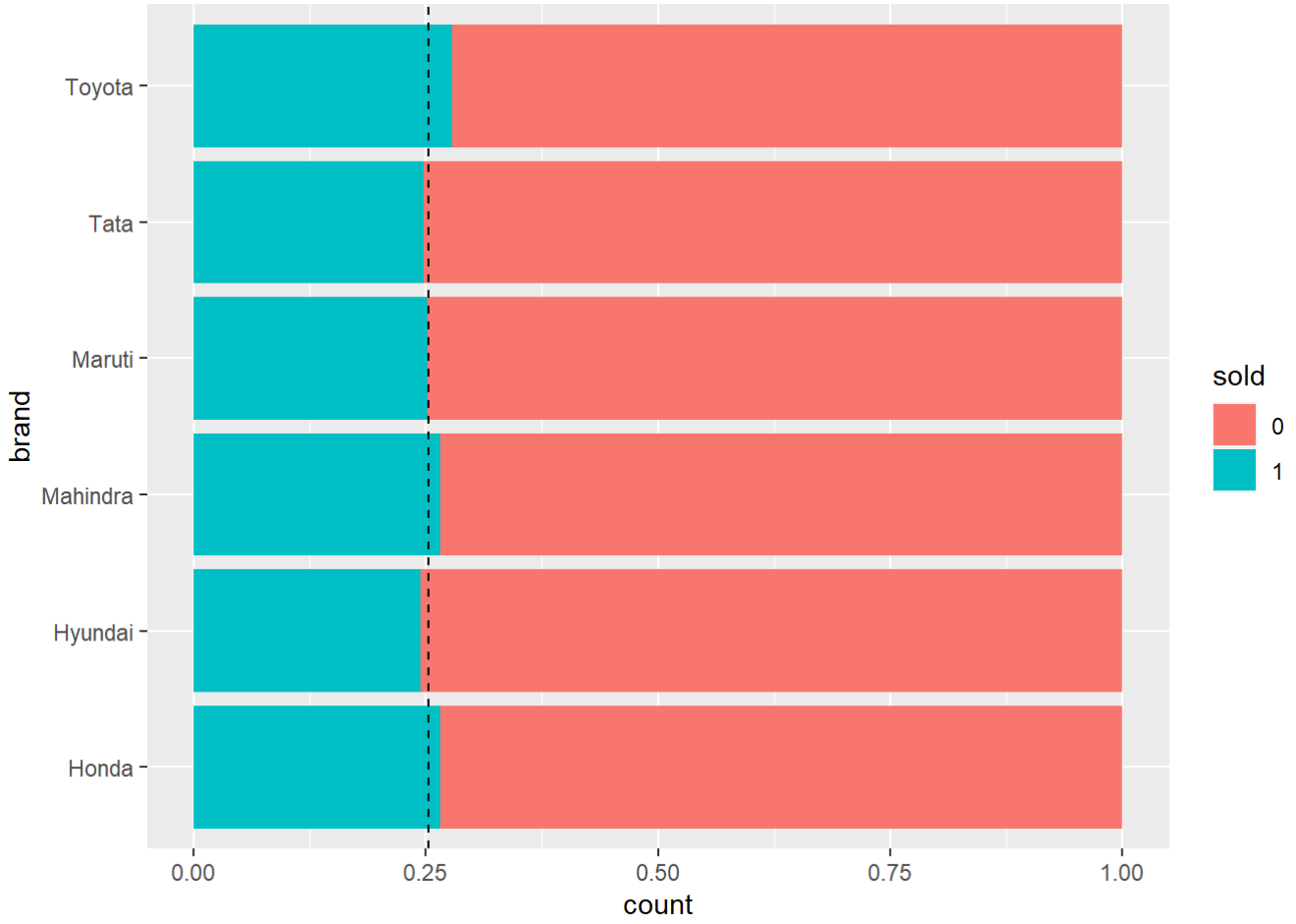
All brands:

```
ggplot(usedcardata, aes(y = brand, fill = sold)) + geom_bar(position = "fill")
```



Top 6 brands:

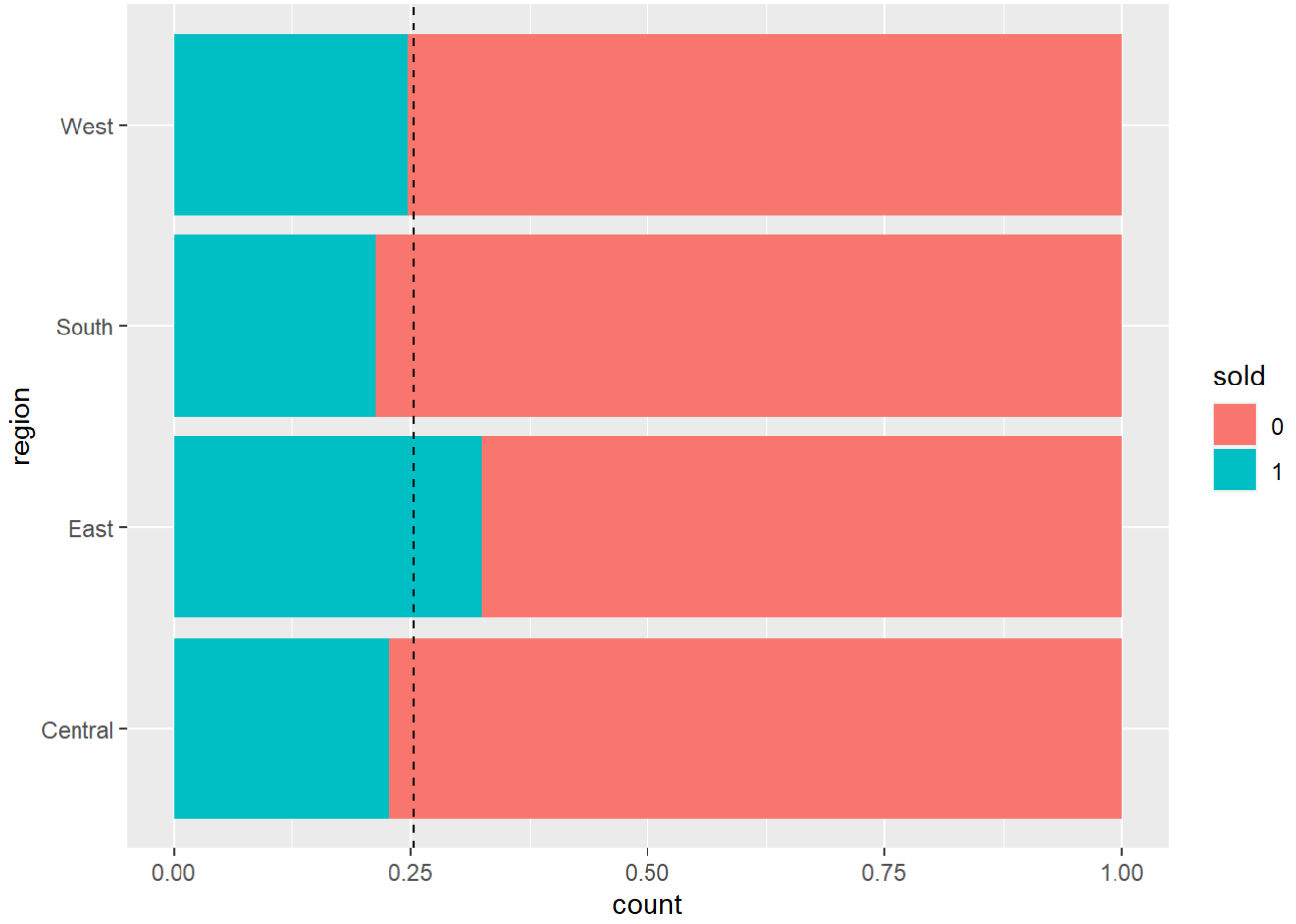
```
top_brand_names <- c("Maruti", "Hyundai", "Mahindra", "Tata", "Honda", "Toyota")

top_brands <- filter(usedcardata, brand %in% top_brand_names)

ggplot(top_brands, aes(y = brand, fill = sold)) +
  geom_bar(position = "fill") +
  geom_vline(xintercept = sold_prop, color="black", linetype="dashed")
```



1.3.1.2 Region

The eastern region seems to have a higher proportion of sold cars compared to the other three regions and compared to the overall 25% proportion for the dataset. The region where a car is being made available for sale could help us predict its sold status.

```
ggplot(usedcardata, aes(y = region, fill = sold)) +
  geom_bar(position = "fill") +
  geom_vline(xintercept = sold_prop, color="black", linetype="dashed")
```



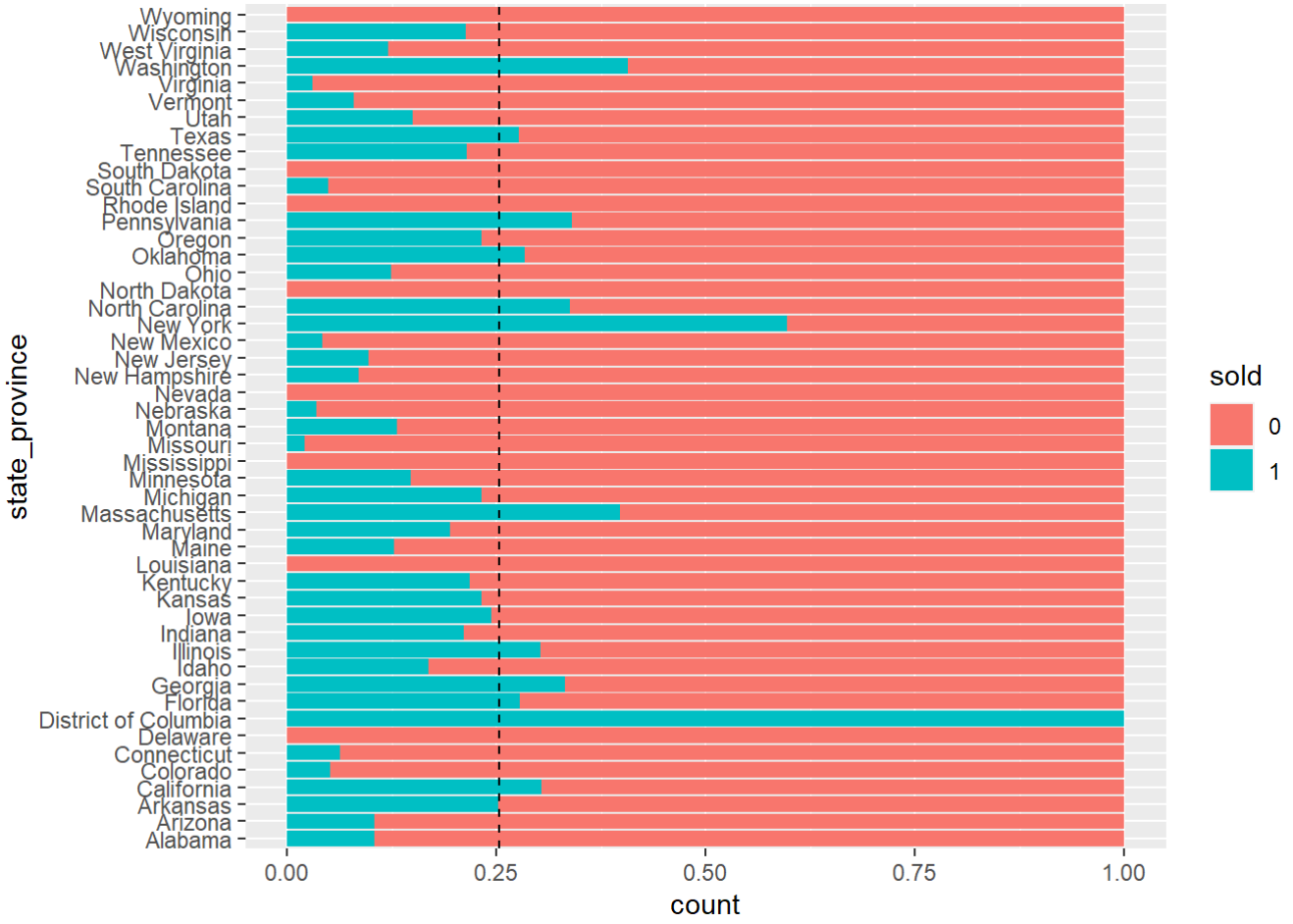
1.3.1.3 State/Province

Overall there seems to be a lot of variability in the proportion of sold cars across the states and provinces.

When looking at just the top 6 states that together represent just under 40% of cars in the dataset, there still seems to be variability in the proportion of sold cars, especially for New York. The state or province where a car is being made available for sale could help us predict its sold status.

