Food Mart

2022-10-21

Loading Packages

```
library(tidyverse)
library(tidyr)
library(ggplot2)
library(dplyr)
#install.packages('corrplot')
library(corrplot)
library(xgboost)
library(caret)
library(splitTools)
library(ranger)
library(lightgbm)
library(caret)
library(ggplot2)
library(stats)
library(caret)
library(randomForest)
library(ggplot2)
```

About FoodMart:

Food Mart (CFM) is a chain of convenience stores in the United States. The private company's headquarters are located in Mentor, Ohio, and there are currently approximately 325 stores located in the US. Convenient Food Mart operates on the franchise system. Food Mart was the nation's third-largest chain of convenience stores as of 1988. The NASDAQ exchange dropped Convenient Food Mart the same year when the company failed to meet financial reporting requirements. Carden & Cherry advertised Convenient Food Mart with the Ernest character in the 1980s.

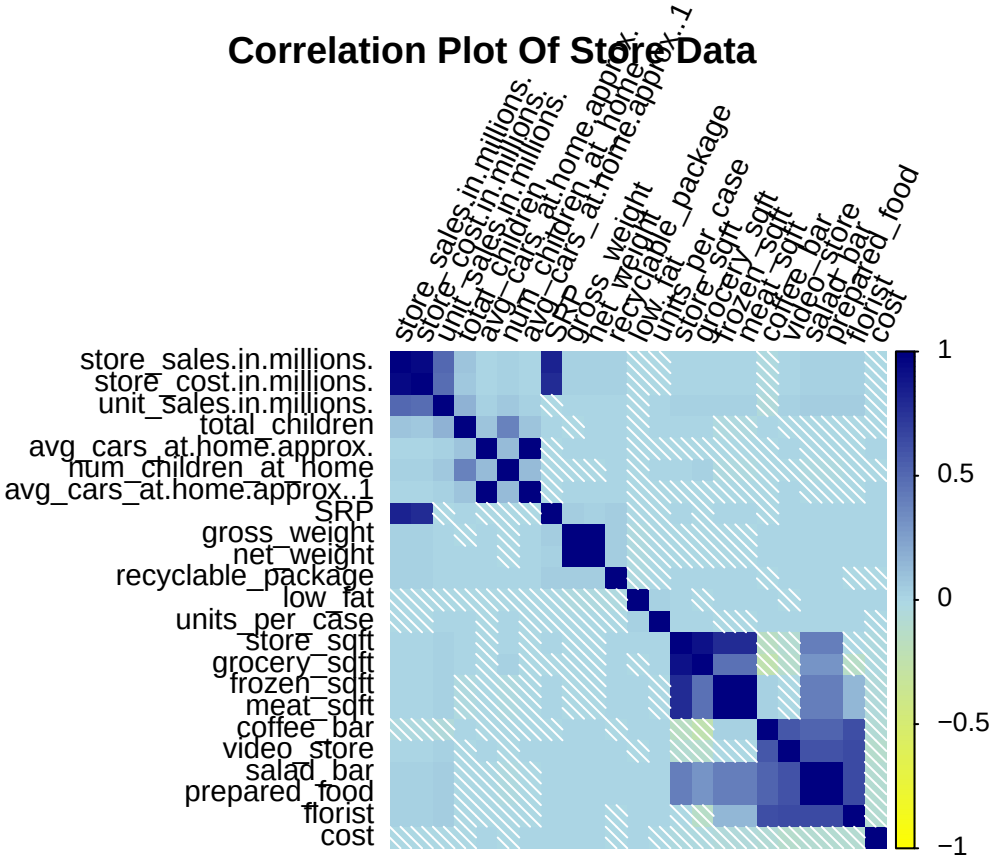
Loadin Data

```
#Load Data
CFM_data <- read.csv("C:/Users/bssup/Documents/fall22/R/project/Food_mart/data.csv")
```

Correlation Matrix

This will show correlation between different variables. Reasons of computing correlation matrix: To summarize a large amount of data where the goal is to see patterns. In our example above, the observable pattern is that all the variables highly correlate with each other.

```
#Correlation Matrix
store_data <- dplyr::select(CFM_data,c('store_sales.in.millions.','store_cost.in.millions.','unit_sales
#store_data
cor_data <- cor(store_data)
#head(cor_data)
corrplot(cor_data,method="shade",tl.col = "black",title = "\n\n Correlation Plot Of Store Data",tl.srt=
```

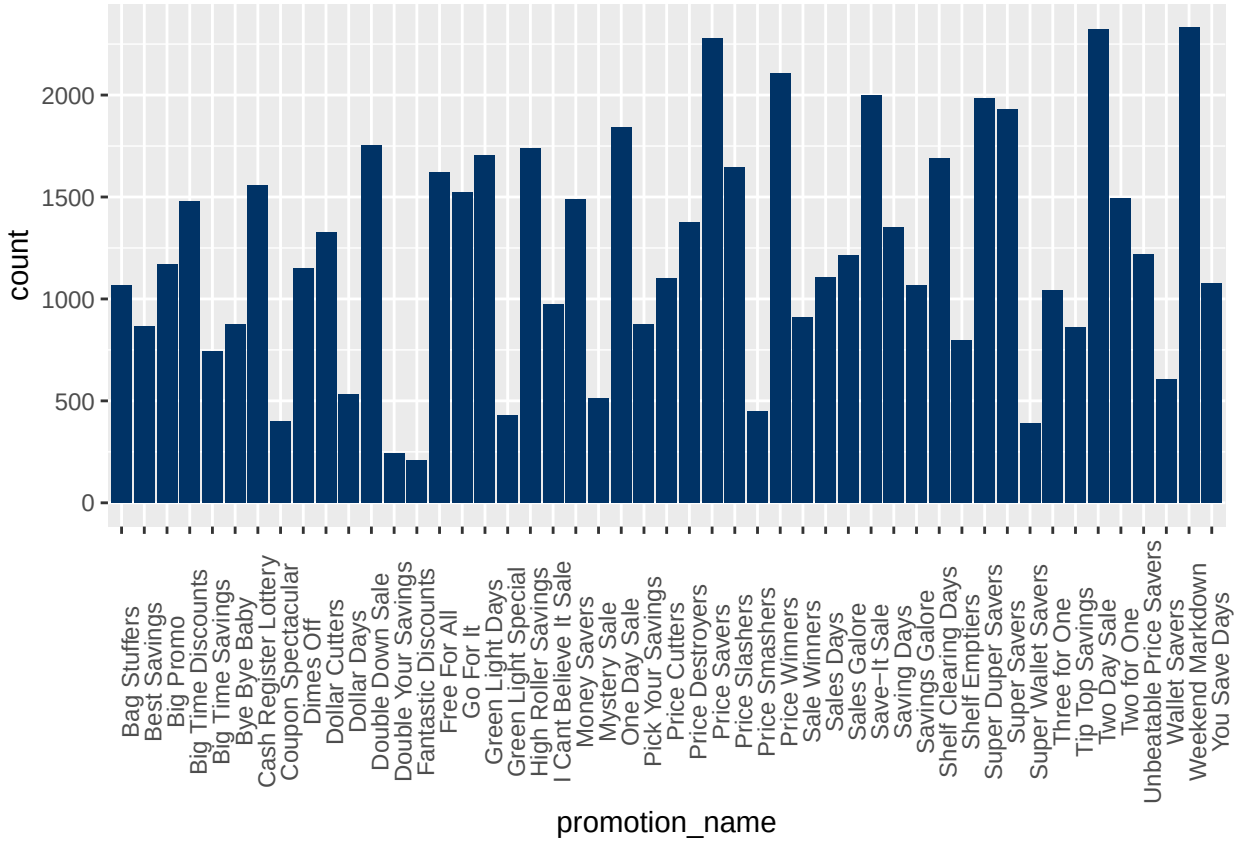



```
library(ggplot2)
library(reshape2)
data <- CFM_data
library(plotly)

fig <- plot_ly(data, x = ~education, color = ~gender, type = 'histogram',
               title = 'To visualize the education level of people',
               text = ~education) %>%
  layout(title = 'Sampled Results',
         bargap = 0.2,
         bargroupgap = 0.1,
         plot_bgcolor = 'pink')
```

```
fig
```

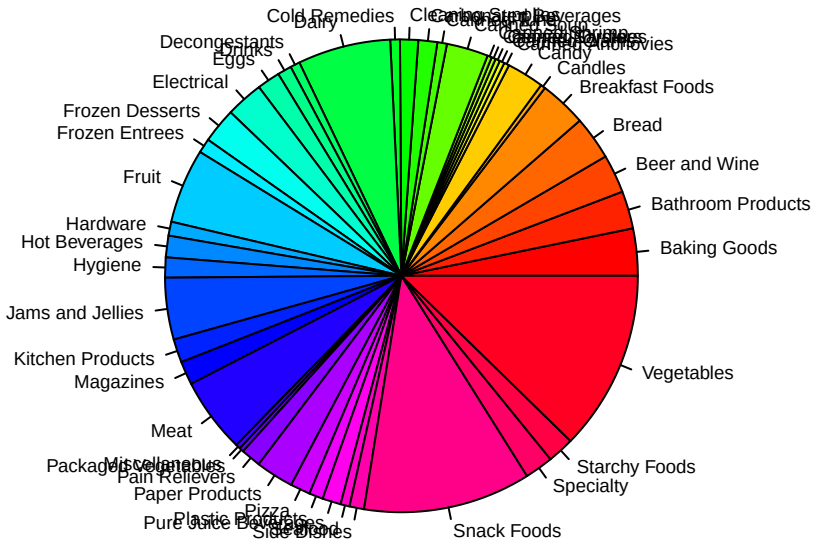
‘ qa
Count Plot Promotion Name

```
promotion_data <- CFM_data %>% dplyr::select(promotion_name) %>% group_by(promotion_name)
promo_count<-promotion_data %>% group_by(promotion_name)
ggplot(promo_count, aes(x=promotion_name),color=promotion_name) + geom_bar(fill = '#003366') + theme(ax
```



Pie Chat to Visualize the food_category in dataset

```
food_cat <- CFM_data %>% dplyr::select(food_category) %>% group_by(food_category)%>% count()
food_cat<- arrange(food_cat)
piepercent<- round(100*food_cat$n/sum(food_cat$n), 1)
#pie(food_cat$n,piepercent)



pie(food_cat$n, labels = food_cat$food_category, main = "Food Category pie chart",col = rainbow(length(
```

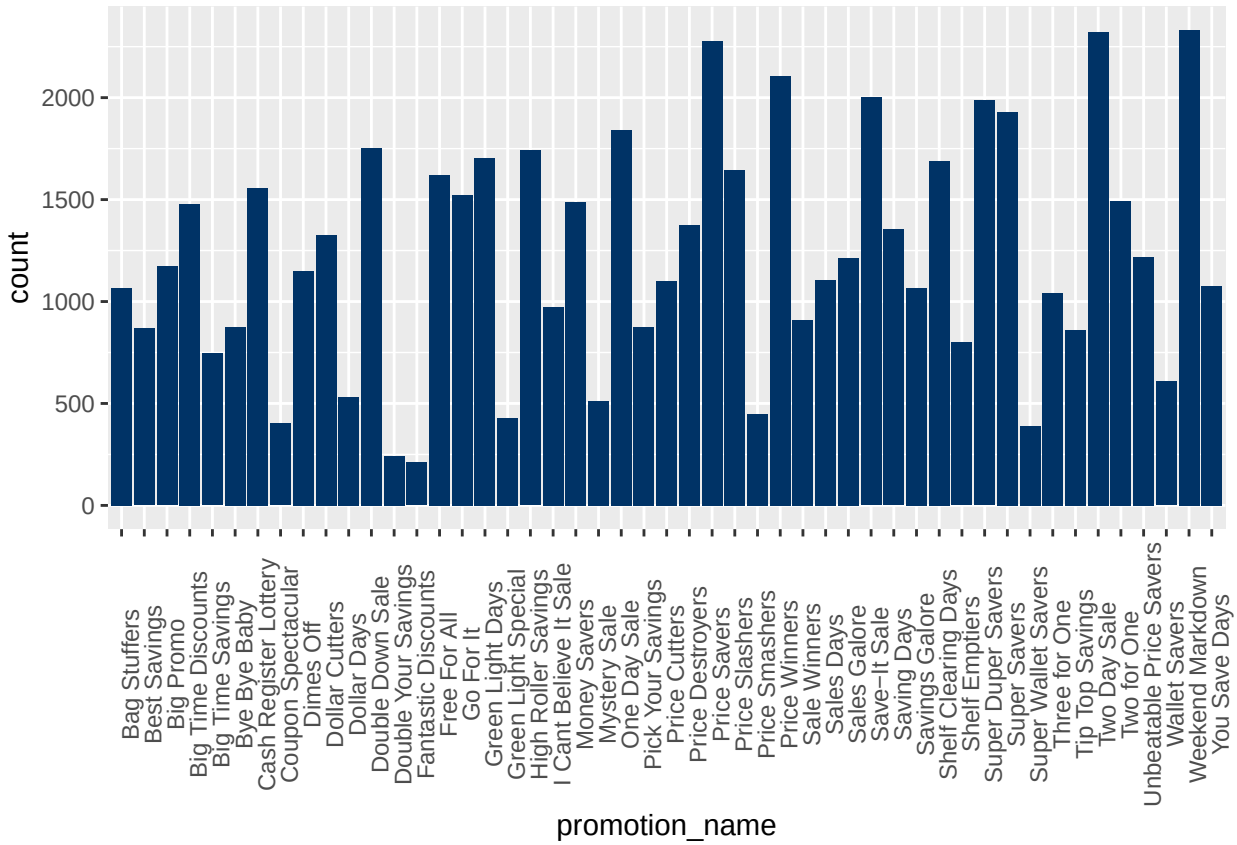


Observation:

From the above pie chart most used food_category such as 1.Vegetables 2.Snack Foods 3.Dairy these products we used in our daily life

To Visualize the promotion_names

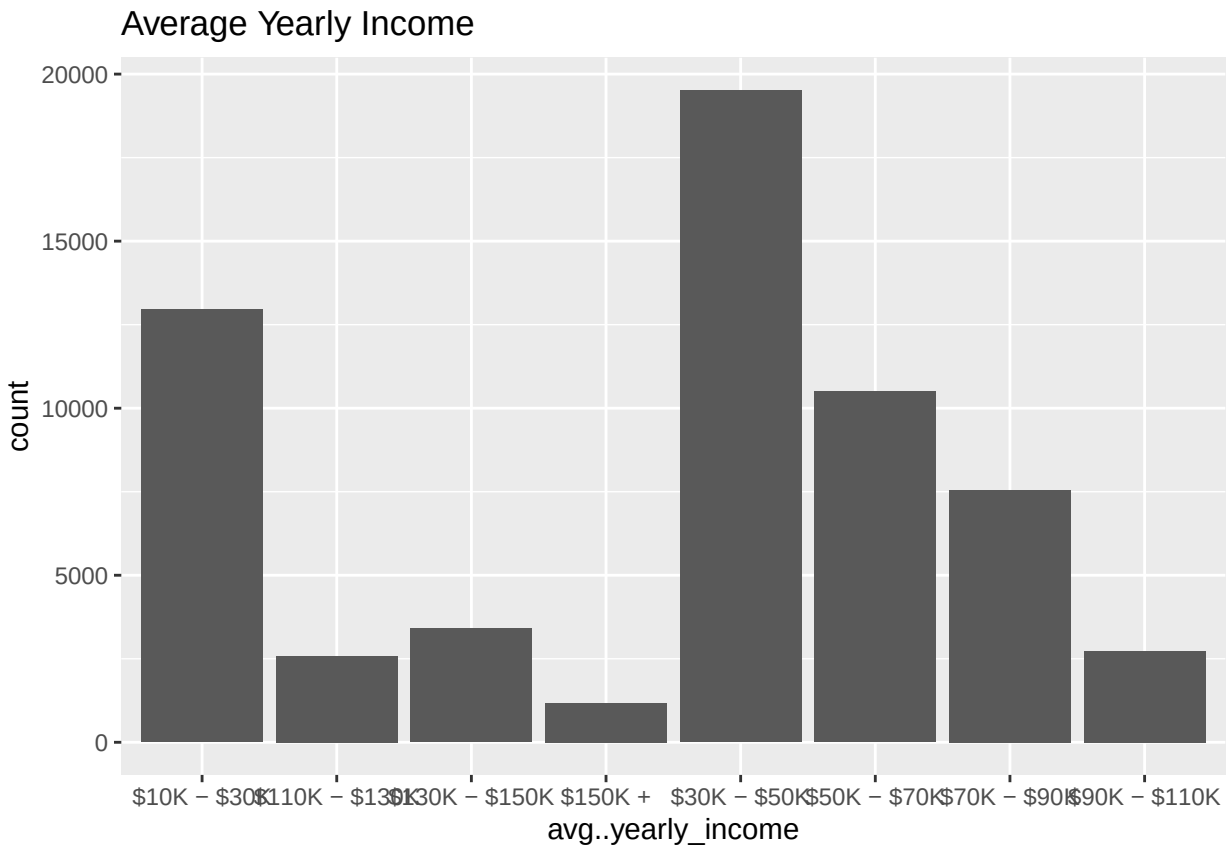
```
promotion_data <- CFM_data %>% dplyr::select(promotion_name) %>% group_by(promotion_name)
promo_count<-promotion_data %>% group_by(promotion_name)
#promo_count
ggplot(promo_count, aes(x=promotion_name),color=promotion_name) + geom_bar(fill = '#003366') + theme(ax
```



Observation From the above graph we found out that the most sales promotors names are: 1. Weekend Markdown 2. Two Day sales 3. Price Slashers

Visualize the Average yearly Income

```
avg_income_count <- CFM_data %>% dplyr::select(avg..yearly_income) %>% group_by(avg..yearly_income) %>%
order <- c("$10K - $30K","$30K - $50K","$50K - $70K","$70K - $90K","$90K - $110K","$110K - $130K","$130
#avg_income_count
ggplot(avg_income_count,aes(avg..yearly_income)) + geom_bar() + theme(axis.text.x = element_text(angle =
```

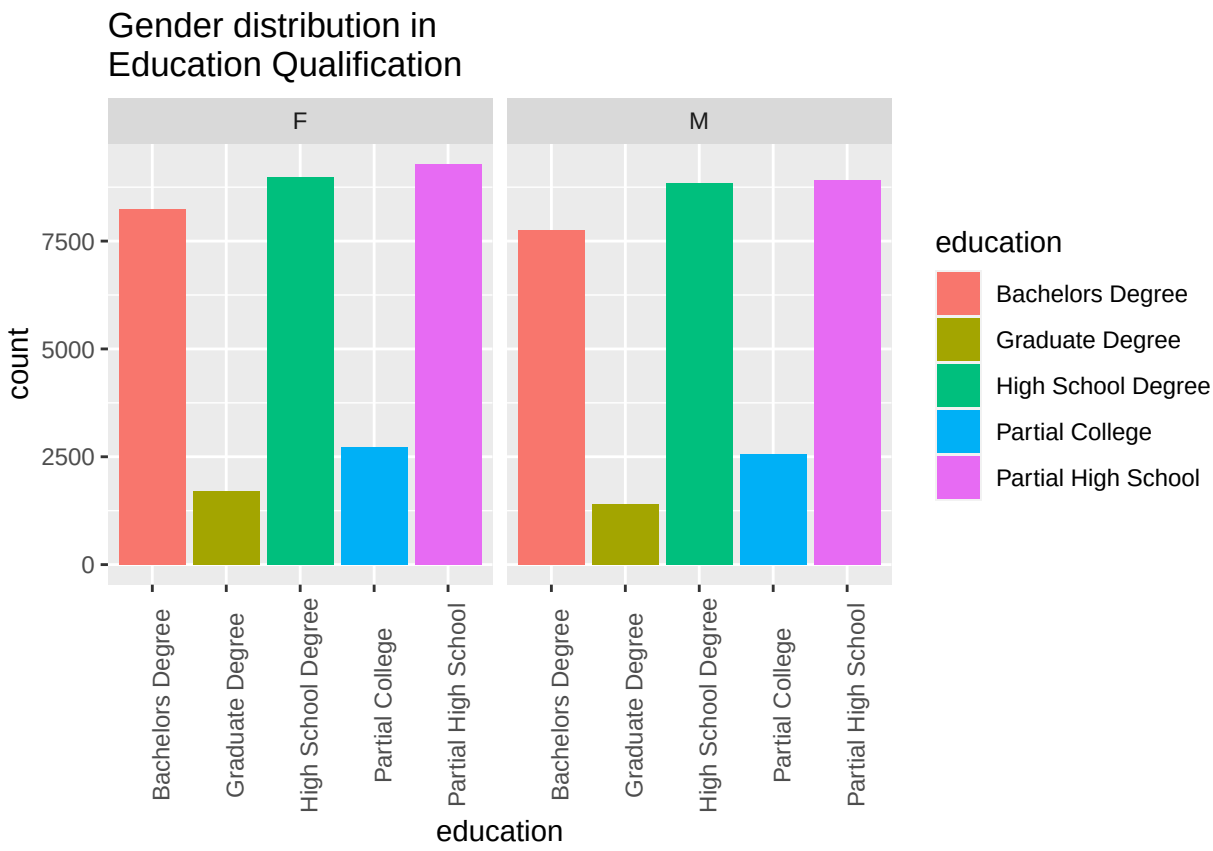


Observation

We found out that the average yearly income of the customers is $30k - $50k.

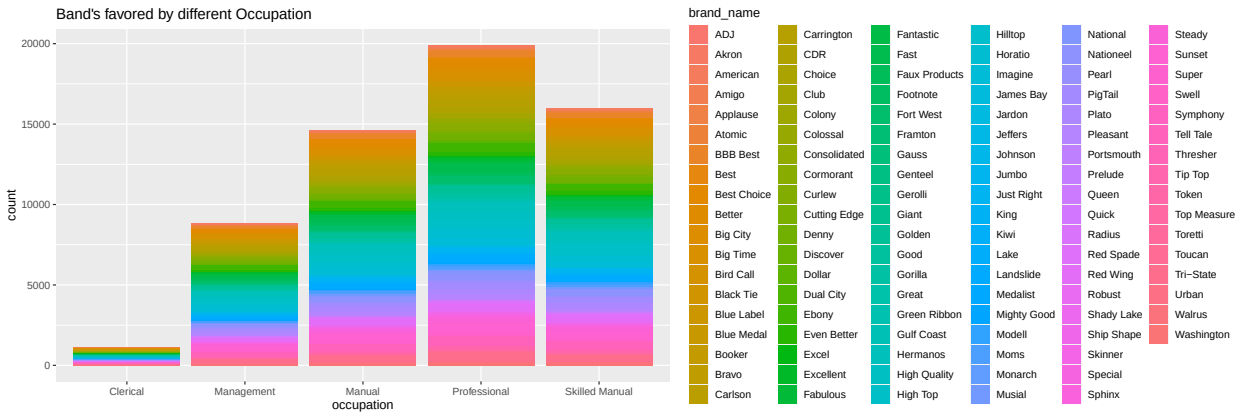
To Visualize the Education with gender in the dataset

```
Ed_gender <- CFM_data %>% dplyr::select(gender,education) %>% arrange(gender) %>% group_by(gender,educa
#Ed_gender %>% count()
ggplot(Ed_gender, aes(x=education,fill=education)) + geom_bar()+ theme(axis.text.x = element_text(angle
```



To Visualize occupation and brand_name divided among gender

```
brand_ocp <- CFM_data %>% dplyr::select(occupation,brand_name) %>% group_by(occupation)
  ggplot(brand_ocp, aes(x=occupation,fill=brand_name)) + geom_bar() + ggtitle("Band's favored by diff
```