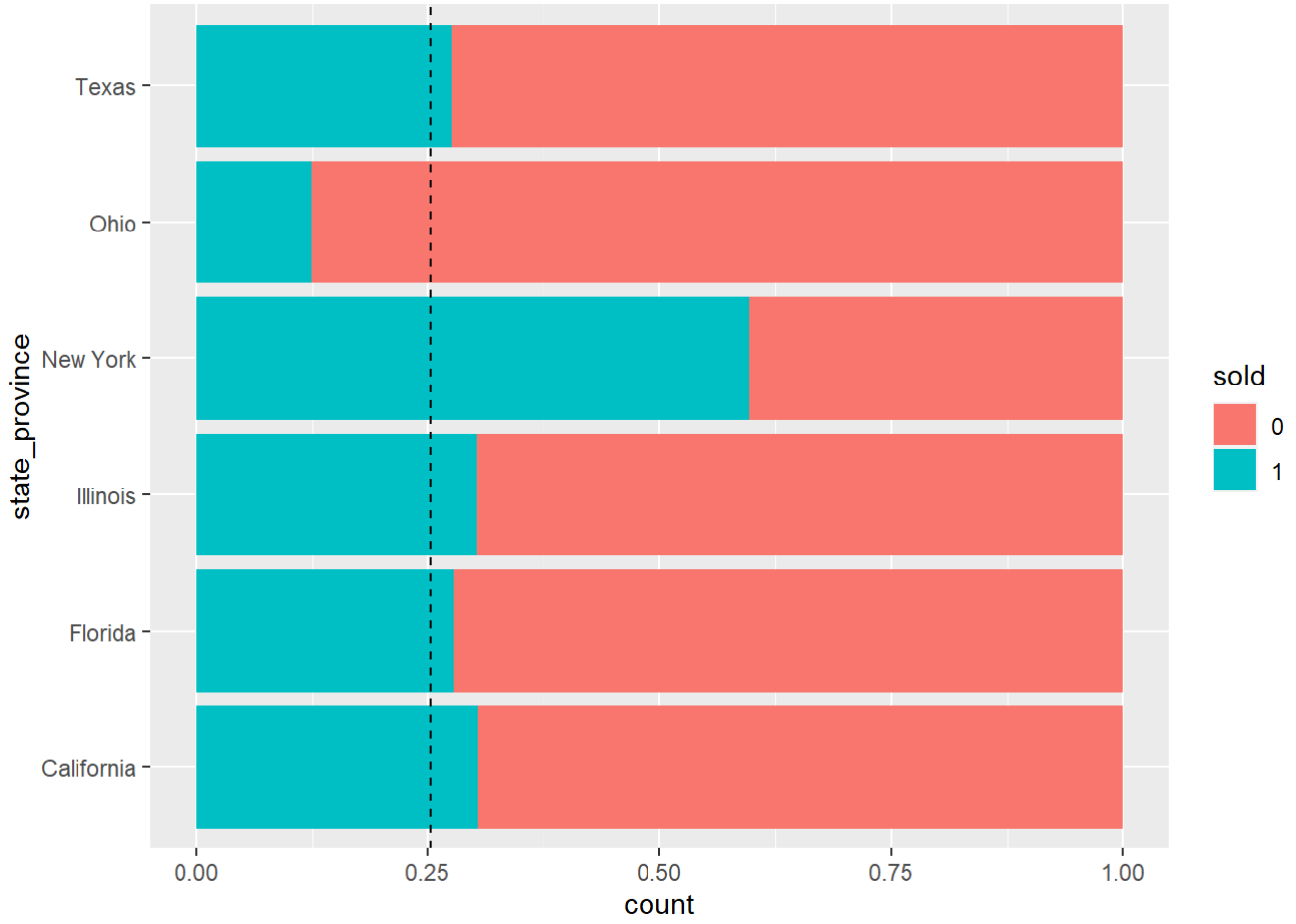
All states/provinces:

```
ggplot(usedcardata, aes(y = state_province, fill = sold)) +
  geom_bar(position = "fill") +
  geom_vline(xintercept = sold_prop, color="black", linetype="dashed")
```



Top 6 states/provinces:

```
top_sp_names <- c("California", "Texas", "New York", "Illinois", "Florida", "Ohio")

top_sp <- filter(usedcardata, state_province %in% top_sp_names)

ggplot(top_sp, aes(y = state_province, fill = sold)) +
  geom_bar(position = "fill") +
  geom_vline(xintercept = sold_prop, color="black", linetype="dashed")
```



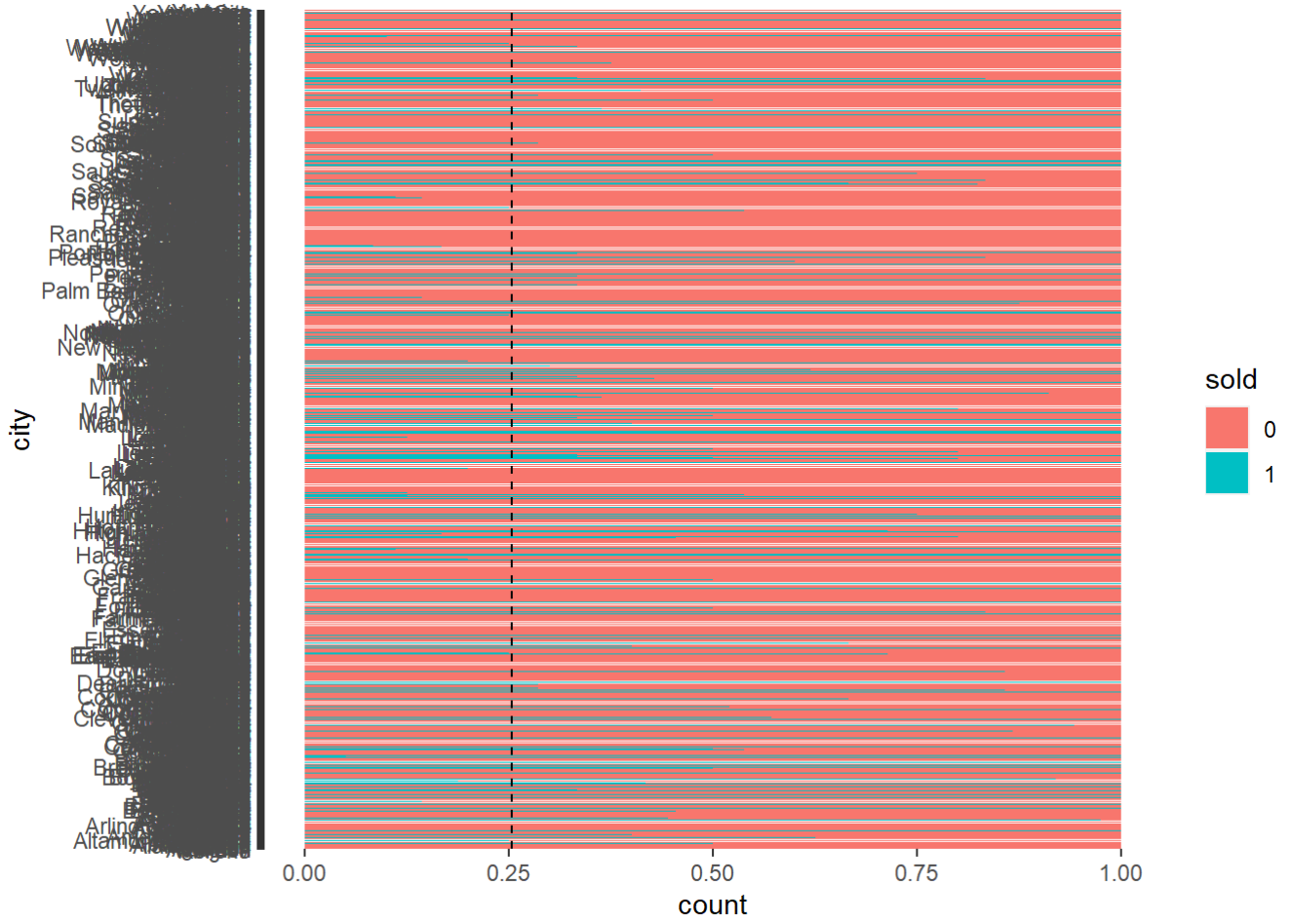
1.3.1.4 City

There are a lot of cities represented in the dataset so it's difficult to visually determine the proportion of sold cars for individual cities but overall there seems to be a lot of variability in the proportion of sold cars across the cities.

When looking at just the top 6 cities, it can be seen that most of the cars from these cities are sold which is in stark contrast to the 25% overall proportion for the dataset. The city where a car is being made available for sale could help us predict its sold status.

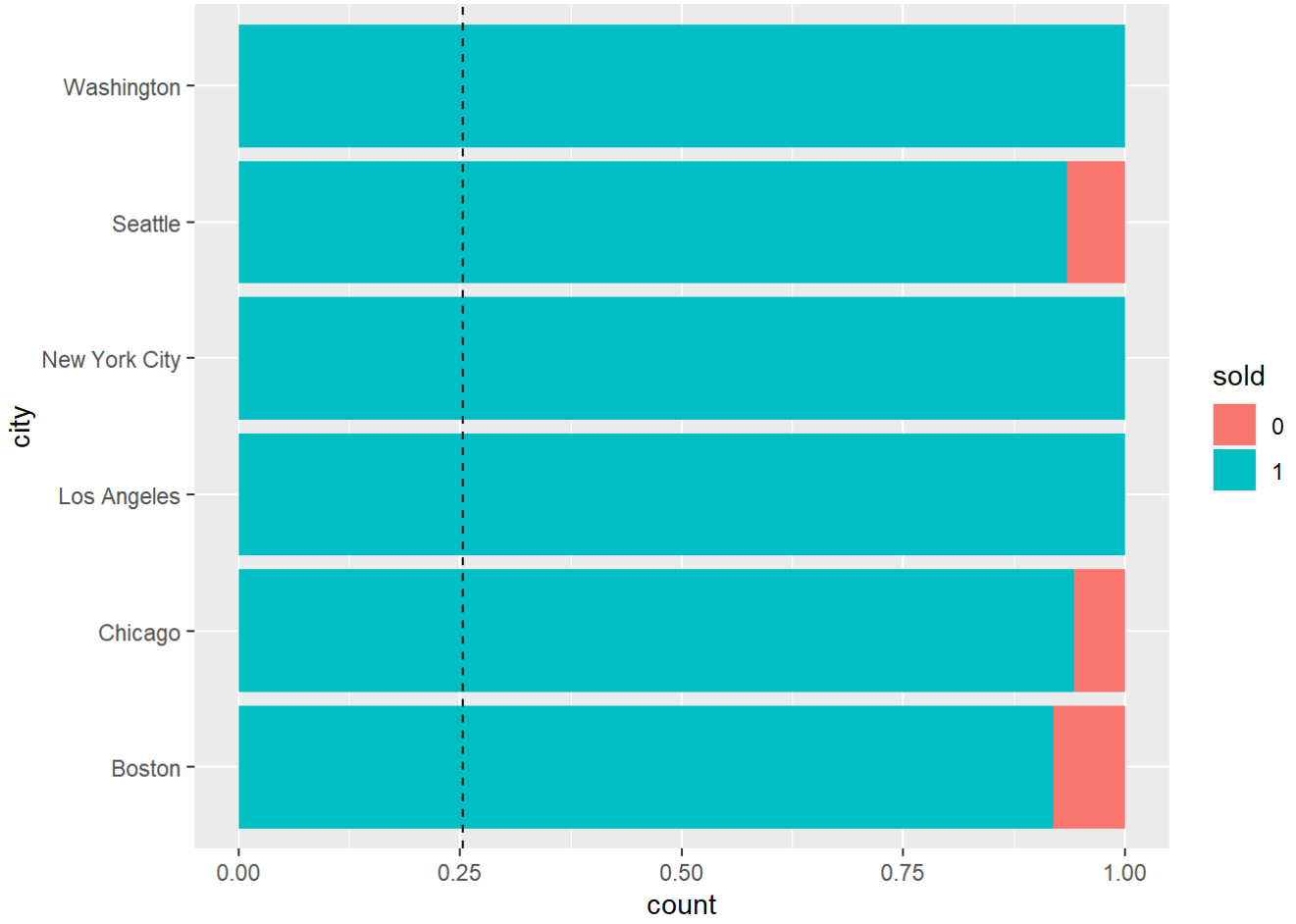
All cities:

```
ggplot(usedcardata, aes(y = city, fill = sold)) +
  geom_bar(position = "fill") +
  geom_vline(xintercept = sold_prop, color="black", linetype="dashed")
```



Top 6 cities:

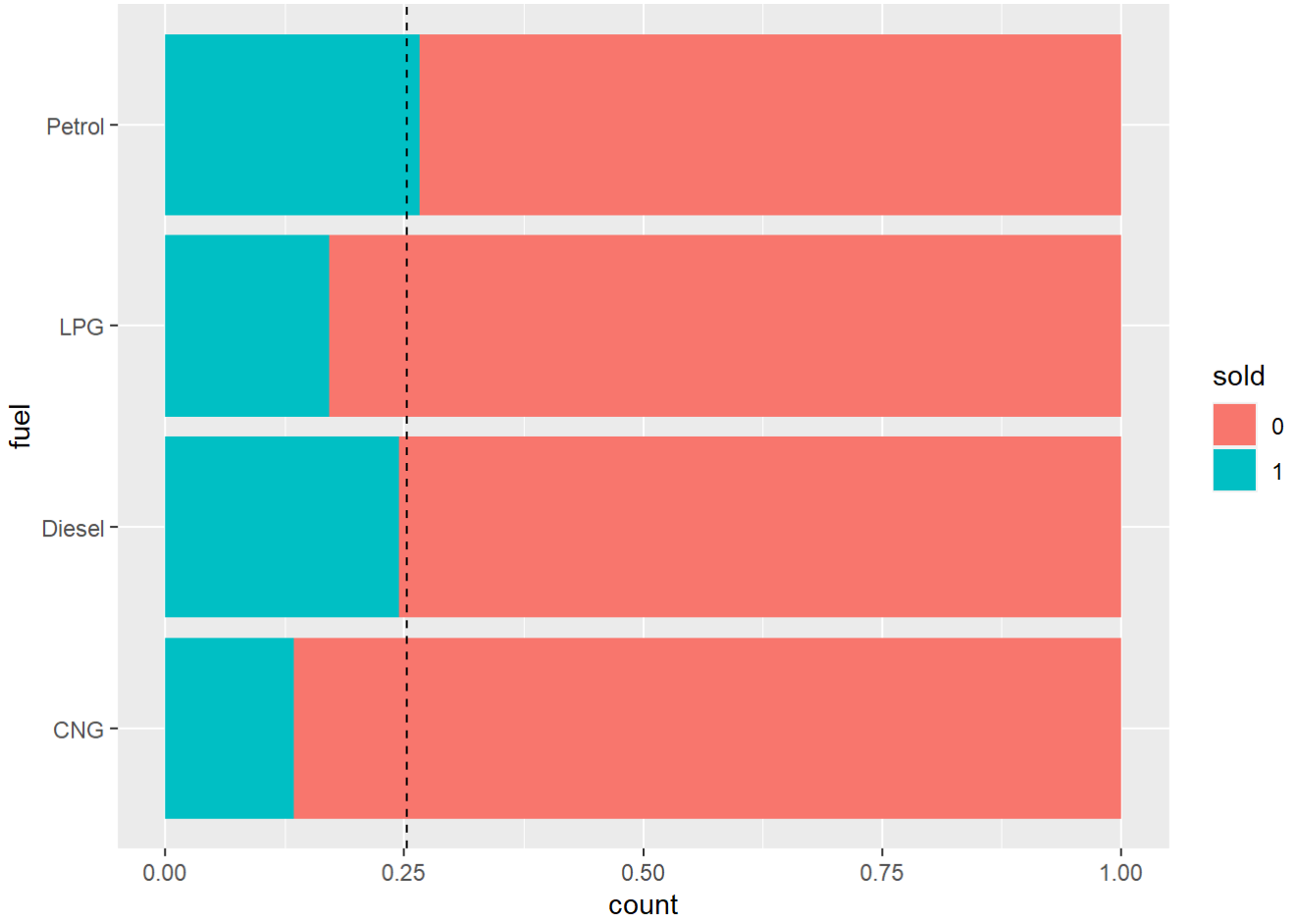
```
top_city_names <- c("New York City", "Los Angeles", "Seattle", "Chicago", "Boston", "Washingt
on")

top_city <- filter(usedcardata, city %in% top_city_names)

ggplot(top_city, aes(y = city, fill = sold)) +
  geom_bar(position = "fill") +
  geom_vline(xintercept = sold_prop, color="black", linetype="dashed")
```



1.3.1.5 Fuel type

For the two fuel types (petrol and diesel) most represented in the dataset, the proportion of sold cars seems to be similar to the 25% overall proportion for the dataset. It seems unlikely that we can use a car's fuel type to predict its sold status.

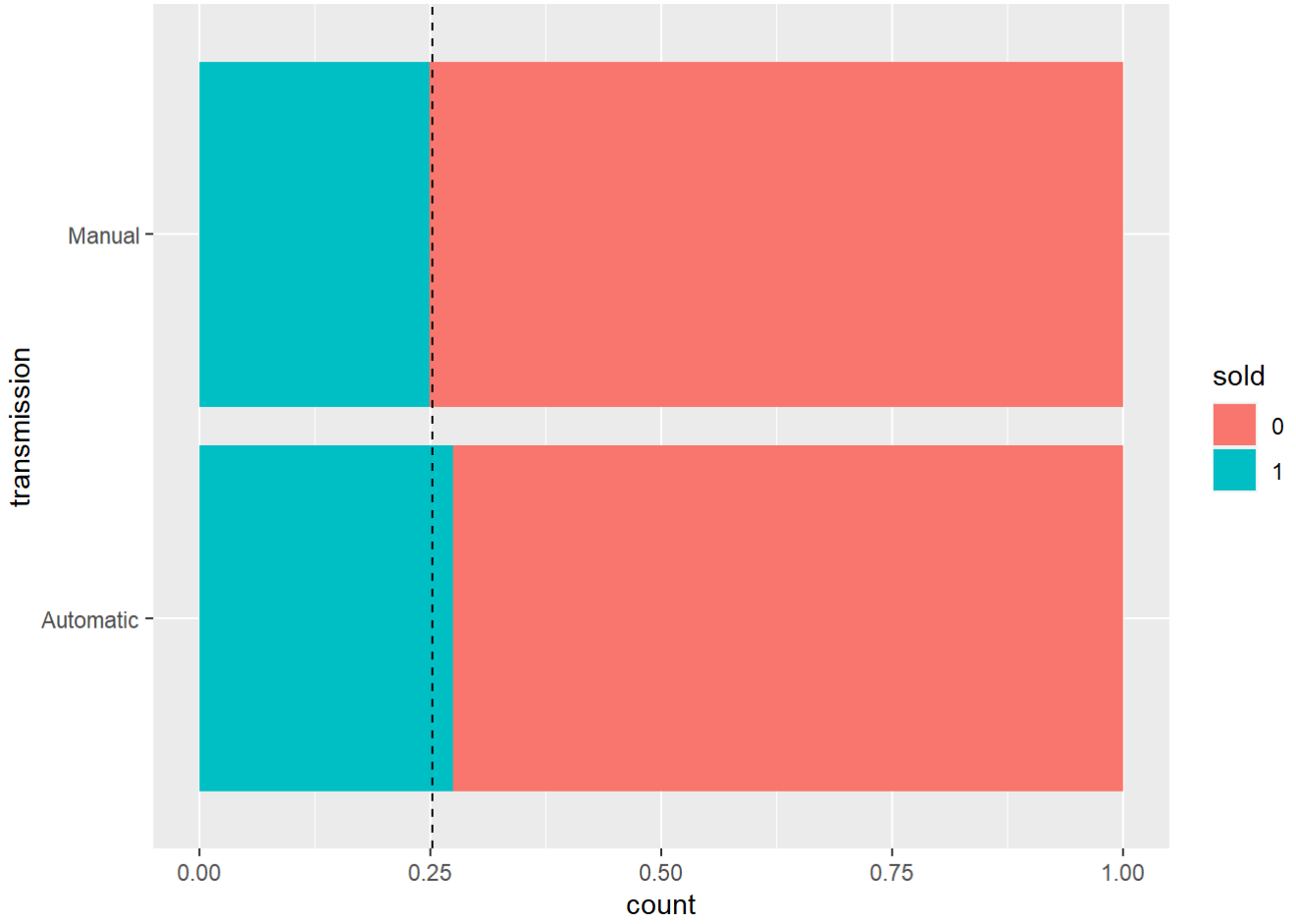
```
ggplot(usedcardata, aes(y = fuel, fill = sold)) +
  geom_bar(position = "fill") +
  geom_vline(xintercept = sold_prop, color="black", linetype="dashed")
```



1.3.1.6 Transmission type

The proportion of sold cars for the two transmission types seems to be similar to the 25% overall proportion for the dataset. It seems unlikely that we can use a car's transmission type to predict its sold status.

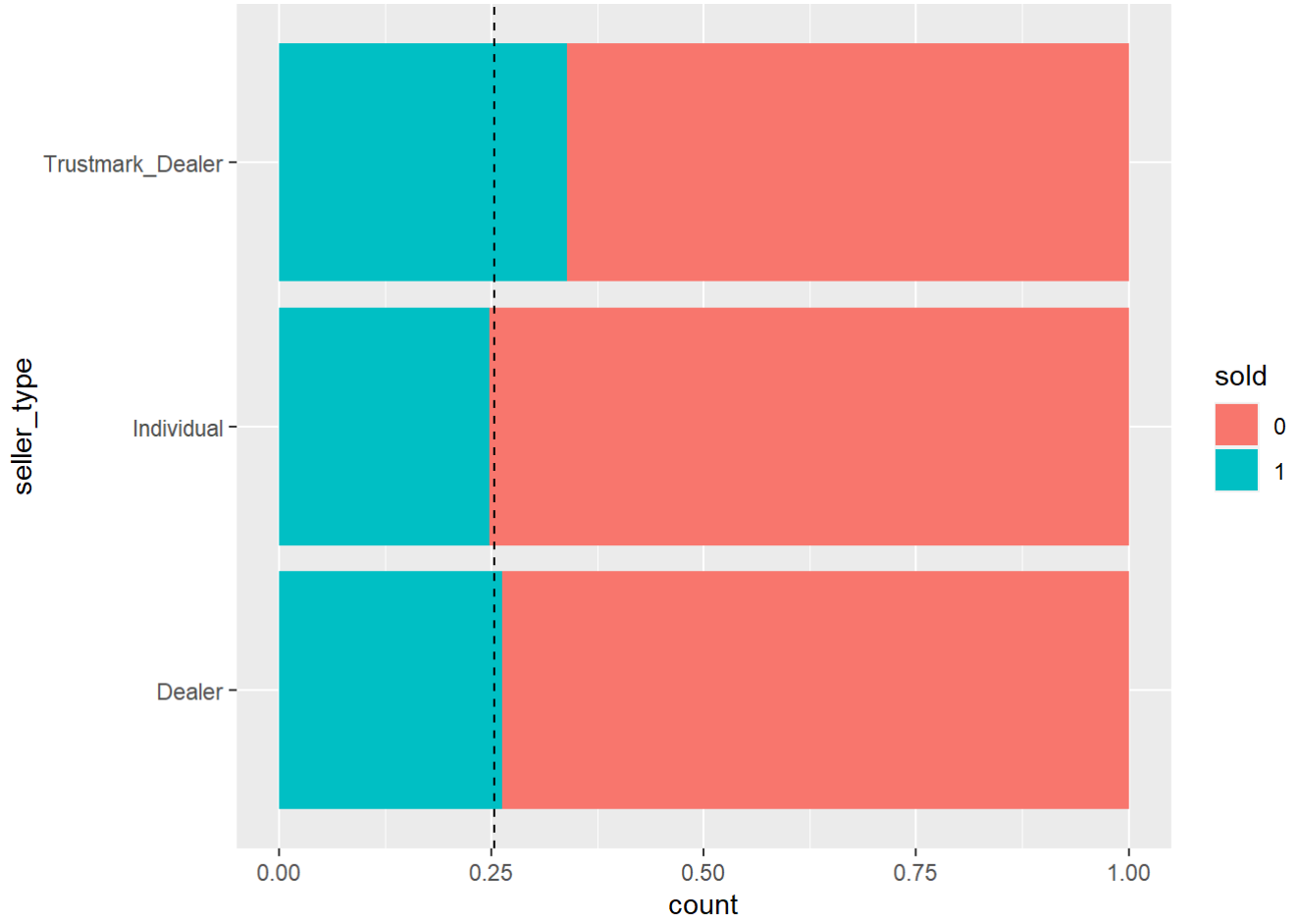
```
ggplot(usedcardata, aes(y = transmission, fill = sold)) +
  geom_bar(position = "fill") +
  geom_vline(xintercept = sold_prop, color="black", linetype="dashed")
```



1.3.1.7 Seller type

For the two most represented seller types (individual and dealer), the proportion of sold cars seems to be similar to the 25% overall proportion for the dataset. It seems unlikely that we can use a car's seller type to predict its sold status.