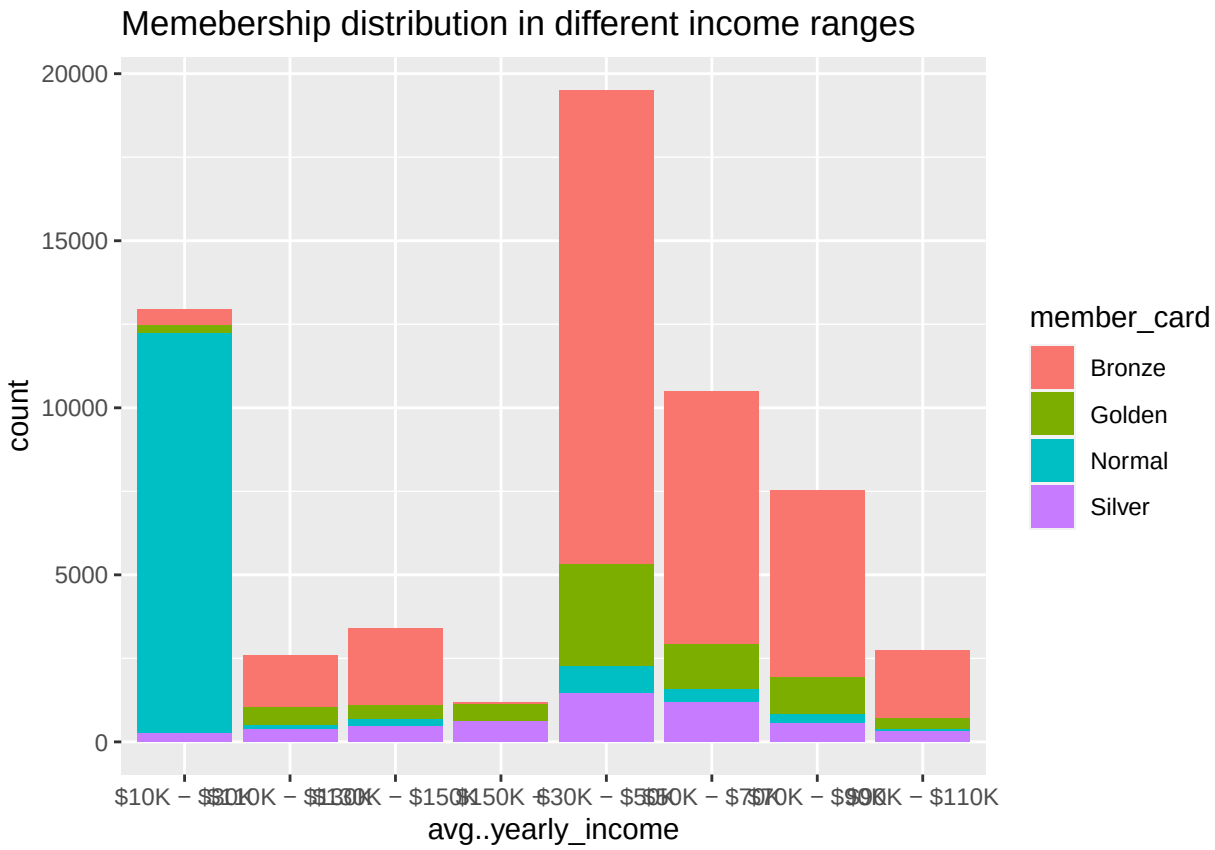
Observation

For Male: 1. Most male have the partial high school degree. 2. Females with graduate degree are low in number.

Visualize the yearly income with member card

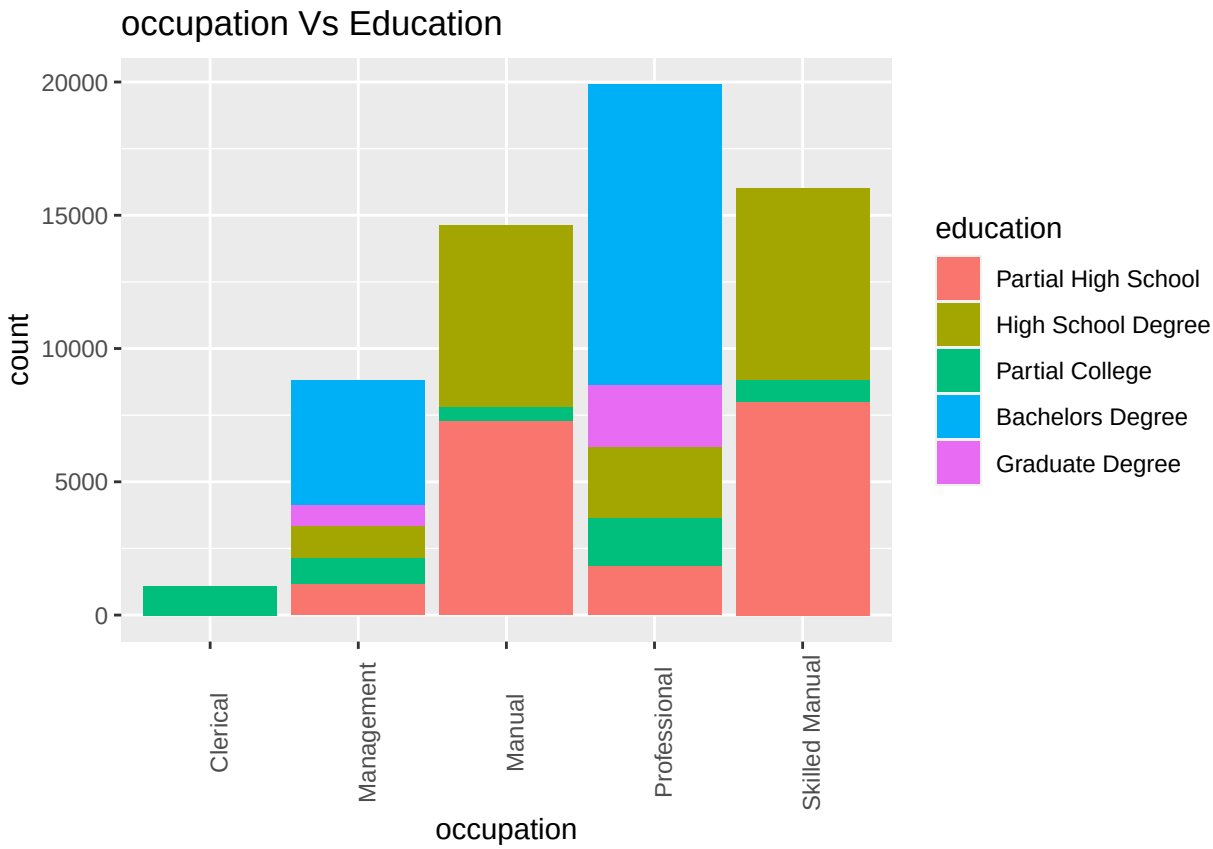
Member card is the membership card that the customer owns.

```
card_income <- CFM_data %>% dplyr::select(member_card,avg..yearly_income) %>% group_by(member_card,avg.
#card_income %>% count()
ggplot(card_income, aes(x=avg..yearly_income,fill=member_card)) + geom_bar()+ theme(axis.text.x = eleme
```



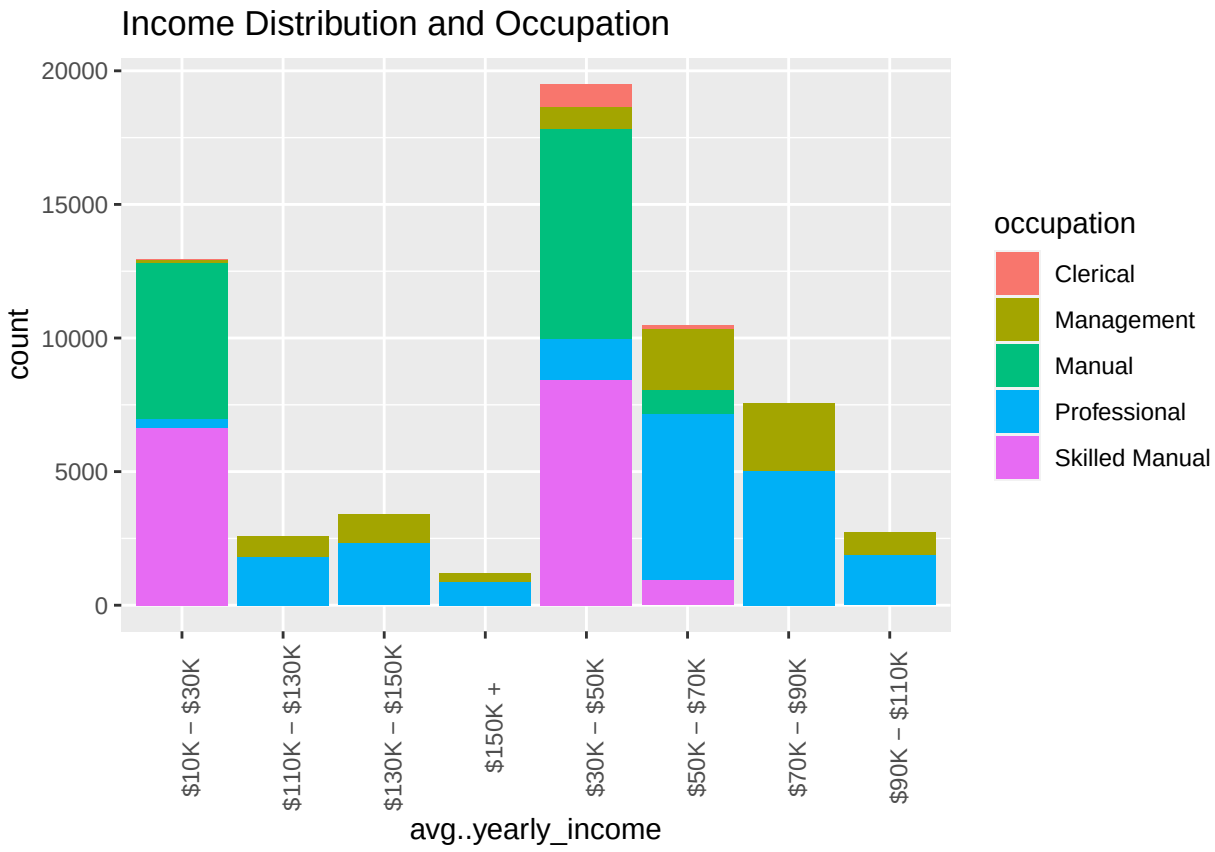
Visualize the education with occupation

```
ed_oc <- CFM_data %>% dplyr::select(occupation,education) %>% group_by(occupation,education)
#ed_oc %>% count()
ggplot(ed_oc, aes(x=occupation,fill=education)) + geom_bar()+ theme(axis.text.x = element_text(angle =
```



Visualize the average income with occupation

```
Income_ocup <- CFM_data %>% dplyr::select(avg..yearly_income,occupation) %>% group_by(occupation,avg..y
#Income_ocup %>% count()
ggplot(Income_ocup,aes(x=avg..yearly_income,fill=occupation)) + geom_bar()+ theme(axis.text.x = element_
```

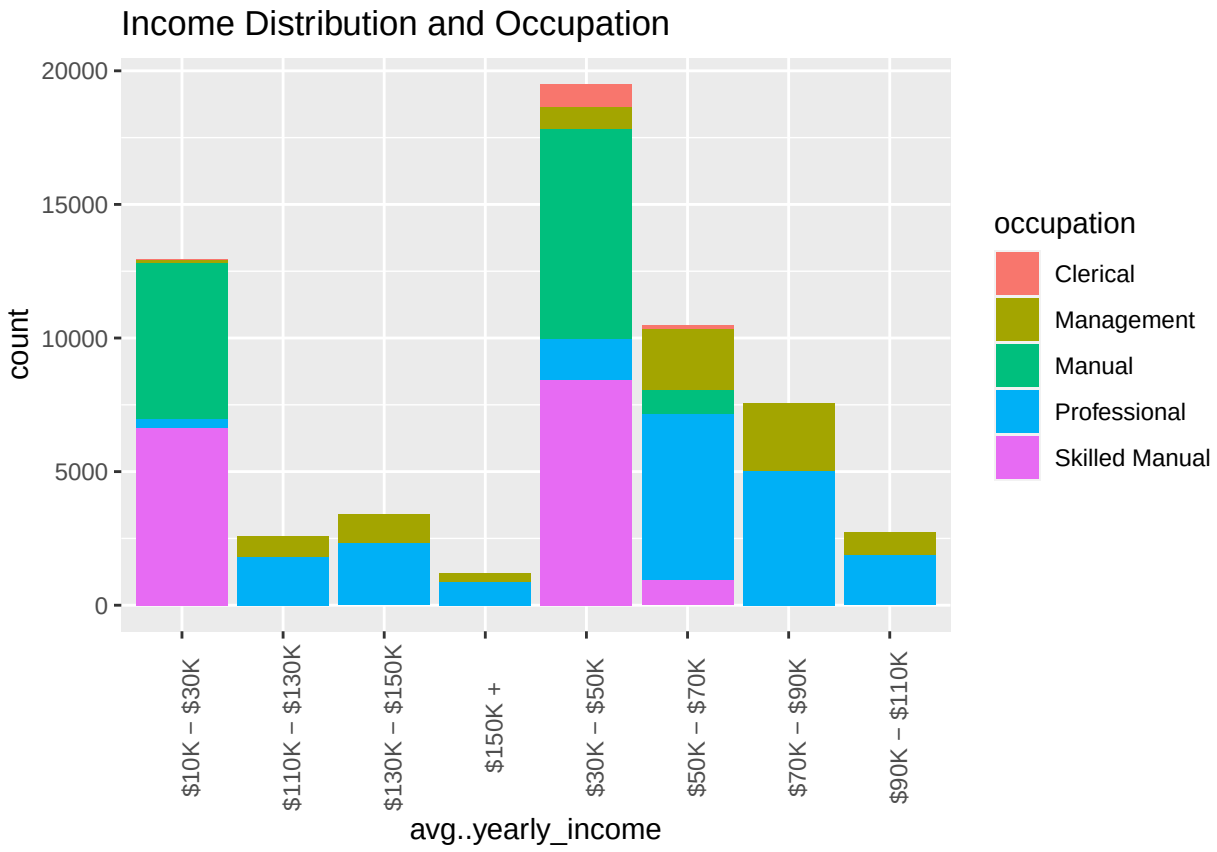


Observation

1. Professional employee average earns around $50k +.
2. Skilled manual employee and manual employee earns $30k +.
3. Management employee earns $30k +.
4. Clerical employee earns $30k +.

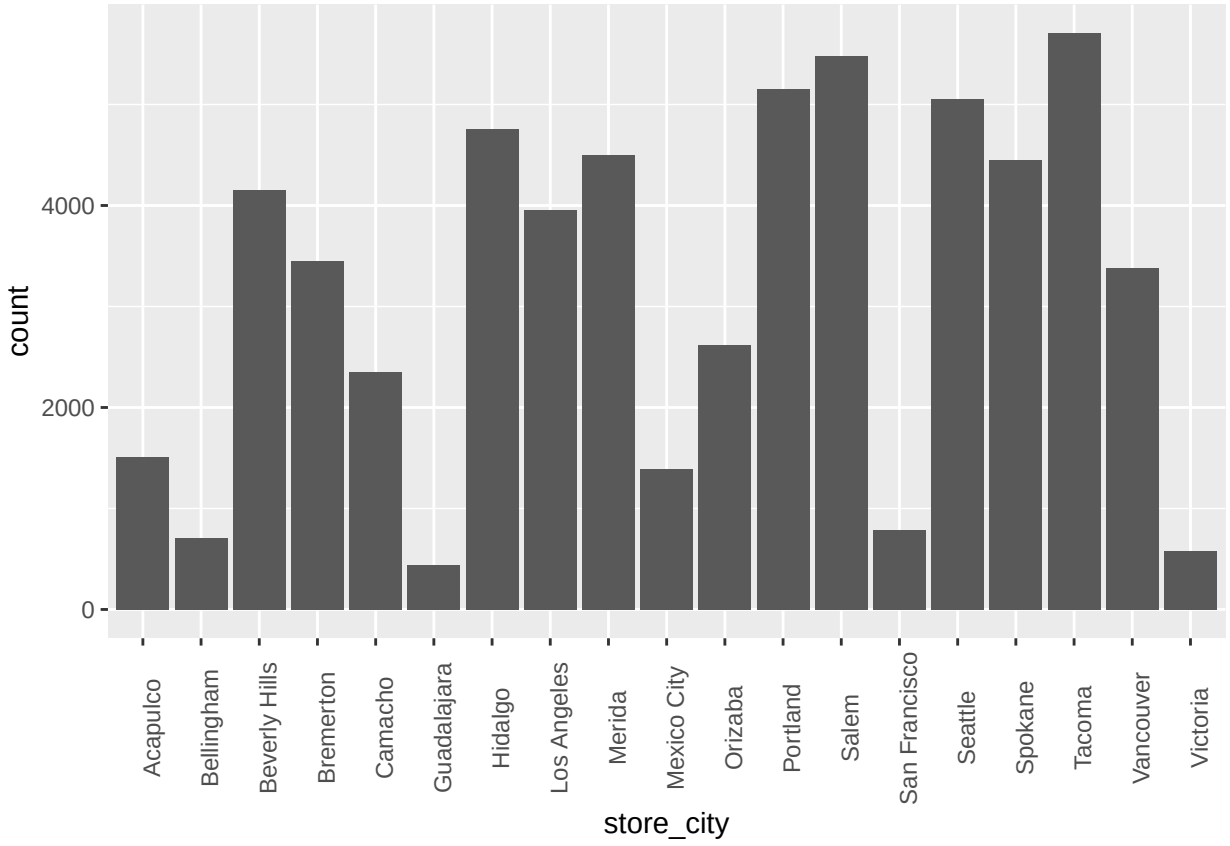
Visualize the average yearly income with occupation

```
Income_ocup <- CFM_data %>% dplyr::select(avg..yearly_income,occupation) %>% group_by(occupation,avg..y
#Income_ocup %>% count()
ggplot(Income_ocup,aes(x=avg..yearly_income,fill=occupation)) + geom_bar()+ theme(axis.text.x = element_
```



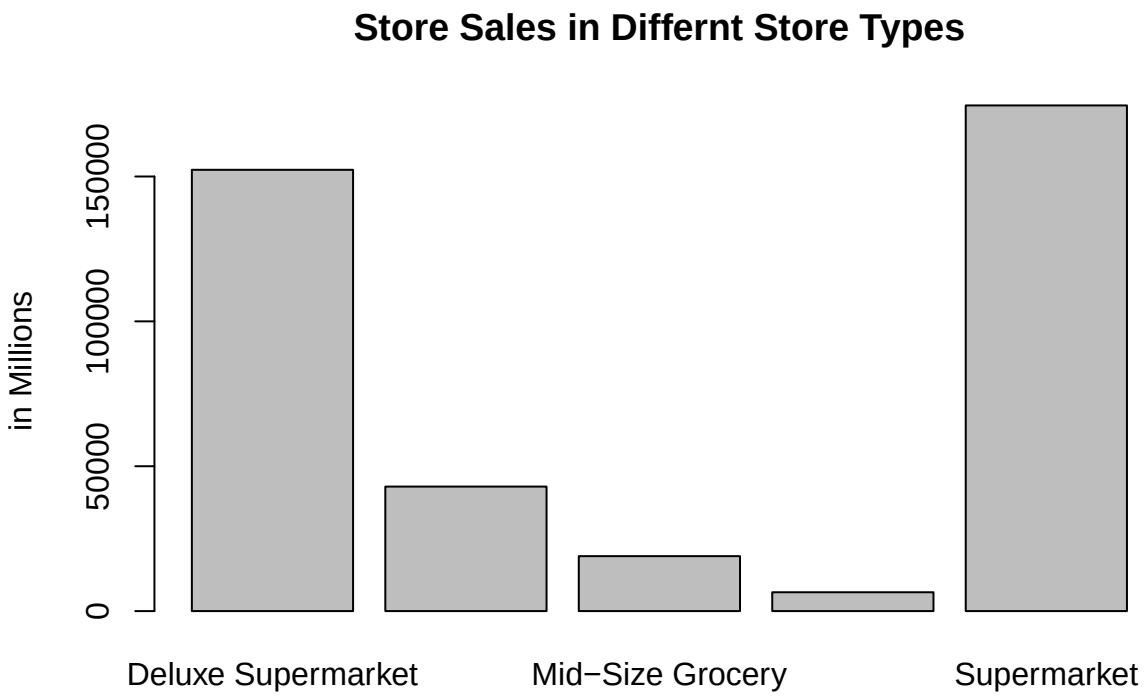
Visualise store sales with the state the store is located.

```
City_income <- CFM_data %>% dplyr::select(store_city,store_sales.in.millions.) %>% group_by(store_city,s
ggplot(City_income, aes(x=store_city),y=store_sales.in.millions.) + geom_bar()+ theme(axis.text.x = ele
```



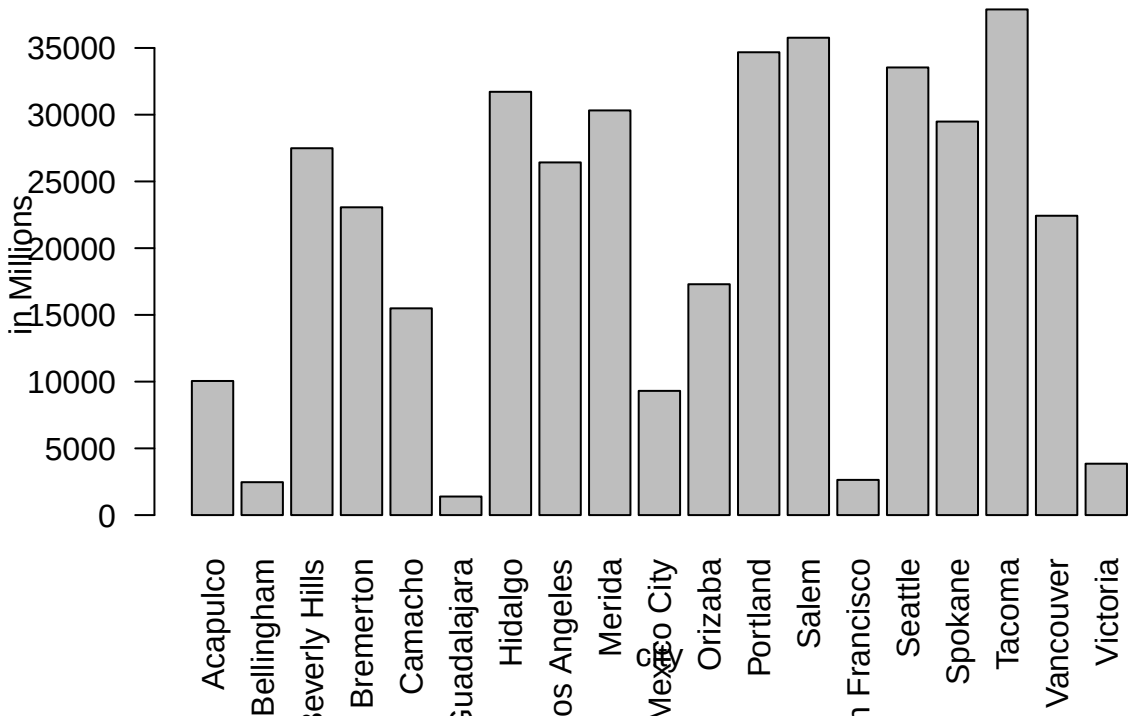
Visualize store type and the state it is located

```
sales_type_rev <- tapply(CFM_data$store_sales.in.millions., CFM_data$store_type, FUN=sum)
#sales_type_rev
p <- barplot(tapply(CFM_data$store_sales.in.millions., CFM_data$store_type, FUN=sum),ylab="in Millions"
```