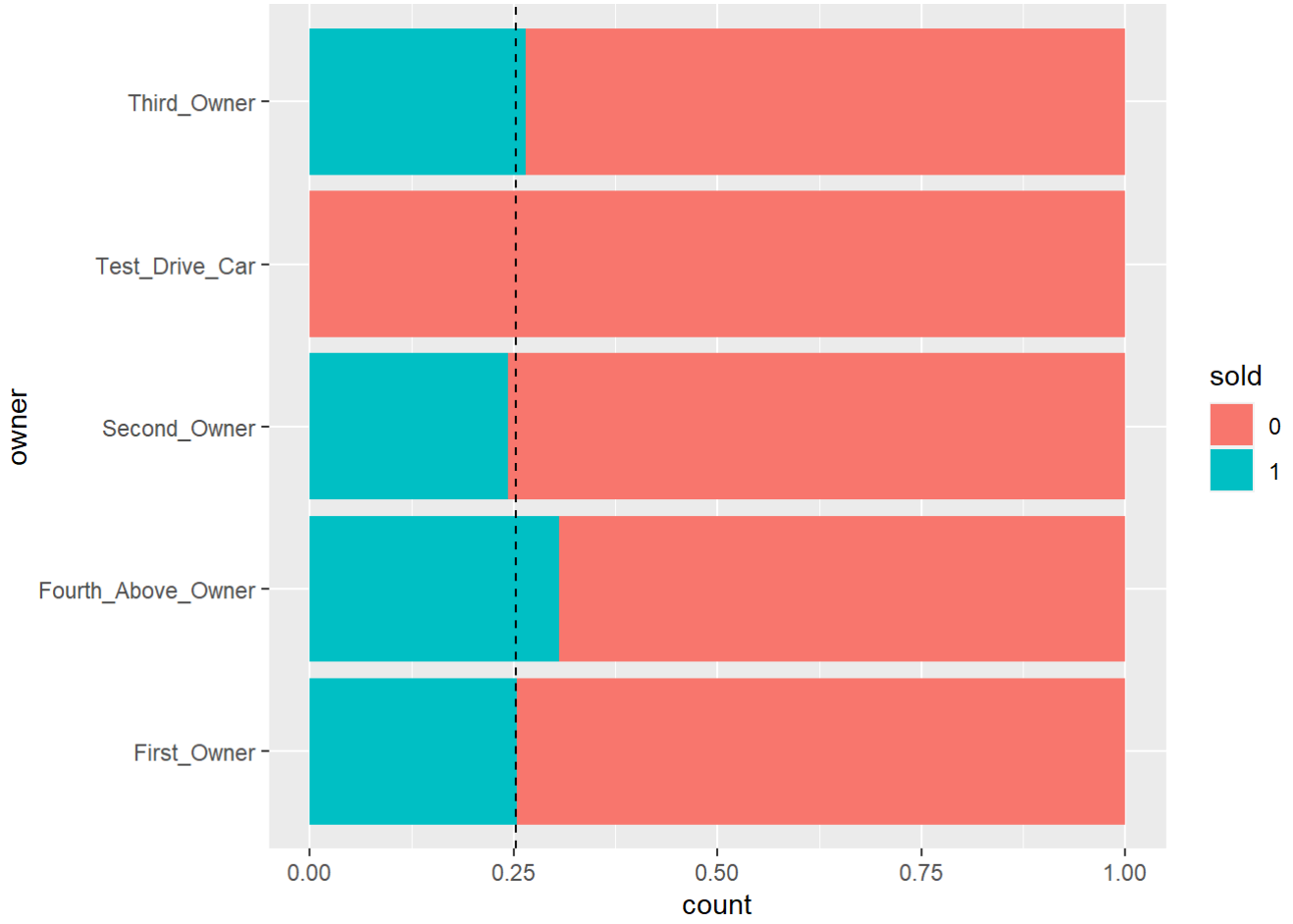
```
ggplot(usedcardata, aes(y = seller_type, fill = sold)) +
  geom_bar(position = "fill") +
  geom_vline(xintercept = sold_prop, color="black", linetype="dashed")
```



1.3.1.8 Owner type

For the two most represented owner types (first and second owners), the proportion of sold cars seems to be similar to the 25% overall proportion for the dataset. It seems unlikely that we can use a car's owner type to predict its sold status.

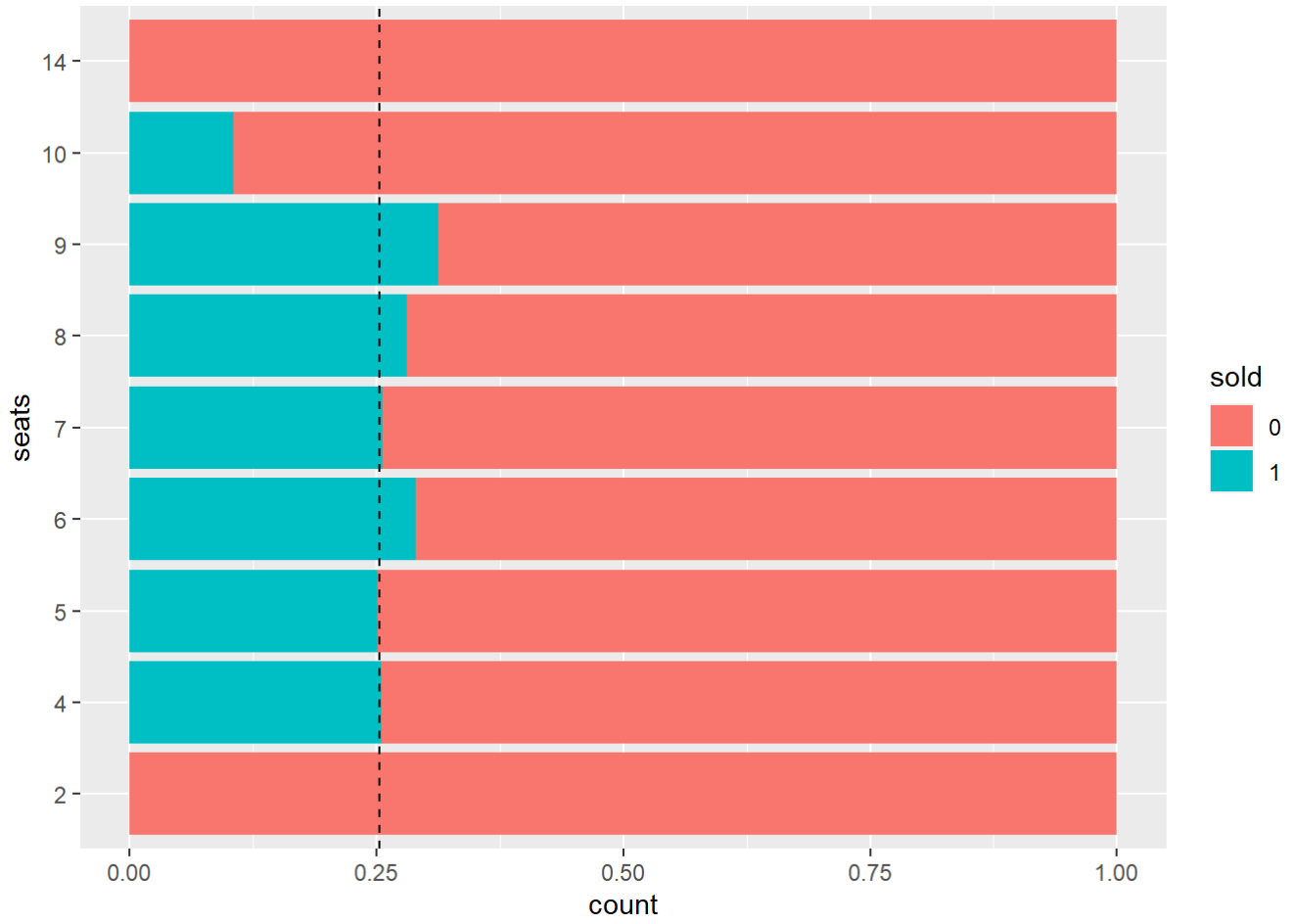
```
ggplot(usedcardata, aes(y = owner, fill = sold)) +
  geom_bar(position = "fill") +
  geom_vline(xintercept = sold_prop, color="black", linetype="dashed")
```



1.3.1.9 Number of seats

For cars with 5 seats (which represents 79% of the cars in the dataset), the proportion of sold cars seems to be similar to the 25% overall proportion for the dataset. It seems unlikely that we can use the number of seats a car has to predict its sold status.

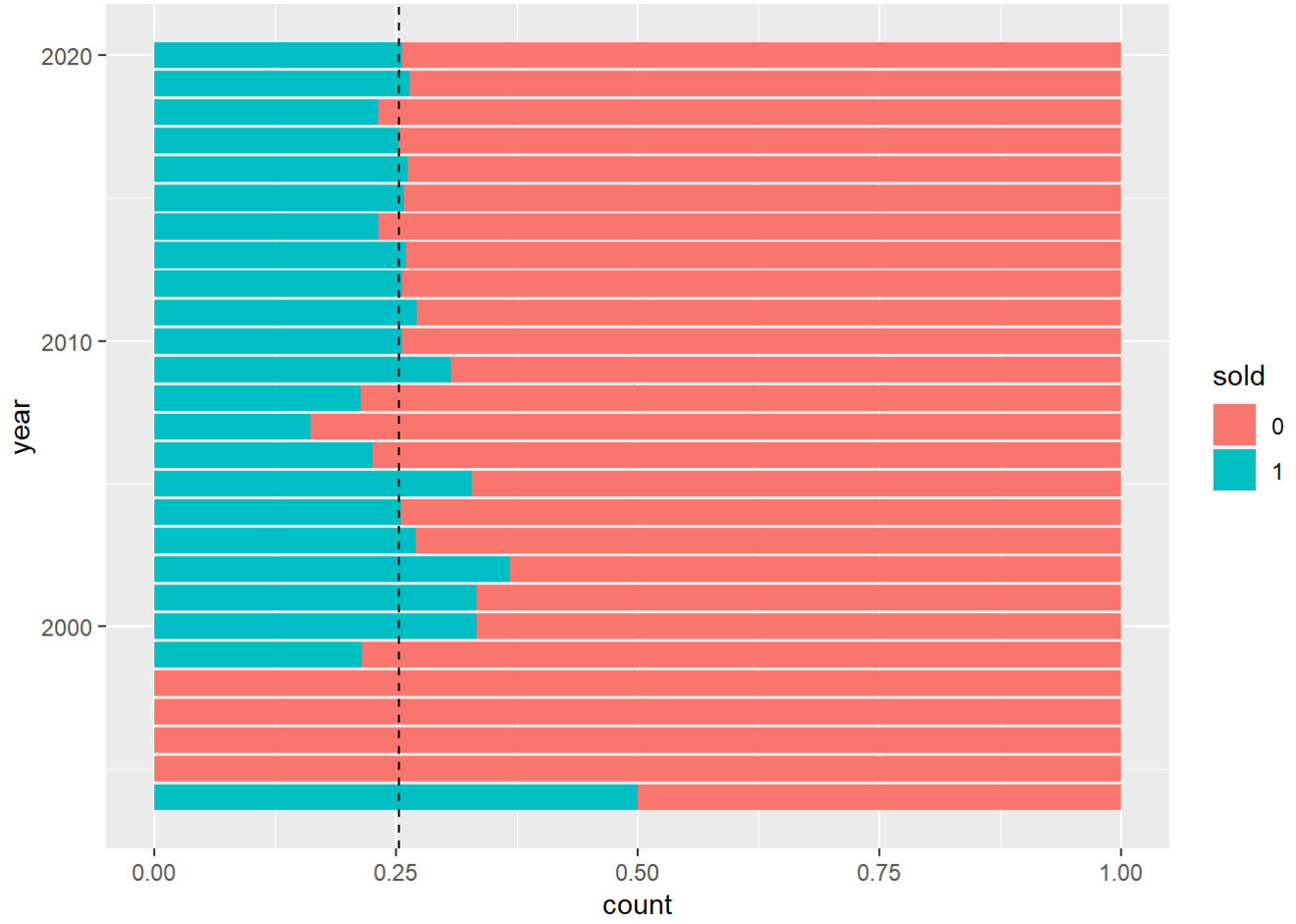
```
ggplot(usedcardata, aes(y = seats, fill = sold)) +
  geom_bar(position = "fill") +
  geom_vline(xintercept = sold_prop, color="black", linetype="dashed")
```



1.3.1.10 Year

In 1.2.2.2 earlier we see that most cars seem to be from 2010 or later years and for these years the the proportion of sold cars seems to be around the 25% overall proportion for the dataset. It seems unlikely that we can use a car's year to predict its sold status.

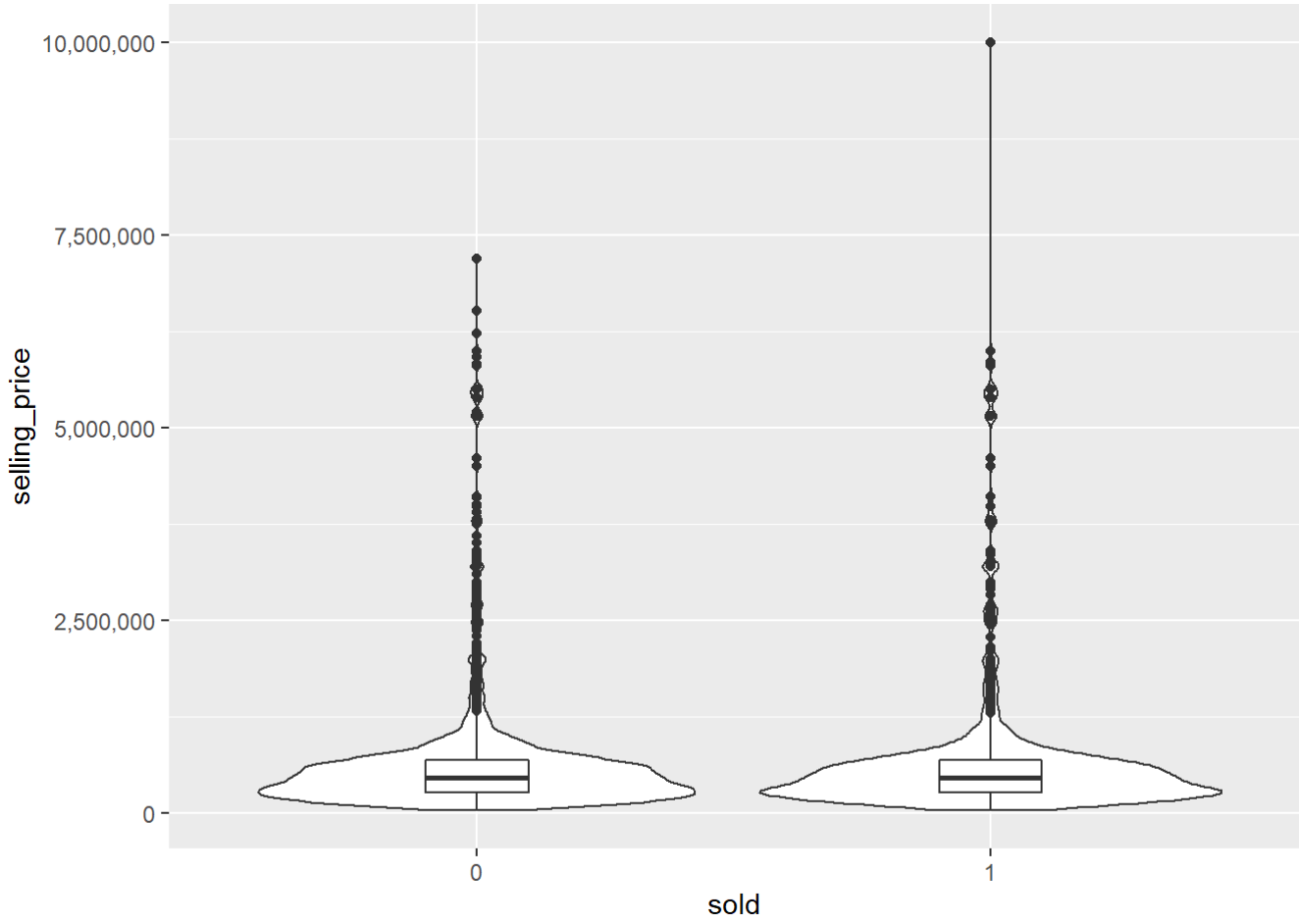
```
ggplot(usedcardata, aes(y = year, fill = sold)) +
  geom_bar(position = "fill") +
  geom_vline(xintercept = sold_prop, color="black", linetype="dashed")
```



1.3.1.11 Selling price

There doesn't seem to be any difference between the selling price data of sold and unsold cars. Based on the plot, it doesn't seem likely that we can use the selling price of a car to predict its sold status.

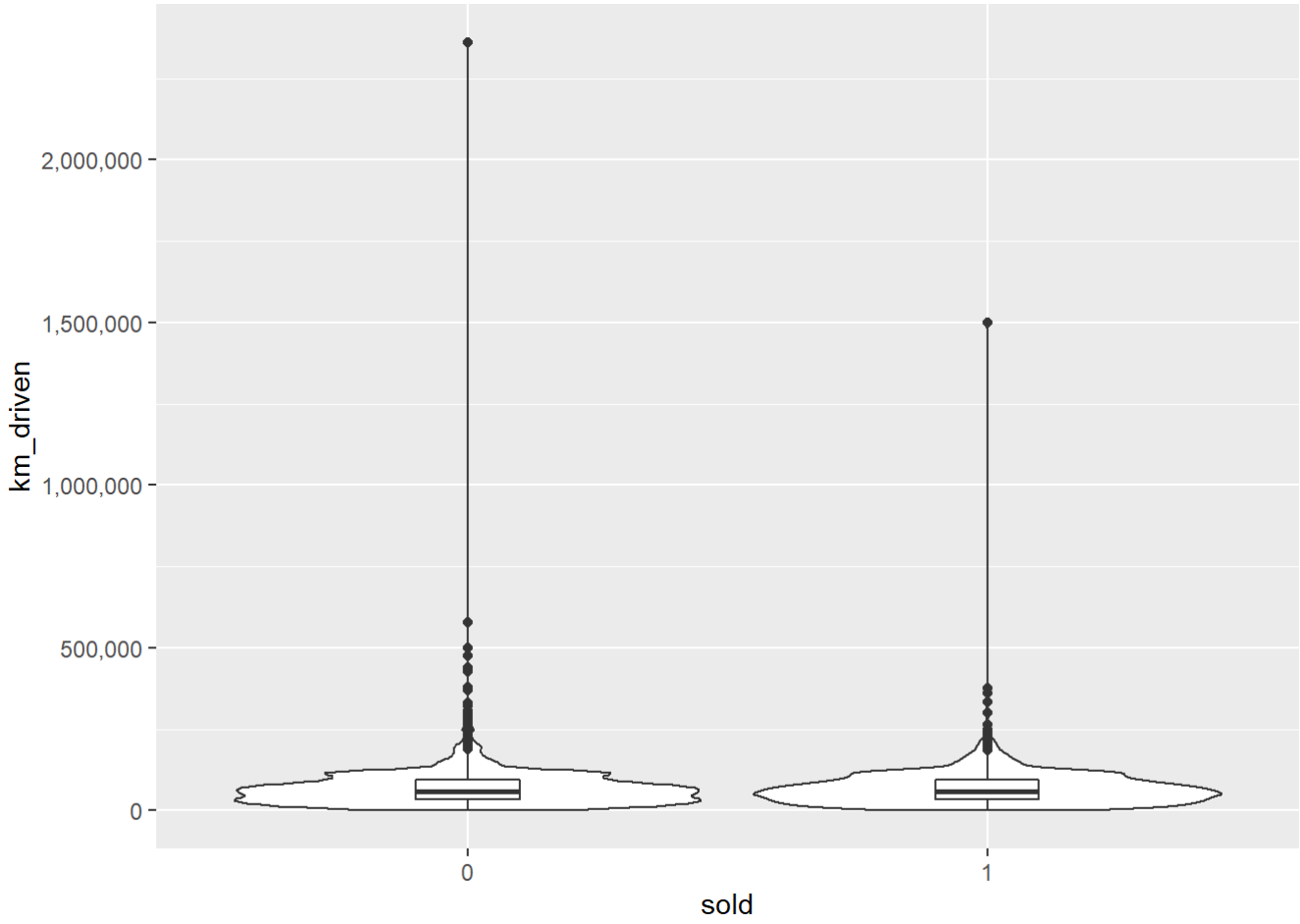
```
ggplot(usedcardata,
       aes(x = selling_price, y = sold)) +
  geom_violin() +
  scale_x_continuous(labels = comma) +
  coord_flip() +
  geom_boxplot(width=0.2)
```



1.3.1.12 Distance driven

There doesn't seem to be any difference between the distance driven data of sold and unsold cars. Based on the plot, it doesn't seem likely that we can use how many kilometers a car has been driven to predict its sold status.

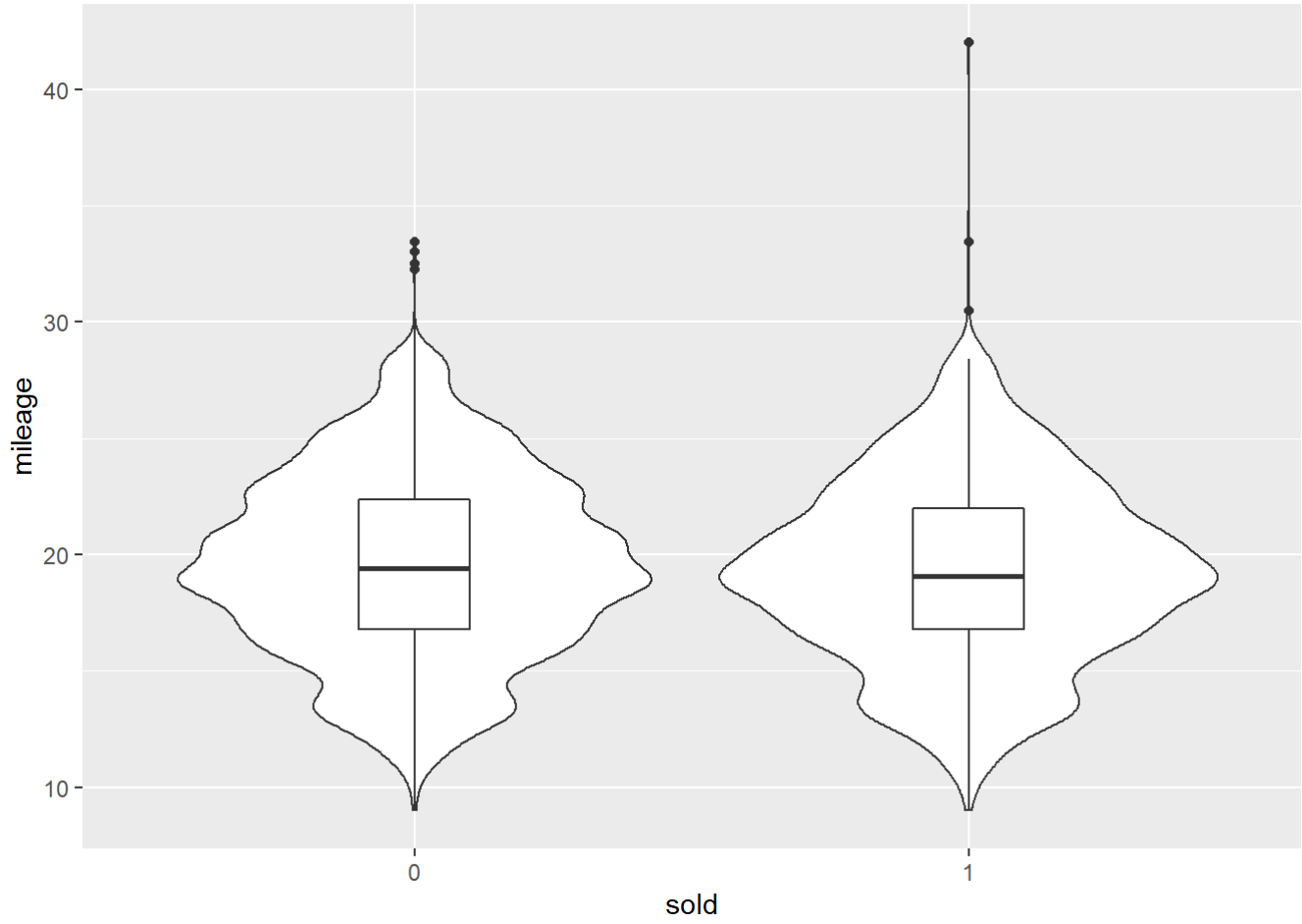
```
ggplot(usedcardata,
       aes(x = km_driven, y = sold)) +
  geom_violin() +
  scale_x_continuous(labels = comma) +
  coord_flip() +
  geom_boxplot(width=0.2)
```



1.3.1.13 Mileage

There doesn't seem to be any difference between the fuel efficiency data of sold and unsold cars. Based on the plot, it doesn't seem likely that we can use the fuel efficiency of a car to predict its sold status.