Visualize store sales in each state

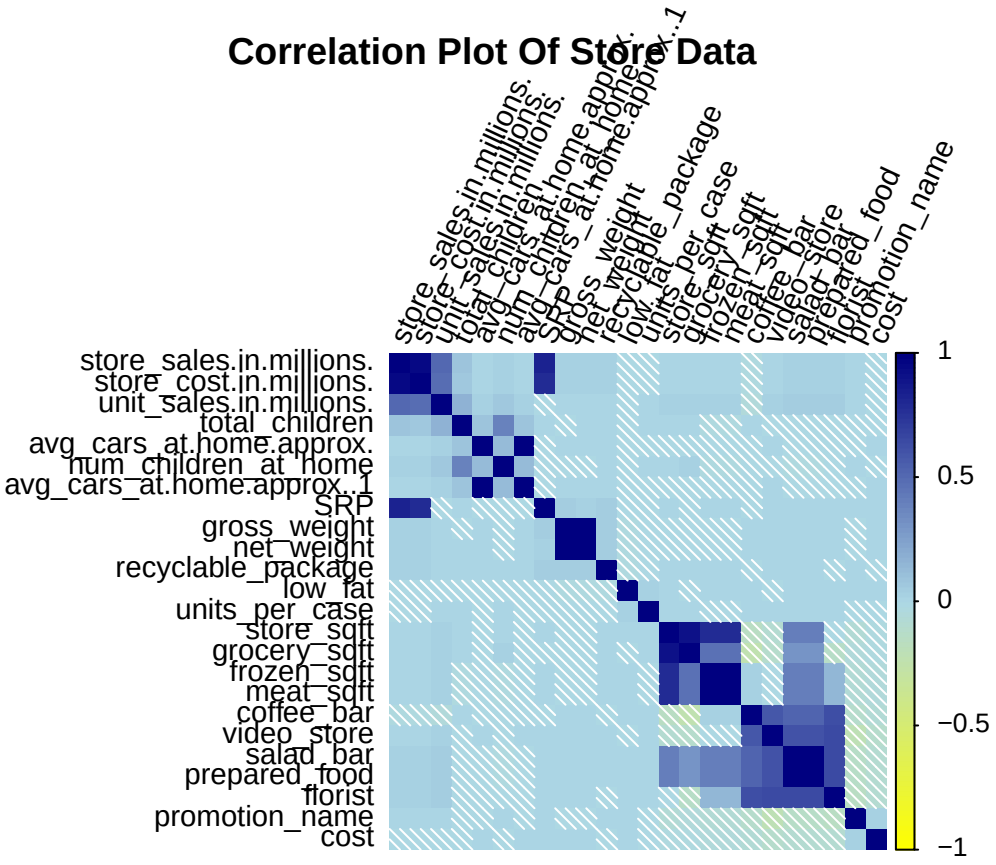
```
sales_city <- CFM_data %>% dplyr::select(store_city,store_sales.in.millions.)
#sales_city
sales_city_rev <- tapply(CFM_data$store_sales.in.millions.,CFM_data$store_city,FUN=sum)
#sales_city_rev
barplot(sales_city_rev,las=2,ylab="in Millions",xlab="city",main="Revenue state-wise")
```




```
library(CatEncoders)
transformed_data = CFM_data
#define original categorical labels and convert all the data for modeling
label = LabelEncoder.fit(CFM_data$food_category)
transformed_data$food_category = transform(label, CFM_data$food_category)

label = LabelEncoder.fit(CFM_data$food_department)
transformed_data$food_department = transform(label, CFM_data$food_department)

label = LabelEncoder.fit(CFM_data$food_family)
transformed_data$food_family = transform(label, CFM_data$food_family)

label = LabelEncoder.fit(CFM_data$promotion_name)
transformed_data$promotion_name = transform(label, CFM_data$promotion_name)

label = LabelEncoder.fit(CFM_data$marital_status)
transformed_data$marital_status = transform(label, CFM_data$marital_status)

label = LabelEncoder.fit(CFM_data$gender)
transformed_data$gender = transform(label, CFM_data$gender)

label = LabelEncoder.fit(CFM_data$education)
transformed_data$education = transform(label, CFM_data$education)

label = LabelEncoder.fit(CFM_data$member_card)
transformed_data$member_card = transform(label, CFM_data$member_card)
```

```
label = LabelEncoder.fit(CFM_data$occupation)
transformed_data$occupation = transform(label, CFM_data$occupation)

label = LabelEncoder.fit(CFM_data$avg..yearly_income)
transformed_data$avg..yearly_income = transform(label, CFM_data$avg..yearly_income)

label = LabelEncoder.fit(CFM_data$sales_country)
transformed_data$sales_country = transform(label, CFM_data$sales_country)

label = LabelEncoder.fit(CFM_data$brand_name)
transformed_data$brand_name = transform(label, CFM_data$brand_name)

label = LabelEncoder.fit(CFM_data$houseowner)
transformed_data$houseowner = transform(label, CFM_data$houseowner)

label = LabelEncoder.fit(CFM_data$store_type)
transformed_data$store_type = transform(label, CFM_data$store_type)

label = LabelEncoder.fit(CFM_data$store_city)
transformed_data$store_city = transform(label, CFM_data$store_city)

label = LabelEncoder.fit(CFM_data$store_state)
transformed_data$store_state = transform(label, CFM_data$store_state)

label = LabelEncoder.fit(CFM_data$media_type)
transformed_data$media_type = transform(label, CFM_data$media_type)

#(transformed_data)
```

```
#Correlation Matrix
store_data <- dplyr::select(transformed_data,c('store_sales.in.millions.','store_cost.in.millions.','uni
#store_data
cor_data <- cor(store_data)
#head(cor_data)
corrplot(cor_data,method="shade",tl.col = "black",title = "\n\n Correlation Plot Of Store Data",tl.srt=
```



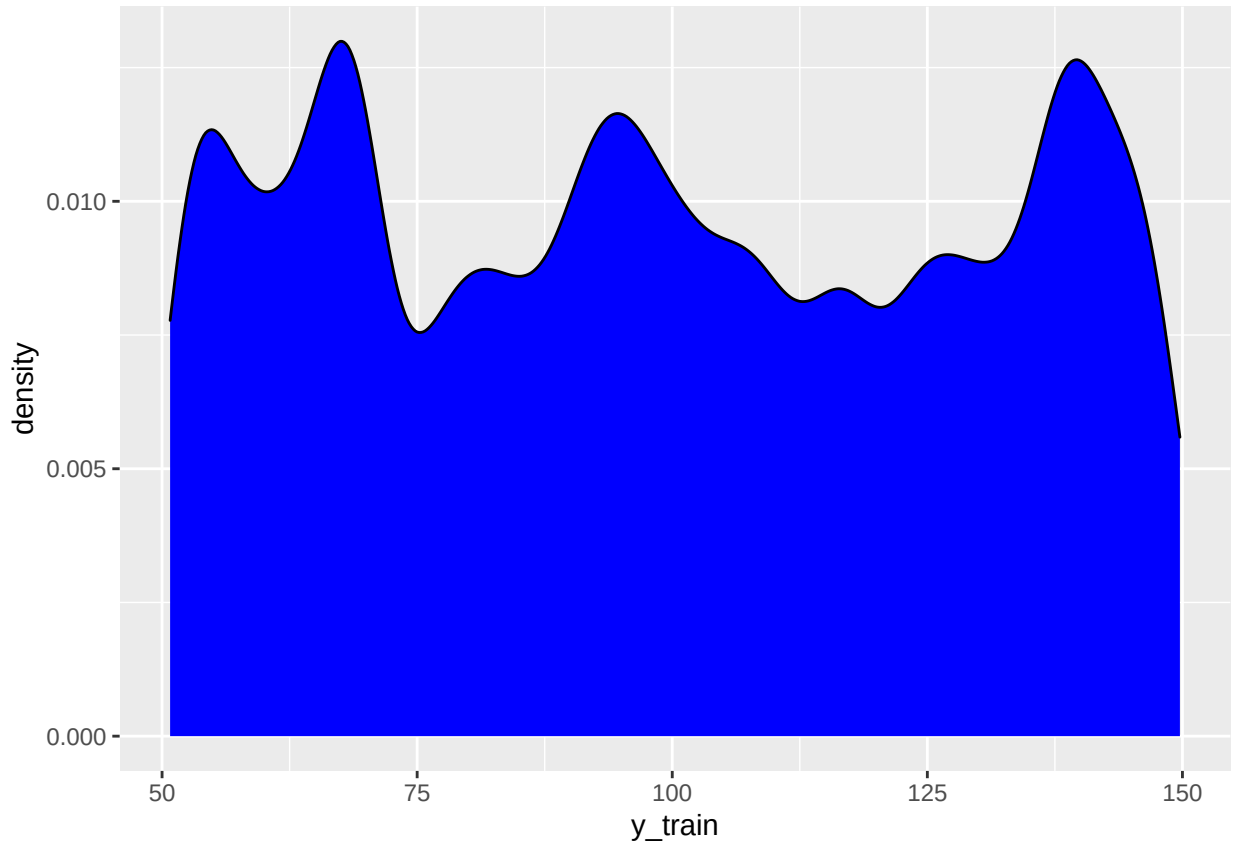

```
#split the data


# Split data into partitions
set.seed(3451)
inds <- partition(transformed_data$food_category, p = c(train = 0.7, valid = 0.2, test = 0.1))
#str(inds)
#> List of 3
#>  $ train: int [1:81] 2 3 6 7 8 10 11 18 19 20 ...
#>  $ valid: int [1:34] 1 12 14 15 27 34 36 38 42 48 ...
#>  $ test : int [1:35] 4 5 9 13 16 17 25 39 41 45 ...

train <- transformed_data[inds$train, ]
X_train=dplyr::select(train,-cost)
y_train=train$cost

valid <- transformed_data[inds$valid, ]
X_valid=dplyr::select(valid,-cost)
y_valid=valid$cost


test <- transformed_data[inds$test, ]
X_test=dplyr::select(test,-cost)
y_test=test$cost

#print(X_train)
```

```
#print(y_train)
#dim(X_train)
#length(y_train)
#dim(X_test)
#length(y_test)
#dim(X_valid)
#length(y_valid)
```

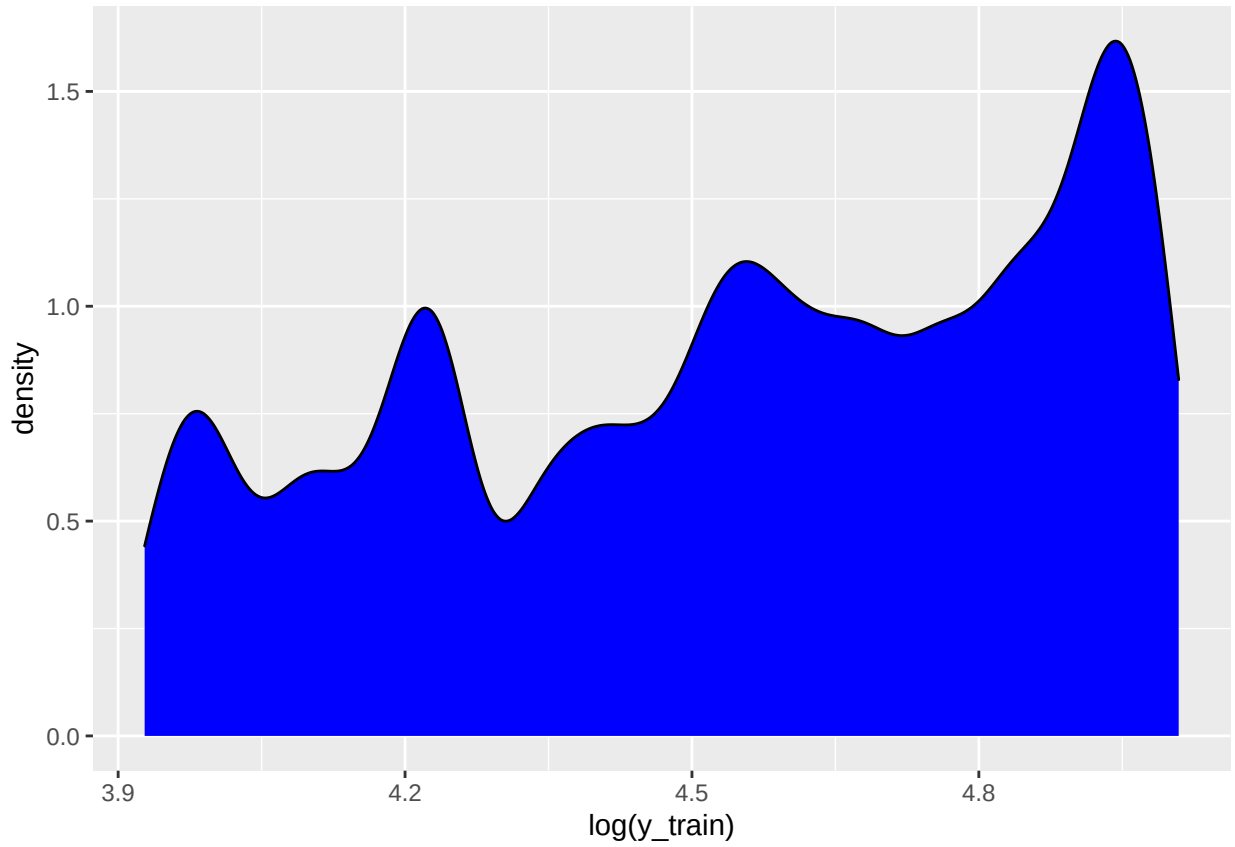
```
##Explore the data.
library(MASS)
```