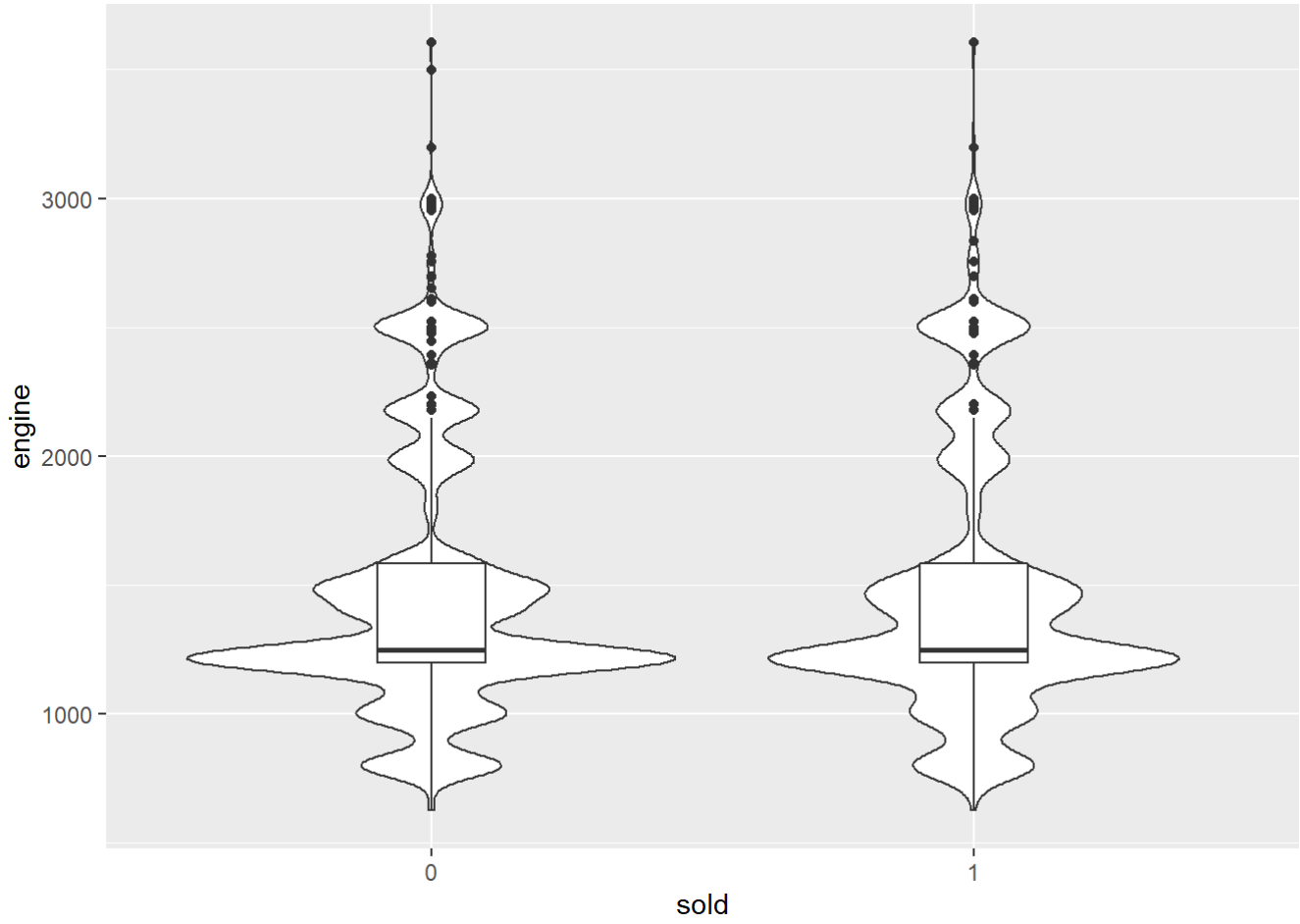
```
ggplot(usedcardata,
       aes(x = mileage, y = sold)) +
  geom_violin() +
  coord_flip() +
  geom_boxplot(width=0.2)
```



1.3.1.14 Engine size

There doesn't seem to be much difference between the engine size data of sold and unsold cars. Based on the plot, it doesn't seem likely that we can use a car's engine size to predict its sold status.

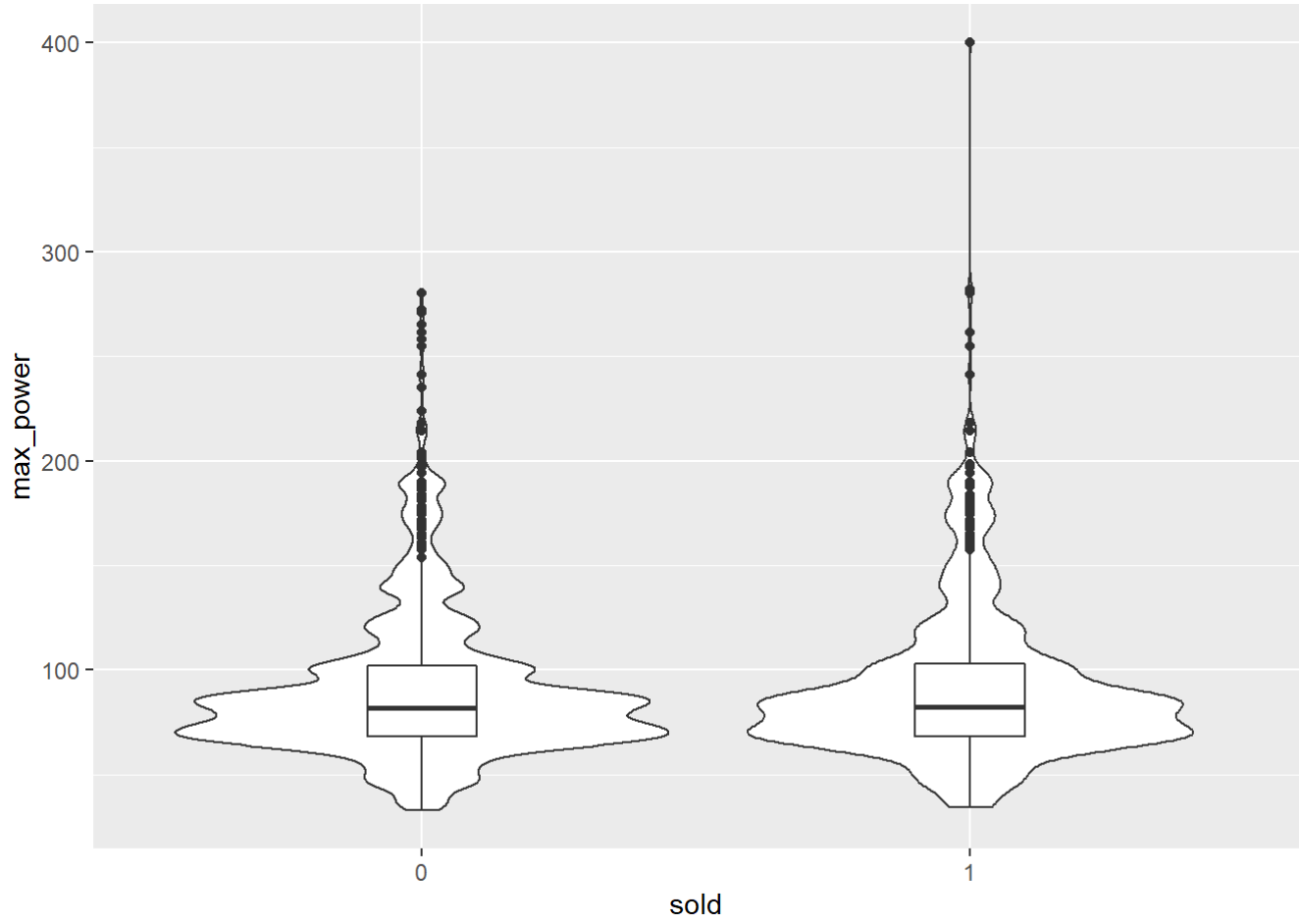
```
ggplot(usedcardata,
       aes(x = engine, y = sold)) +
  geom_violin() +
  coord_flip() +
  geom_boxplot(width=0.2)
```



1.3.1.15 Engine max power

There doesn't seem to be any difference between the maximum engine output data of sold and unsold cars. Based on the plot, it doesn't seem likely that we can use the engine power output of a car to predict its sold status.

```
ggplot(usedcardata,
       aes(x = max_power, y = sold)) +
  geom_violin() +
  coord_flip() +
  geom_boxplot(width=0.2)
```



Based on the above, it seems that region, state/province and city seem to be the most suitable variables to train our models.

2. Modeling

Introduction

Machine learning involves building models that can make decisions or predict the value of variables. These models are built using machine learning algorithms that use datasets to “learn” or identify patterns or relationships that are then used for making predictions or decisions.

Machine learning techniques are typically classed as either supervised or unsupervised. Supervised learning involves using labeled datasets that help train algorithms to classify data which can then be used to make accurate predictions. Unsupervised learning involves algorithms that analyze unlabeled datasets to find patterns which they can then use to make predictions.

Popular techniques

Popular machine learning techniques include linear regression, logistic regression, k nearest neighbors (knn) and decision trees. All four are a supervised type of machine learning techniques.

Linear regression involves predicting the value of a dependent variable (y) based on one or more independent variables (x) in a paired set of data by estimating the coefficients of a linear equation and fitting a straight line that minimizes the distance between the predicted outcome and the actual outcome (least squares method). With linear regression, the variables need to be measured at a continuous level. Linear regression is unsuitable for this project as the aim is to predict a categorical outcome (sold or unsold).

Whilst linear regression is used for modeling relationships between continuous variables and predicting the value of a continuous dependent variable, logistic regression can used for modeling the relationship between categorical variables and predicting the probability of a categorical outcome (0 or 1, yes or no).

knn uses the proximity of an individual data point to its specified k-number of neighboring points to make a prediction of its value (regression) or its category (classification). This technique has higher computation requirements as all the processing is done during testing or making predictions.

Decision trees use simple yes-no/if-else decision rules based on the features of a data set to predict the value of, or categorize, a variable. The predictions in the decision trees are referred to as nodes. This method is comparatively faster than knn.

Modeling plan

Four logistic regression models will be trained first:

1. Using the region as the predictor
2. Using the state/province as the predictor
3. Using the city as the predictor
4. Using the region, state/province and city combined as predictors

The best predictor/s will then be determined based on the accuracy and f1 score of the four above models, and subsequently be used to train a model using the decision tree technique.

Accuracy is measured by calculating the proportion of outcomes that are correctly predicted in the test set.

The f1 score is the harmonic mean of Precision and Recall, and is used as a one number summary in machine learning tasks. Precision is the percentage of correct predicted positive outcomes out of all the predicted positive outcomes (true positive + false positive). Recall is the percentage of positive outcomes the model succeeded in predicting.

The decision tree technique is preferred over knn due to the lower computation requirements, and also the fact that a decision tree model can be plotted to provide a graphical representation of the decision rules. The ultimate aim of machine learning projects such as this is not purely academic, it is also to help industry users make informed decisions where a visual decision tree can be more useful.

The accuracy of a basic model that guesses the outcome (sold or unsold) at random will serve as a benchmark for the other models.

2.1 Creating training and test sets

Before training the models, it is first necessary to split the dataset into two - a training set to train the models and a test set to measure the performance of the models.

There is no “optimum” ratio for splitting a dataset, however, commonly used ratios for splitting a dataset into training and testing sets include 50:50, 60:40, 70:30, 80:20 and 90:10. If the training set is not sufficiently large enough, the model may as a result show a high variance during training and likewise if the test set is not sufficiently large enough the performance value may show a high variance.

For this project the dataset will be split using a 80:20 ratio (80% training and 20% testing). As the dataset contains 7906 observations, this will result in a training set with just over 6300 observations and a test set with just over 1500 observations which should be large enough for their respective purposes.

```
set.seed(1, sample.kind = "Rounding")

test_index <- createDataPartition(usedcardata$sold, times = 1, p = 0.2, list = FALSE)
test_set <- usedcardata[test_index,]
train_set <- usedcardata[-test_index,]

nrow(train_set)/nrow(usedcardata)
```

```
## [1] 0.7998988
```

The ratio of sold:unsold cars in the training set is in line with the ratio for the whole dataset:

```
mean(train_set$sold == 1)
```

```
## [1] 0.2530044
```

2.2 Basic model (random guessing)

This model predicts if a car is sold or unsold purely by guessing the outcome.

The accuracy is 48%:

```
set.seed(2, sample.kind = "Rounding")

base <- sample(c(0,1), nrow(test_set), replace = TRUE)

mean(base == test_set$sold)
```

```
## [1] 0.482933
```

```
model_accuracy <- tibble(model = "Basic", accuracy = mean(base == test_set$sold))
```

2.3 Logistic regression

2.3.1 Region

This model is trained using the “region” variable as the sole predictor.