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

## The following object is masked from 'package:plotly':
##
##     select

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

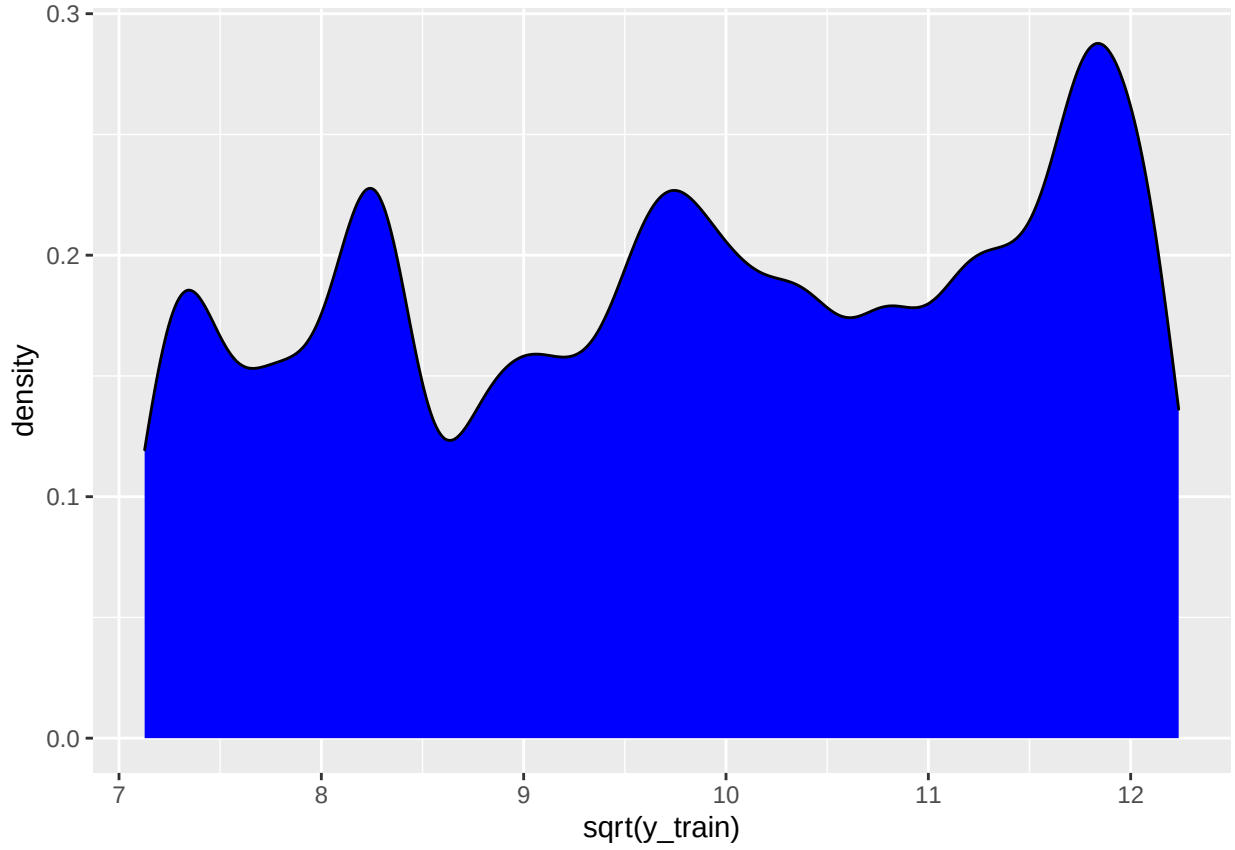
```
library(ggplot2)
ggplot(X_train, aes(y_train)) + geom_density(fill="blue")
```



```
ggplot(X_train, aes(log(y_train))) + geom_density(fill="blue")
```




```
ggplot(X_train, aes(sqrt(y_train))) + geom_density(fill="blue")
```

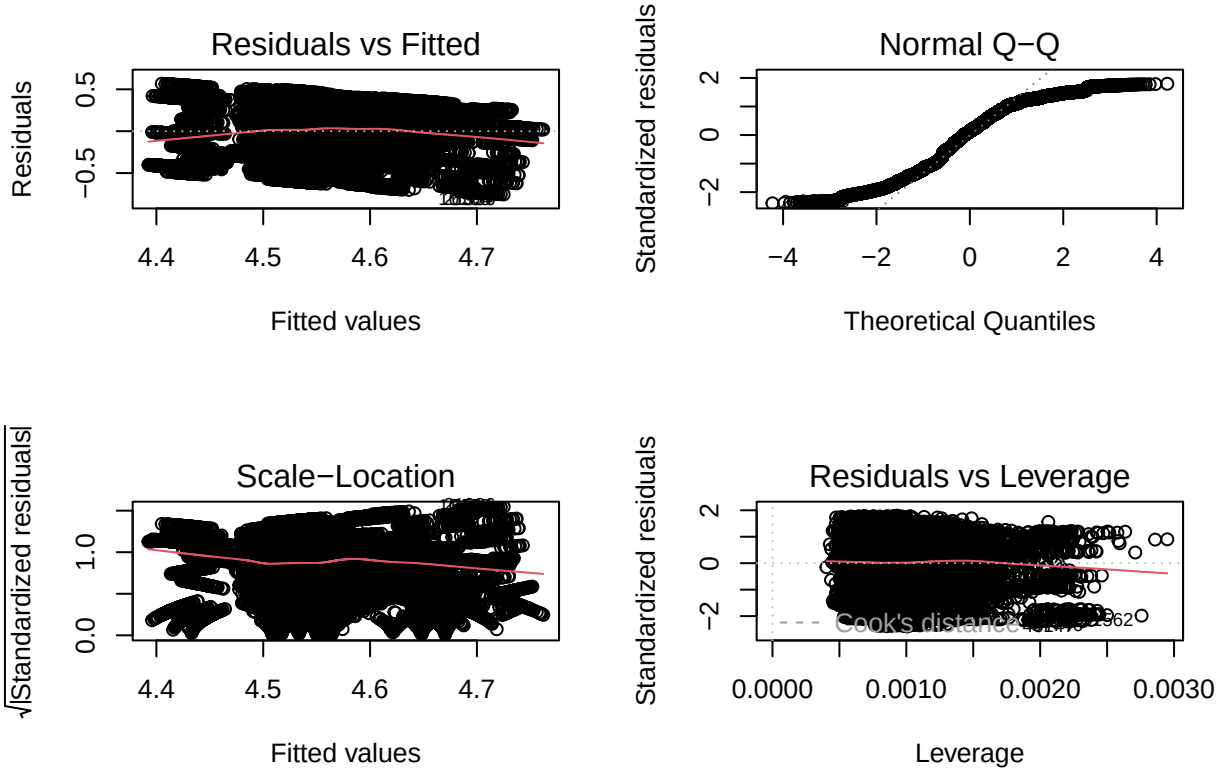



```
model1 = lm(log(y_train)~., data=X_train)
summary(model1)
```

```
##
## Call:
## lm(formula = log(y_train) ~ ., data = X_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.75533 -0.28494  0.04028  0.27737  0.56736
##
## Coefficients: (2 not defined because of singularities)
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                5.179e+00  4.245e-02 122.011  < 2e-16 ***
## food_category             -3.183e-05  1.304e-04  -0.244 0.807098
## food_department           -1.940e-04  3.280e-04  -0.591 0.554270
## food_family                1.741e-03  3.262e-03   0.534 0.593568
## store_sales.in.millions.   5.652e-04  2.459e-03   0.230 0.818233
## store_cost.in.millions.    2.504e-03  3.569e-03   0.702 0.482967
## unit_sales.in.millions.   -3.512e-03  4.639e-03  -0.757 0.449026
## promotion_name            -1.500e-05  1.161e-04  -0.129 0.897201
## sales_country             -5.376e-02  4.038e-03 -13.312  < 2e-16 ***
## marital_status             1.388e-03  4.270e-03   0.325 0.745206
## gender                    -7.261e-03  3.094e-03  -2.347 0.018949 *
## total_children            -1.764e-03  1.194e-03  -1.478 0.139446
```

```
## education                  -2.097e-04  1.239e-03  -0.169 0.865611
## member_card                -4.701e-04  1.711e-03  -0.275 0.783542
## occupation                  9.089e-04  1.463e-03   0.621 0.534487
## houseowner                  4.211e-03  3.295e-03   1.278 0.201140
## avg_cars_at.home.approx.    4.388e-03  1.624e-03   2.701 0.006910 **
## avg..yearly_income         -8.418e-04  1.002e-03  -0.840 0.401018
## num_children_at_home       -3.075e-04  1.780e-03  -0.173 0.862856
## avg_cars_at.home.approx..1         NA         NA      NA       NA
## brand_name                  5.621e-06  4.990e-05   0.113 0.910307
## SRP                        -3.937e-03  6.401e-03  -0.615 0.538471
## gross_weight                1.457e-03  2.241e-03   0.650 0.515472
## net_weight                 -1.323e-03  2.212e-03  -0.598 0.549803
## recyclable_package         -3.379e-03  3.109e-03  -1.087 0.277225
## low_fat                     7.755e-03  3.286e-03   2.360 0.018275 *
## units_per_case              3.332e-05  1.503e-04   0.222 0.824552
## store_type                 -3.932e-02  3.634e-03 -10.821  < 2e-16 ***
## store_city                 -4.803e-03  5.077e-04  -9.460  < 2e-16 ***
## store_state                -4.409e-03  7.551e-04  -5.839 5.28e-09 ***
## store_sqft                  1.637e-02  4.282e-03   3.824 0.000132 ***
## grocery_sqft               -1.638e-02  4.282e-03  -3.825 0.000131 ***
## frozen_sqft                 9.744e-03  7.383e-03   1.320 0.186915
## meat_sqft                  -5.556e-02  9.858e-03  -5.636 1.75e-08 ***
## coffee_bar                 -8.572e-03  7.912e-03  -1.083 0.278657
## video_store                -1.438e-01  6.472e-03 -22.221  < 2e-16 ***
## salad_bar                  -1.676e-02  6.072e-03  -2.760 0.005777 **
## prepared_food                      NA         NA      NA       NA
## florist                    -1.017e-01  8.977e-03 -11.328  < 2e-16 ***
## media_type                 -4.301e-03  4.531e-04  -9.492  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3161 on 42256 degrees of freedom
## Multiple R-squared:  0.03513,    Adjusted R-squared:  0.03428
## F-statistic: 41.58 on 37 and 42256 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(model1)
```




```
#Random forest
#library(randomForest)
#rf.fit <- randomForest(y_train ~ ., data=X_train, ntree=1000,
#                       keep.forest=FALSE, importance=TRUE)
#rf.fit
#pred <- rf.predict( dtest)
```

```
#XGBRegressor
dtrain <- xgb.DMatrix(data = as.matrix(X_train), label = y_train)
dtest <- xgb.DMatrix(data = as.matrix(X_test), label = y_test)
# Instantiate the model
model <- xgboost(data = dtrain, # the data
                 nround = 350, # max number of boosting iterations)
                    max.depth = 5
                 )
```

```
## [1]  train-rmse:74.668993
## [2]  train-rmse:55.054931
## [3]  train-rmse:41.788303
## [4]  train-rmse:33.253145
## [5]  train-rmse:27.768861
## [6]  train-rmse:24.111993
## [7]  train-rmse:21.634981
## [8]  train-rmse:19.551783
## [9]  train-rmse:18.501236
```

```
## [10] train-rmse:17.529593
## [11] train-rmse:16.913653
## [12] train-rmse:16.280023
## [13] train-rmse:15.759865
## [14] train-rmse:15.358209
## [15] train-rmse:15.003864
## [16] train-rmse:14.379773
## [17] train-rmse:13.883081
## [18] train-rmse:13.689130
## [19] train-rmse:13.275439
## [20] train-rmse:12.818343
## [21] train-rmse:12.217828
## [22] train-rmse:11.963897
## [23] train-rmse:11.749819
## [24] train-rmse:11.305673
## [25] train-rmse:10.944367
## [26] train-rmse:10.685916
## [27] train-rmse:10.425577
## [28] train-rmse:9.867064
## [29] train-rmse:9.201626
## [30] train-rmse:8.756177
## [31] train-rmse:8.577275
## [32] train-rmse:8.081662
## [33] train-rmse:8.027418
## [34] train-rmse:7.698742
## [35] train-rmse:7.534771
## [36] train-rmse:7.244088
## [37] train-rmse:7.033832
## [38] train-rmse:6.764427
## [39] train-rmse:6.590716
## [40] train-rmse:6.354999
## [41] train-rmse:6.119393
## [42] train-rmse:5.981929
## [43] train-rmse:5.741202
## [44] train-rmse:5.623326
## [45] train-rmse:5.486665
## [46] train-rmse:5.440589
## [47] train-rmse:5.268226
## [48] train-rmse:5.074642
## [49] train-rmse:4.910233
## [50] train-rmse:4.695644
## [51] train-rmse:4.566101
## [52] train-rmse:4.417273
## [53] train-rmse:4.327428
## [54] train-rmse:4.254701
## [55] train-rmse:4.183321
## [56] train-rmse:4.012519
## [57] train-rmse:3.929526
## [58] train-rmse:3.794505
## [59] train-rmse:3.737595
## [60] train-rmse:3.682647
## [61] train-rmse:3.643896
## [62] train-rmse:3.584644
## [63] train-rmse:3.433692
```

```
## [64] train-rmse:3.378350
## [65] train-rmse:3.265028
## [66] train-rmse:3.188539
## [67] train-rmse:3.036526
## [68] train-rmse:3.002756
## [69] train-rmse:2.961227
## [70] train-rmse:2.931690
## [71] train-rmse:2.909127
## [72] train-rmse:2.839373
## [73] train-rmse:2.759056
## [74] train-rmse:2.693929
## [75] train-rmse:2.664173
## [76] train-rmse:2.628090
## [77] train-rmse:2.567603
## [78] train-rmse:2.542182
## [79] train-rmse:2.517285
## [80] train-rmse:2.418305
## [81] train-rmse:2.410779
## [82] train-rmse:2.374005
## [83] train-rmse:2.350904
## [84] train-rmse:2.269375
## [85] train-rmse:2.234036
## [86] train-rmse:2.148239
## [87] train-rmse:2.118119
## [88] train-rmse:2.085225
## [89] train-rmse:2.075290
## [90] train-rmse:2.065255
## [91] train-rmse:2.054090
## [92] train-rmse:2.027880
## [93] train-rmse:1.982380
## [94] train-rmse:1.931072
## [95] train-rmse:1.916374
## [96] train-rmse:1.909266
## [97] train-rmse:1.874727
## [98] train-rmse:1.860486
## [99] train-rmse:1.850681
## [100]    train-rmse:1.838526
## [101]    train-rmse:1.828236
## [102]    train-rmse:1.817591
## [103]    train-rmse:1.812192
## [104]    train-rmse:1.806388
## [105]    train-rmse:1.799041
## [106]    train-rmse:1.783649
## [107]    train-rmse:1.762765
## [108]    train-rmse:1.761982
## [109]    train-rmse:1.734699
## [110]    train-rmse:1.721522
## [111]    train-rmse:1.685173
## [112]    train-rmse:1.679880
## [113]    train-rmse:1.665578
## [114]    train-rmse:1.662632
## [115]    train-rmse:1.653254
## [116]    train-rmse:1.634807
## [117]    train-rmse:1.630730
```

```
## [118]    train-rmse:1.629233
## [119]    train-rmse:1.619429
## [120]    train-rmse:1.601703
## [121]    train-rmse:1.595704
## [122]    train-rmse:1.571707
## [123]    train-rmse:1.553266
## [124]    train-rmse:1.542786
## [125]    train-rmse:1.535163
## [126]    train-rmse:1.528575
## [127]    train-rmse:1.521553
## [128]    train-rmse:1.507724
## [129]    train-rmse:1.494909
## [130]    train-rmse:1.486399
## [131]    train-rmse:1.477581
## [132]    train-rmse:1.462264
## [133]    train-rmse:1.452365
## [134]    train-rmse:1.446930
## [135]    train-rmse:1.446098
## [136]    train-rmse:1.438923
## [137]    train-rmse:1.433391
## [138]    train-rmse:1.429680
## [139]    train-rmse:1.425271
## [140]    train-rmse:1.419690
## [141]    train-rmse:1.419029
## [142]    train-rmse:1.417571
## [143]    train-rmse:1.412911
## [144]    train-rmse:1.397378
## [145]    train-rmse:1.380969
## [146]    train-rmse:1.366775
## [147]    train-rmse:1.356106
## [148]    train-rmse:1.346018
## [149]    train-rmse:1.329356
## [150]    train-rmse:1.326858
## [151]    train-rmse:1.320560
## [152]    train-rmse:1.314428
## [153]    train-rmse:1.309682
## [154]    train-rmse:1.298239
## [155]    train-rmse:1.295513
## [156]    train-rmse:1.294009
## [157]    train-rmse:1.277916
## [158]    train-rmse:1.262285
## [159]    train-rmse:1.253351
## [160]    train-rmse:1.250518
## [161]    train-rmse:1.244747
## [162]    train-rmse:1.236106
## [163]    train-rmse:1.218414
## [164]    train-rmse:1.216883
## [165]    train-rmse:1.211795
## [166]    train-rmse:1.197379
## [167]    train-rmse:1.196360
## [168]    train-rmse:1.188902
## [169]    train-rmse:1.182875
## [170]    train-rmse:1.171528
## [171]    train-rmse:1.167588
```

```
## [172]    train-rmse:1.164667
## [173]    train-rmse:1.158325
## [174]    train-rmse:1.156049
## [175]    train-rmse:1.149087
## [176]    train-rmse:1.146840
## [177]    train-rmse:1.135938
## [178]    train-rmse:1.124346
## [179]    train-rmse:1.120447
## [180]    train-rmse:1.119957
## [181]    train-rmse:1.114058
## [182]    train-rmse:1.108677
## [183]    train-rmse:1.105093
## [184]    train-rmse:1.103869
## [185]    train-rmse:1.096584
## [186]    train-rmse:1.089942
## [187]    train-rmse:1.083054
## [188]    train-rmse:1.078324
## [189]    train-rmse:1.071301
## [190]    train-rmse:1.061869
## [191]    train-rmse:1.052278
## [192]    train-rmse:1.045126
## [193]    train-rmse:1.037886
## [194]    train-rmse:1.033132
## [195]    train-rmse:1.032476
## [196]    train-rmse:1.025971
## [197]    train-rmse:1.019074
## [198]    train-rmse:1.014197
## [199]    train-rmse:1.013742
## [200]    train-rmse:1.013150
## [201]    train-rmse:1.008904
## [202]    train-rmse:1.003661
## [203]    train-rmse:1.001596
## [204]    train-rmse:0.995645
## [205]    train-rmse:0.984108
## [206]    train-rmse:0.982825
## [207]    train-rmse:0.979843
## [208]    train-rmse:0.978936
## [209]    train-rmse:0.972532
## [210]    train-rmse:0.968397
## [211]    train-rmse:0.961028
## [212]    train-rmse:0.957877
## [213]    train-rmse:0.956760
## [214]    train-rmse:0.951585
## [215]    train-rmse:0.947912
## [216]    train-rmse:0.945849
## [217]    train-rmse:0.939818
## [218]    train-rmse:0.936623
## [219]    train-rmse:0.935533
## [220]    train-rmse:0.930975
## [221]    train-rmse:0.919302
## [222]    train-rmse:0.916431
## [223]    train-rmse:0.912701
## [224]    train-rmse:0.910544
## [225]    train-rmse:0.908624
```

```
## [226]    train-rmse:0.901811
## [227]    train-rmse:0.894753
## [228]    train-rmse:0.893654
## [229]    train-rmse:0.890322
## [230]    train-rmse:0.885007
## [231]    train-rmse:0.884308
## [232]    train-rmse:0.883309
## [233]    train-rmse:0.873894
## [234]    train-rmse:0.868784
## [235]    train-rmse:0.864768
## [236]    train-rmse:0.857506
## [237]    train-rmse:0.853856
## [238]    train-rmse:0.848180
## [239]    train-rmse:0.838622
## [240]    train-rmse:0.836877
## [241]    train-rmse:0.833431
## [242]    train-rmse:0.827696
## [243]    train-rmse:0.827031
## [244]    train-rmse:0.826968
## [245]    train-rmse:0.821369
## [246]    train-rmse:0.816443
## [247]    train-rmse:0.814243
## [248]    train-rmse:0.813013
## [249]    train-rmse:0.810607
## [250]    train-rmse:0.810283
## [251]    train-rmse:0.809787
## [252]    train-rmse:0.807787
## [253]    train-rmse:0.798395
## [254]    train-rmse:0.793326
## [255]    train-rmse:0.788886
## [256]    train-rmse:0.787454
## [257]    train-rmse:0.786915
## [258]    train-rmse:0.785949
## [259]    train-rmse:0.783931
## [260]    train-rmse:0.780170
## [261]    train-rmse:0.780024
## [262]    train-rmse:0.779904
## [263]    train-rmse:0.775197
## [264]    train-rmse:0.771613
## [265]    train-rmse:0.763402
## [266]    train-rmse:0.756498
## [267]    train-rmse:0.749948
## [268]    train-rmse:0.748823
## [269]    train-rmse:0.748510
## [270]    train-rmse:0.746312
## [271]    train-rmse:0.744482
## [272]    train-rmse:0.741754
## [273]    train-rmse:0.741271
## [274]    train-rmse:0.735793
## [275]    train-rmse:0.735143
## [276]    train-rmse:0.733524
## [277]    train-rmse:0.732201
## [278]    train-rmse:0.730735
## [279]    train-rmse:0.725443
```

```
## [280]    train-rmse:0.722874
## [281]    train-rmse:0.718040
## [282]    train-rmse:0.712750
## [283]    train-rmse:0.710874
## [284]    train-rmse:0.710073
## [285]    train-rmse:0.710037
## [286]    train-rmse:0.703614
## [287]    train-rmse:0.695620
## [288]    train-rmse:0.689807
## [289]    train-rmse:0.682741
## [290]    train-rmse:0.681495
## [291]    train-rmse:0.679789
## [292]    train-rmse:0.678178
## [293]    train-rmse:0.675140
## [294]    train-rmse:0.665193
## [295]    train-rmse:0.661266
## [296]    train-rmse:0.659971
## [297]    train-rmse:0.659635
## [298]    train-rmse:0.657059
## [299]    train-rmse:0.654786
## [300]    train-rmse:0.654139
## [301]    train-rmse:0.652361
## [302]    train-rmse:0.648351
## [303]    train-rmse:0.647139
## [304]    train-rmse:0.646704
## [305]    train-rmse:0.645600
## [306]    train-rmse:0.645003
## [307]    train-rmse:0.644398
## [308]    train-rmse:0.642024
## [309]    train-rmse:0.641912
## [310]    train-rmse:0.639728
## [311]    train-rmse:0.637309
## [312]    train-rmse:0.634386
## [313]    train-rmse:0.633676
## [314]    train-rmse:0.630316
## [315]    train-rmse:0.628541
## [316]    train-rmse:0.627638
## [317]    train-rmse:0.626817
## [318]    train-rmse:0.626226
## [319]    train-rmse:0.625462
## [320]    train-rmse:0.623960
## [321]    train-rmse:0.622909
## [322]    train-rmse:0.617240
## [323]    train-rmse:0.615793
## [324]    train-rmse:0.613926
## [325]    train-rmse:0.608581
## [326]    train-rmse:0.602948
## [327]    train-rmse:0.602401
## [328]    train-rmse:0.600431
## [329]    train-rmse:0.598891
## [330]    train-rmse:0.598342
## [331]    train-rmse:0.597971
## [332]    train-rmse:0.597864
## [333]    train-rmse:0.594365
```

```
## [334]    train-rmse:0.591659
## [335]    train-rmse:0.589770
## [336]    train-rmse:0.589482
## [337]    train-rmse:0.585945
## [338]    train-rmse:0.583696
## [339]    train-rmse:0.581464
## [340]    train-rmse:0.580422
## [341]    train-rmse:0.578026
## [342]    train-rmse:0.576476
## [343]    train-rmse:0.575178
## [344]    train-rmse:0.574925
## [345]    train-rmse:0.574255
## [346]    train-rmse:0.573210
## [347]    train-rmse:0.570667
## [348]    train-rmse:0.569007
## [349]    train-rmse:0.566593
## [350]    train-rmse:0.566091
```