The accuracy is 75% - an improvement on the basic model:

```
set.seed(1, sample.kind = "Rounding")

# model
train_glm_region <- train(sold ~ region,
                          method = "glm",
                          data = train_set)

# accuracy
pred_glm_region <- predict(train_glm_region, test_set)

cm_region_glm <- confusionMatrix(data = pred_glm_region,
                                 reference = test_set$sold)

# add to table
model_accuracy <- model_accuracy %>% add_row(model = "Region GLM", accuracy = cm_region_glm$o
verall["Accuracy"])
model_f1 <- tibble(model = "Region GLM", f1 = cm_region_glm$byClass["F1"])

cm_region_glm$table
```

```
##           Reference
## Prediction    0    1
##          0 1182  400
##          1    0    0
```

```
cm_region_glm$overall["Accuracy"]
```

```
##  Accuracy
## 0.7471555
```

2.3.2 State/Province

This model is trained using the “state or province” variable as the sole predictor.

The accuracy is also 75%:

```
set.seed(1, sample.kind = "Rounding")

# model
train_glm_state <- train(sold ~ state_province,
                         method = "glm",
                         data = train_set)

# accuracy
pred_glm_state <- predict(train_glm_state, test_set)

cm_state_glm <- confusionMatrix(pred_glm_state, test_set$sold)

# add to table
model_accuracy <- model_accuracy %>% add_row(model = "State/Province GLM", accuracy = cm_stat
e_glm$overall["Accuracy"])
model_f1 <- model_f1 %>% add_row(model = "State/Province GLM", f1 = cm_state_glm$byClass["F1"
])

cm_state_glm$table
```

```
##           Reference
## Prediction    0    1
##          0 1133  340
##          1   49   60
```

```
cm_state_glm$overall["Accuracy"]
```

```
##  Accuracy
## 0.7541087
```

2.3.3 City

This model is trained using the “city” variable as the sole predictor.

The accuracy is 92% - an improvement of 17 percentage points over the other variables:

```
set.seed(1, sample.kind = "Rounding")

# model
train_glm_city <- train(sold ~ city,
                        method = "glm",
                        data = train_set)

# accuracy
pred_glm_city <- predict(train_glm_city, test_set)

cm_city_glm <- confusionMatrix(pred_glm_city, test_set$sold)

# add to table
model_accuracy <- model_accuracy %>% add_row(model = "City GLM", accuracy = cm_city_glm$overa
ll["Accuracy"])
model_f1 <- model_f1 %>% add_row(model = "City GLM", f1 = cm_city_glm$byClass["F1"])

cm_city_glm$table
```

```
##           Reference
## Prediction    0    1
##          0 1128   66
##          1   54  334
```

```
cm_city_glm$overall["Accuracy"]
```

```
##  Accuracy
## 0.9241466
```

2.3.4 Location (region & state/provide & city)

This model is trained using all the three variables regarding the car's location - region, state/province and city.

The accuracy is 94% - the best performance:

```
set.seed(1, sample.kind = "Rounding")

# model
train_glm_location <- train(sold ~ region + state_province + city,
                            method = "glm",
                            data = train_set)

# accuracy
pred_glm_location <- predict(train_glm_location, test_set)

cm_location_glm <- confusionMatrix(pred_glm_location, test_set$sold)

# add to table
model_accuracy <- model_accuracy %>% add_row(model = "Location GLM", accuracy = cm_location_g
lm$overall["Accuracy"])
model_f1 <- model_f1 %>% add_row(model = "Location GLM", f1 = cm_location_glm$byClass["F1"])

cm_location_glm$table
```

```
##            Reference
## Prediction    0    1
##          0 1137   49
##          1   45  351
```

```
cm_location_glm$overall["Accuracy"]
```

```
##  Accuracy
## 0.9405815
```

2.4 Decision Trees

2.4.1 Location (region & state/provide & city)

As the best performing logistic regression model is the one trained using all the three location variables, a model using the decision tree technique is trained using region, state/province and city as the predictors.

21 values between 0 and 0.02 are used for the complexity parameter (cp) to tune the model.

The complexity parameter (cp) controls the complexity of the decision tree by defining the minimum improvement in the model required at each node. A cp value of 0 would yield the most complex decision tree.

The accuracy is 87% - 7 percentage points below the logistic regression model trained with the same predictors:

```
set.seed(2, sample.kind = "Rounding")

rpart_tunegrid <- data.frame(cp = seq(0, 0.02, 0.001))

# model
train_rpart_location <- train(sold ~ region + state_province + city,
                              method = "rpart",
                              tuneGrid = rpart_tunegrid,
                              data = train_set)

# accuracy
pred_rpart_location <- predict(train_rpart_location, test_set)

cm_location_rpart <- confusionMatrix(pred_rpart_location, test_set$sold)

# add to table
model_accuracy <- model_accuracy %>% add_row(model = "Location RPART", accuracy = cm_location
_rpart$overall["Accuracy"])
model_f1 <- model_f1 %>% add_row(model = "Location RPART", f1 = cm_location_rpart$byClass["F
1"])

cm_location_rpart$table
```

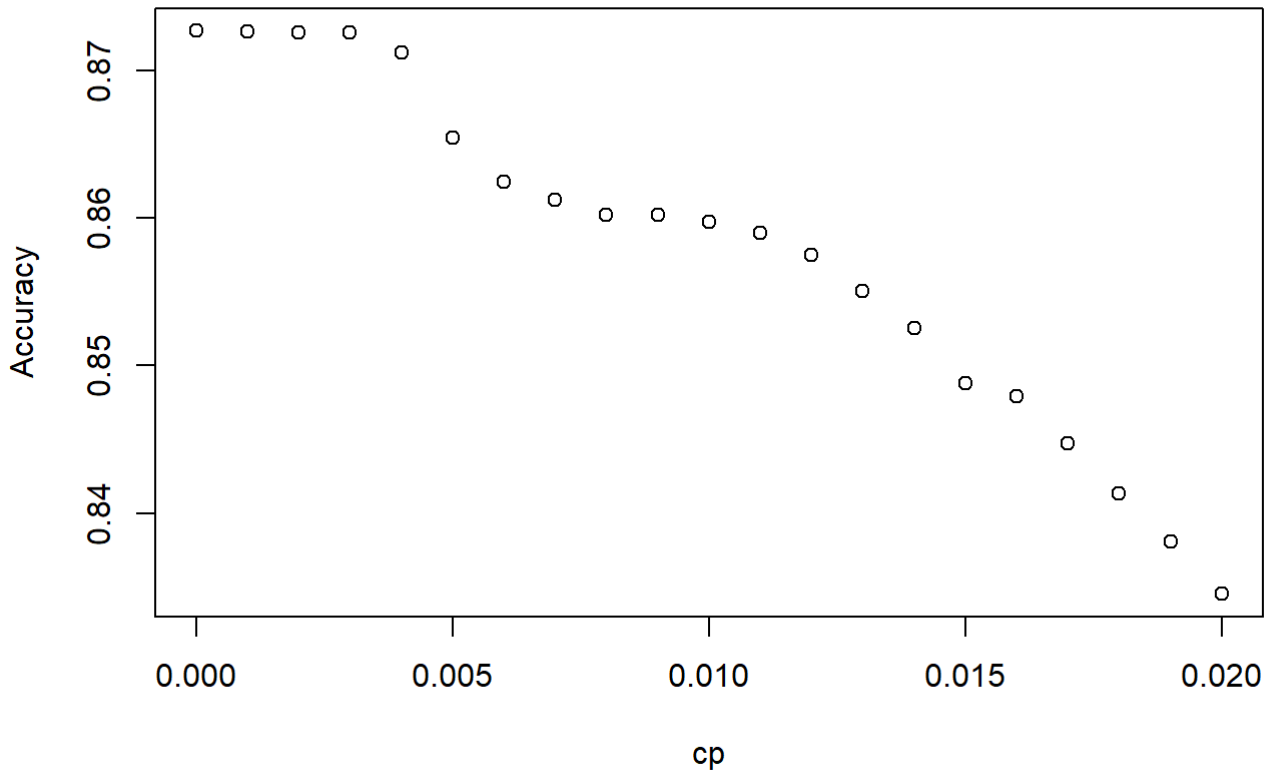
```
##           Reference
## Prediction    0    1
##          0 1163  188
##          1   19  212
```

```
cm_location_rpart$overall["Accuracy"]
```

```
## Accuracy
## 0.869153
```

The best performing cp value is 0:

```
# best cp
train_rpart_location$results %>%
  select(cp, Accuracy) %>%
  plot()
```

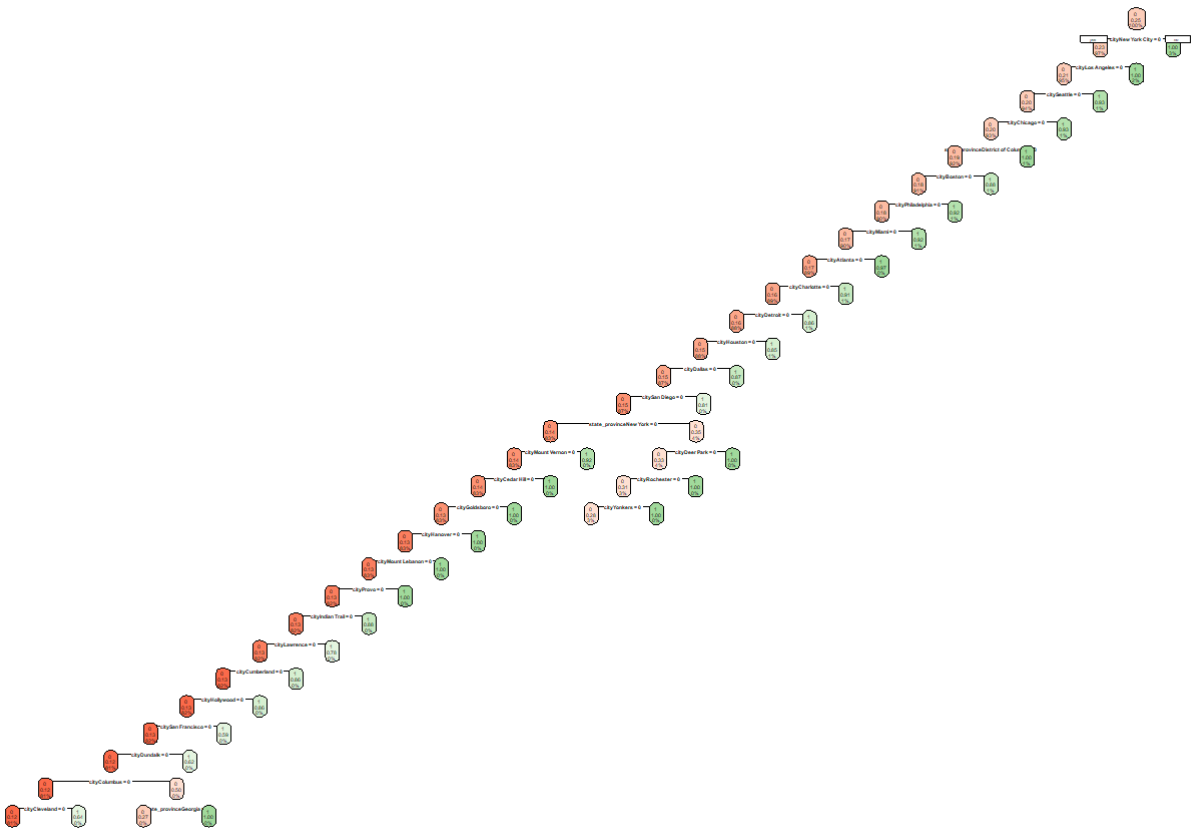



```
train_rpart_location$bestTune
```

```
##   cp
## 1  0
```

Plotting the decision tree yields a very complex tree that is undecipherable. This is not surprising considering that the decision tree is based on a model tuned with a cp value of 0.

```
rpart.plot(train_rpart_location$finalModel,
           type = 2,
           fallen.leaves = FALSE,
           box.palette="RdGn")
```



2.4.2 Location 2 (higher cp, lower complexity)

One of the reasons for using the decision tree technique is to obtain a visual decision tree that can be used by an end user to inform their decision making. Although the model trained in 2.4.1 is more accurate than the base model relying on guessing, the resulting decision tree is unfit for use.

To yield a more decipherable decision tree, a new model is trained using the same three location variables as 2.4.2, but with a higher range of cp values between 0.03 and 0.05. This reduces the complexity of the model which will result in a model with a lower accuracy compared to the model in 2.4.1, but hopefully yield a decision tree that is more understandable.

The accuracy of the decision tree model and the usability of its resulting decision tree seem to be inversely related so it is essential to find a cp value that provides a balance between the two.

The accuracy of the model is 82%:

```
# model
set.seed(2, sample.kind = "Rounding")

train_rpart_location_2 <- train(sold ~ region + state_province + city,
                                method = "rpart",
                                tuneGrid = data.frame(cp = seq(0.03, 0.05, 0.001)),
                                data = train_set)

# accuracy
pred_rpart_location_2 <- predict(train_rpart_location_2, test_set)

cm_location_rpart_2 <- confusionMatrix(pred_rpart_location_2, test_set$sold)

# add to table
model_accuracy <- model_accuracy %>% add_row(model = "Location RPART 2", accuracy = cm_locati
on_rpart_2$overall["Accuracy"])
model_f1 <- model_f1 %>% add_row(model = "Location RPART 2", f1 = cm_location_rpart_2$byClass
["F1"])

cm_location_rpart_2$table
```

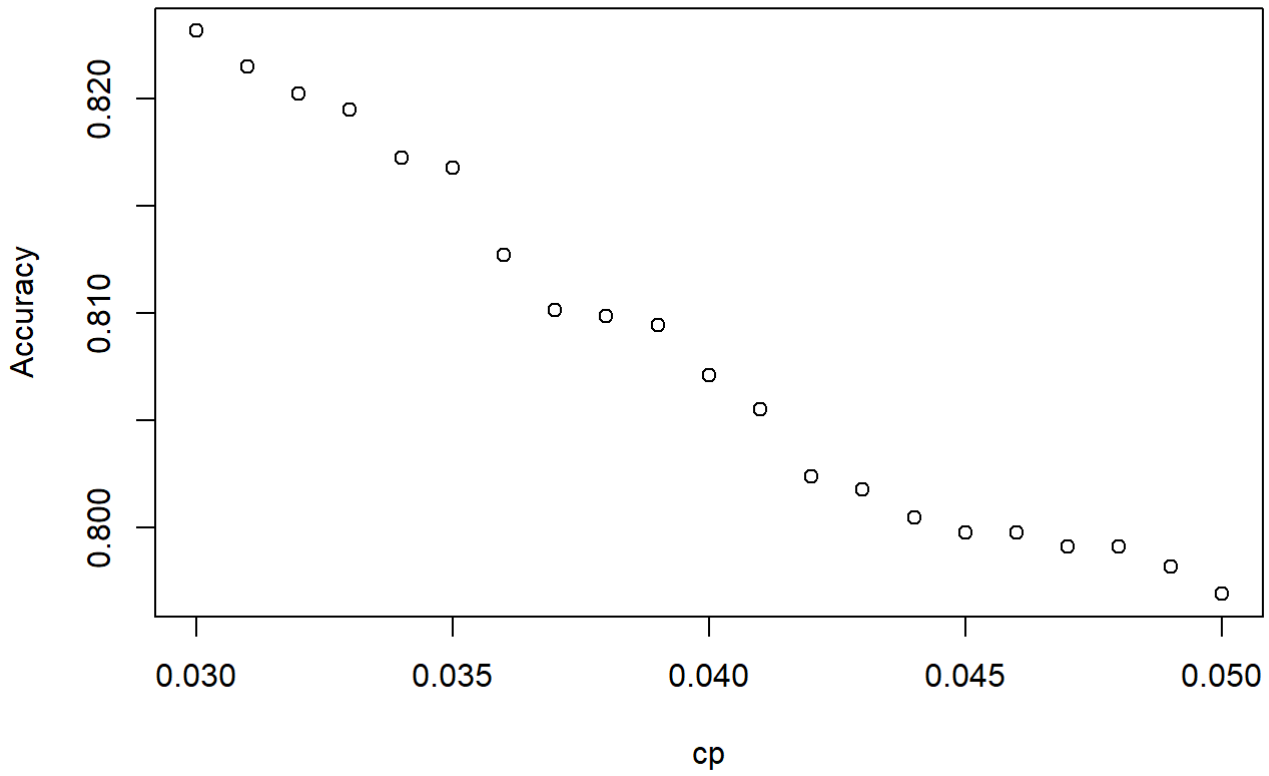
```
##           Reference
## Prediction    0    1
##          0 1181  286
##          1    1  114
```

```
cm_location_rpart_2$overall["Accuracy"]
```

```
##  Accuracy
## 0.8185841
```

The best performing cp value is 0.03:

```
# best cp
train_rpart_location_2$results %>%
  select(cp, Accuracy) %>%
  plot()
```



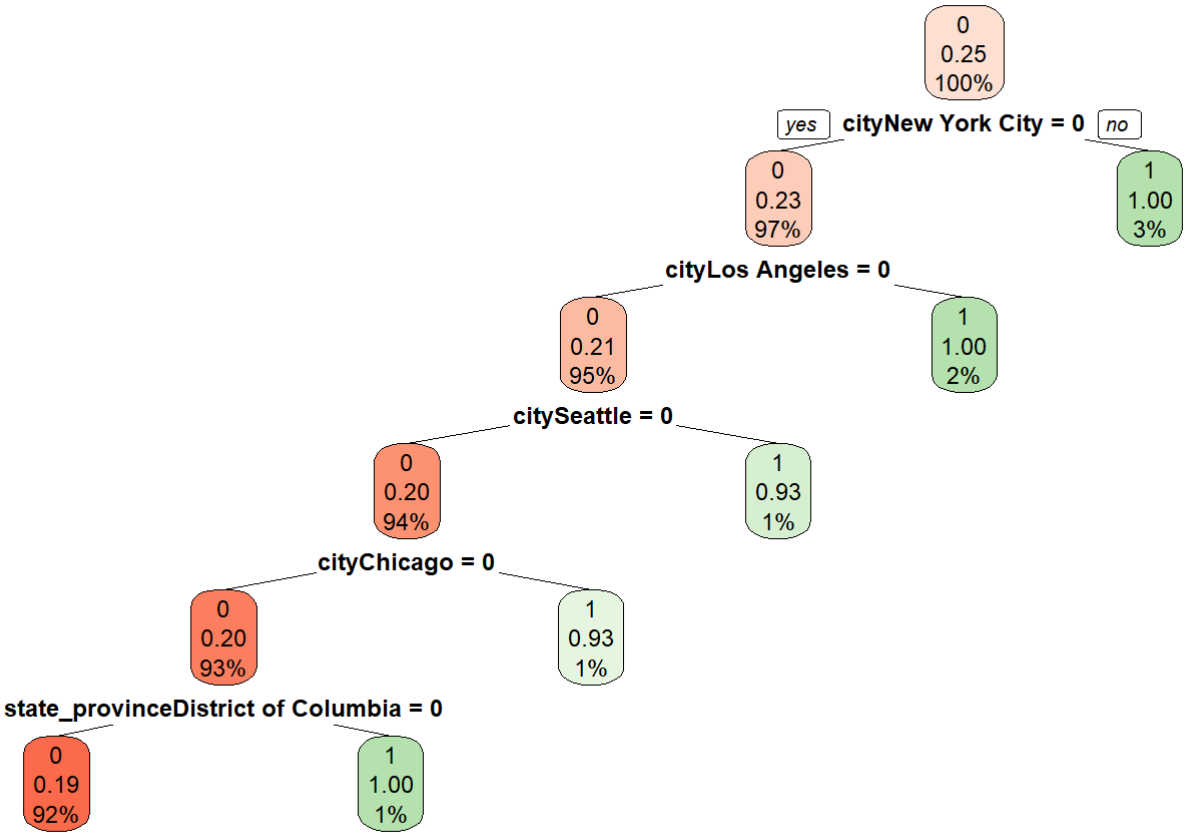

```
train_rpart_location_2$bestTune
```

```
##     cp
## 1 0.03
```

Plotting the decision tree yields a tree that is a lot more easy to understand.

A sacrifice of 5 percentage points in terms of the accuracy of the model (still 34 percentage points better than guessing) has resulted in a more usable decision tree:

```
rpart.plot(train_rpart_location_2$finalModel,
           type = 2,
           fallen.leaves = FALSE,
           box.palette="RdGn")
```



2.5 Model performance

In terms of accuracy, all models trained using either the logistic regression or decision tree method outperform the basic model based on guessing the outcome:

```
model_accuracy
```

```
## # A tibble: 7 x 2
##   model              accuracy
##   <chr>                 <dbl>
## 1 Basic                 0.483
## 2 Region GLM            0.747
## 3 State/Province GLM    0.754
## 4 City GLM              0.924
## 5 Location GLM          0.941
## 6 Location RPART        0.869
## 7 Location RPART 2      0.819
```

The model with the best performance is the one utilizing the logistic regression method and trained using the region, state/province and city variables together as predictors.

The same is true when assessing the performance of the models based on the f1 score:

```
model_f1
```

```
## # A tibble: 6 x 2
##   model                 f1
##   <chr>              <dbl>
## 1 Region GLM         0.855
## 2 State/Province GLM 0.853
## 3 City GLM           0.949
## 4 Location GLM       0.960
## 5 Location RPART     0.918
## 6 Location RPART 2   0.892
```

In this project, using the same three variables, the logistic regression method resulted in a better performing model compared to the decision tree method based on both accuracy and f1 score.

Among the three logistic regression models trained using only one variable as a predictor, the best performing model was the one trained using city as a predictor. Looking at the decision tree for the decision tree model trained using region, state/province and city as predictors also seems to suggest that city is the more important than state/province or region. Therefore, individually, the city variable seems to be the best for predicting the outcome.

3. Conclusion

3.1 Summary

This project focused on designing models using two machine learning techniques (logistic regression and decision trees) to predict the selling outcome of a particular car on sale.

The suitability of variables as potential predictors was assessed through analysis and visualization. Three variables (region, state/province & city) were identified as most suitable.

Six models were trained (four using logistic regression and two using decision trees) and their performance was judged based on accuracy and f1 score. All models were benchmarked against a basic model that guessed the outcome.

The best performing model was trained using the logistic regression method with region, state/province and city as predictors.

3.2 Potential impact and limitations

This project's data analysis and its resulting insights have practical uses in the industry for second-hand automobile sales as it seems to suggest that the location where a car is put on sale has a greater impact on whether it is sold or not than the attributes of the car itself.

The models trained as part of this project can help with making informed decision regarding the best locations to put a car on sale to maximize their chances of being sold. The decision tree plot can serve as a visual aid to the decision making process.

Although the dataset used in this project has nearly 8000 car sales records and the models were trained using a training set with over 6000 records, it is possible that this dataset may not be statistically representative of all cars sales in the US.

Also, considering that all the records in this dataset are for cars on sale in the US, it would not be possible to apply the insights gained or the models trained in this project to the second-hand automobile sales industry in other countries.

3.3 Future work

The scope of this project was constrained due to limited computation resources.

Future work on this dataset could look at improving the best performing logistic regression model in this project by including additional variables for training.

Other computationally demanding learning techniques such as k-nearest neighbors or random forests could be considered for training models to see if they result in models that have accuracy higher than 94%.

Random forests in particular would be interesting as it would build on the decision tree model trained in this project by training multiple random decision trees and taking the average.

Bibliography

1. Baheti, P (2022) V7 *Train Test Validation Split: How To & Best Practices [2022]* https://www.v7labs.com/blog/train-validation-test-set (https://www.v7labs.com/blog/train-validation-test-set)
2. Brown, S. (2021) *Machine learning, explained* https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained (https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained)
3. Chang, W. (2022) *R Graphics Cookbook, 2nd edition* https://r-graphics.org/ (https://r-graphics.org/)
4. IBM *About Linear Regression* https://www.ibm.com/uk-en/topics/linear-regression (https://www.ibm.com/uk-en/topics/linear-regression)
5. IBM *What is k nearest neighbors algorithm?* https://www.ibm.com/uk-en/topics/knn (https://www.ibm.com/uk-en/topics/knn)
6. IBM *What is Logistic Regression* https://www.ibm.com/uk-en/topics/logistic-regression (https://www.ibm.com/uk-en/topics/logistic-regression)
7. Irizarry, R. (2022) *Introduction to Data Science* https://rafalab.github.io/dsbook/ (https://rafalab.github.io/dsbook/)
8. Oracle *Skewness* https://docs.oracle.com/cd/E57185_01/CBREG/ch03s02s03s01.html (https://docs.oracle.com/cd/E57185_01/CBREG/ch03s02s03s01.html)
9. Varghese, D. (2018) Towards Data Science *Comparative Study on Classic Machine learning Algorithms* https://towardsdatascience.com/comparative-study-on-classic-machine-learning-algorithms-24f9ff6ab222 (https://towardsdatascience.com/comparative-study-on-classic-machine-learning-algorithms-24f9ff6ab222)