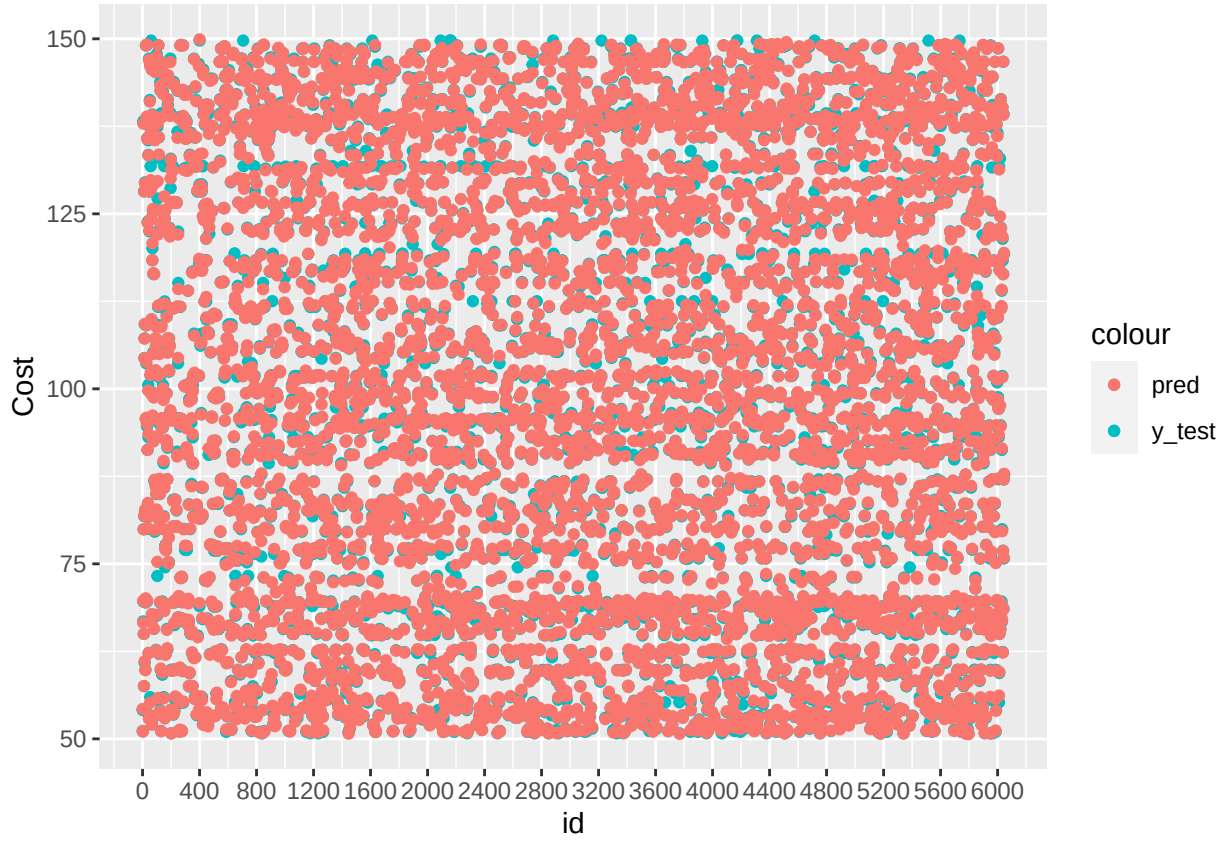
```
# Fit the model to the data
pred <- predict(model, dtest)
mse = mean((y_test - pred)^2)
mae = caret::MAE(y_test, pred)
rmse = caret::RMSE(y_test, pred)
postResample(y_test,pred)
```

```
##      RMSE  Rsquared       MAE
## 1.3369519 0.9980331 0.3387503
```

```
cat("MSE: ", mse, "\nMAE: ", mae, "\nRMSE: ", rmse)
```

```
## MSE:  1.78744
## MAE:  0.3387503
## RMSE:  1.336952
```

```
df = data.frame(y_test, pred)
df$id = 1:nrow(df)
ggplot() + geom_point(data = df, aes(x = id, y = y_test, color = 'y_test')) +
  geom_point(data = df, aes(x=id, y = pred, color = 'pred',)) +
  theme(plot.title = element_text(hjust = 0.5)) + scale_x_continuous(breaks = seq(0, 6000, by = 400))+
  ylab('Cost')
```




```
importance_matrix = xgb.importance(colnames(dtrain), model = model)
importance_matrix
```

```
##                     Feature         Gain        Cover   Frequency
##  1:          promotion_name 4.112074e-01 0.2955619192 0.275249152
##  2:              media_type 2.256189e-01 0.1300243777 0.115380664
##  3:              store_city 7.719638e-02 0.0684008216 0.091236001
##  4:              store_sqft 5.425779e-02 0.0276490969 0.024761122
##  5:             frozen_sqft 5.252147e-02 0.0298025831 0.028459879
##  6:            grocery_sqft 4.835854e-02 0.0275165555 0.024555635
##  7:             store_state 4.081676e-02 0.0215408650 0.025377581
##  8:           sales_country 2.857899e-02 0.0143797648 0.020343162
##  9:              store_type 2.460080e-02 0.0067202961 0.022500771
## 10:             video_store 1.340757e-02 0.0022683176 0.004006987
## 11:               salad_bar 1.270019e-02 0.0054902598 0.002671324
## 12:              coffee_bar 5.108575e-03 0.0027020918 0.003698757
## 13:                 florist 3.930630e-03 0.0013378242 0.001541149
## 14:  unit_sales.in.millions. 6.010412e-04 0.0110269712 0.015308743
## 15:          total_children 1.127210e-04 0.0083176196 0.020343162
## 16: store_sales.in.millions. 1.064580e-04 0.0858086510 0.041097298
## 17:              occupation 9.629202e-05 0.0093606055 0.018082811
## 18:  store_cost.in.millions. 9.043126e-05 0.0744371048 0.042227474
## 19: avg_cars_at.home.approx. 8.037547e-05 0.0169027554 0.025788554
## 20:             member_card 8.012358e-05 0.0071547327 0.011712730
## 21:      avg..yearly_income 7.052257e-05 0.0133138370 0.022398027
```

```
## 22:     num_children_at_home 6.650226e-05 0.0081662191 0.015719716
## 23:                education 6.279449e-05 0.0048508193 0.011507243
## 24:              net_weight 4.510006e-05 0.0289964385 0.016952635
## 25:             food_family 3.522649e-05 0.0010387139 0.002568581
## 26:           food_category 3.358751e-05 0.0070182708 0.022192541
## 27:          marital_status 3.237951e-05 0.0037555820 0.005959108
## 28:            gross_weight 3.140634e-05 0.0250068407 0.019315730
## 29:          units_per_case 2.989694e-05 0.0094689612 0.011609987
## 30:              brand_name 2.828311e-05 0.0145573519 0.015411487
## 31:                     SRP 2.501177e-05 0.0269655706 0.019315730
## 32:         food_department 2.228214e-05 0.0044110291 0.009760608
## 33:              houseowner 1.741416e-05 0.0031948093 0.005239905
## 34:                  gender 1.364180e-05 0.0013790575 0.005137162
## 35:       recyclable_package 8.345011e-06 0.0003334489 0.001335662
## 36:                  low_fat 6.219092e-06 0.0011398368 0.001232919
##                      Feature         Gain        Cover   Frequency
```

```
xgb.plot.multi.trees(feature_names = names(CFM_data),
                     model = model)
```

```
postResample(y_test,pred)
```

```
##      RMSE  Rsquared       MAE
## 1.3369519 0.9980331 0.3387503
```

```
#LGBMRegressor

dtrain = lgb.Dataset(as.matrix(X_train), label = y_train)

dtest = lgb.Dataset.create.valid(dtrain,as.matrix(X_test), label = y_test)

#dvalid = lgb.Dataset.create.valid(dtrain,as.matrix(X_valid),label = y_valid)
set.seed(123)
valids = list(test = dtest)
model <- lgb.train(
  params = list(
    objective = "regression"
  , metric = "l2"
  , min_data = 1L
  , learning_rate = .3
  , nrounds = 10L
  ),
  valids = list(test = dtest),
  data = dtrain
)
```

```
## [LightGBM] [Warning] Unknown parameter: nrounds
## [LightGBM] [Warning] Unknown parameter: nrounds
## [LightGBM] [Warning] Unknown parameter: nrounds
## [LightGBM] [Warning] Unknown parameter: nrounds
## [LightGBM] [Warning] Auto-choosing col-wise multi-threading, the overhead of testing was 0.004388 se
## You can set `force_col_wise=true` to remove the overhead.
## [LightGBM] [Info] Total Bins 1702
```

```
## [LightGBM] [Info] Number of data points in the train set: 42294, number of used features: 39
## [LightGBM] [Warning] Unknown parameter: nrounds
## [LightGBM] [Info] Start training from score 99.260843
## [1] "[1]:  test's l2:710.263"
## [1] "[2]:  test's l2:584.339"
## [1] "[3]:  test's l2:482.015"
## [1] "[4]:  test's l2:419.735"
## [1] "[5]:  test's l2:351.859"
## [1] "[6]:  test's l2:292.996"
## [1] "[7]:  test's l2:251.669"
## [1] "[8]:  test's l2:220.562"
## [1] "[9]:  test's l2:185.811"
## [1] "[10]:  test's l2:161.336"
## [1] "[11]:  test's l2:136.547"
## [1] "[12]:  test's l2:118.939"
## [1] "[13]:  test's l2:107.119"
## [1] "[14]:  test's l2:96.8358"
## [1] "[15]:  test's l2:85.92"
## [1] "[16]:  test's l2:74.9486"
## [1] "[17]:  test's l2:66.8776"
## [1] "[18]:  test's l2:60.41"
## [1] "[19]:  test's l2:56.0922"
## [1] "[20]:  test's l2:49.1349"
## [1] "[21]:  test's l2:44.942"
## [1] "[22]:  test's l2:40.8247"
## [1] "[23]:  test's l2:35.6826"
## [1] "[24]:  test's l2:33.6455"
## [1] "[25]:  test's l2:30.4383"
## [1] "[26]:  test's l2:28.2424"
## [1] "[27]:  test's l2:25.8925"
## [1] "[28]:  test's l2:24.4908"
## [1] "[29]:  test's l2:22.1462"
## [1] "[30]:  test's l2:19.5922"
## [1] "[31]:  test's l2:18.2237"
## [1] "[32]:  test's l2:16.1163"
## [1] "[33]:  test's l2:14.6782"
## [1] "[34]:  test's l2:12.7222"
## [1] "[35]:  test's l2:11.7326"
## [1] "[36]:  test's l2:11.2003"
## [1] "[37]:  test's l2:10.3079"
## [1] "[38]:  test's l2:9.56079"
## [1] "[39]:  test's l2:8.87113"
## [1] "[40]:  test's l2:8.01551"
## [1] "[41]:  test's l2:7.78681"
## [1] "[42]:  test's l2:7.04141"
## [1] "[43]:  test's l2:6.35438"
## [1] "[44]:  test's l2:6.0828"
## [1] "[45]:  test's l2:5.67688"
## [1] "[46]:  test's l2:5.31381"
## [1] "[47]:  test's l2:4.96518"
## [1] "[48]:  test's l2:4.78566"
## [1] "[49]:  test's l2:4.74263"
## [1] "[50]:  test's l2:4.39161"
## [1] "[51]:  test's l2:4.09168"
```

```
## [1] "[52]:  test's l2:3.87773"
## [1] "[53]:  test's l2:3.79563"
## [1] "[54]:  test's l2:3.56741"
## [1] "[55]:  test's l2:3.42906"
## [1] "[56]:  test's l2:3.38298"
## [1] "[57]:  test's l2:3.31438"
## [1] "[58]:  test's l2:3.16048"
## [1] "[59]:  test's l2:2.90976"
## [1] "[60]:  test's l2:2.8169"
## [1] "[61]:  test's l2:2.72748"
## [1] "[62]:  test's l2:2.70435"
## [1] "[63]:  test's l2:2.54009"
## [1] "[64]:  test's l2:2.46328"
## [1] "[65]:  test's l2:2.36903"
## [1] "[66]:  test's l2:2.28287"
## [1] "[67]:  test's l2:2.24265"
## [1] "[68]:  test's l2:2.12184"
## [1] "[69]:  test's l2:2.06114"
## [1] "[70]:  test's l2:1.97336"
## [1] "[71]:  test's l2:1.94237"
## [1] "[72]:  test's l2:1.86936"
## [1] "[73]:  test's l2:1.78856"
## [1] "[74]:  test's l2:1.74868"
## [1] "[75]:  test's l2:1.79526"
## [1] "[76]:  test's l2:1.76653"
## [1] "[77]:  test's l2:1.74023"
## [1] "[78]:  test's l2:1.71875"
## [1] "[79]:  test's l2:1.68205"
## [1] "[80]:  test's l2:1.65467"
## [1] "[81]:  test's l2:1.63762"
## [1] "[82]:  test's l2:1.6162"
## [1] "[83]:  test's l2:1.59113"
## [1] "[84]:  test's l2:1.65067"
## [1] "[85]:  test's l2:1.63119"
## [1] "[86]:  test's l2:1.61819"
## [1] "[87]:  test's l2:1.60074"
## [1] "[88]:  test's l2:1.57118"
## [1] "[89]:  test's l2:1.55958"
## [1] "[90]:  test's l2:1.54979"
## [1] "[91]:  test's l2:1.54282"
## [1] "[92]:  test's l2:1.52958"
## [1] "[93]:  test's l2:1.51889"
## [1] "[94]:  test's l2:1.51252"
## [1] "[95]:  test's l2:1.50647"
## [1] "[96]:  test's l2:1.49688"
## [1] "[97]:  test's l2:1.49178"
## [1] "[98]:  test's l2:1.47429"
## [1] "[99]:  test's l2:1.45413"
## [1] "[100]:  test's l2:1.44562"
```

```
#model = lightgbm(boosting_type = 'gbdt', objective = "regression", metric = 'mae', dtrain, nrounds = 5
pred <- predict(model,dat = as.matrix(X_test))
length(pred)
```

```
## [1] 6046
```

```
length(y_test)
```

```
## [1] 6046
```

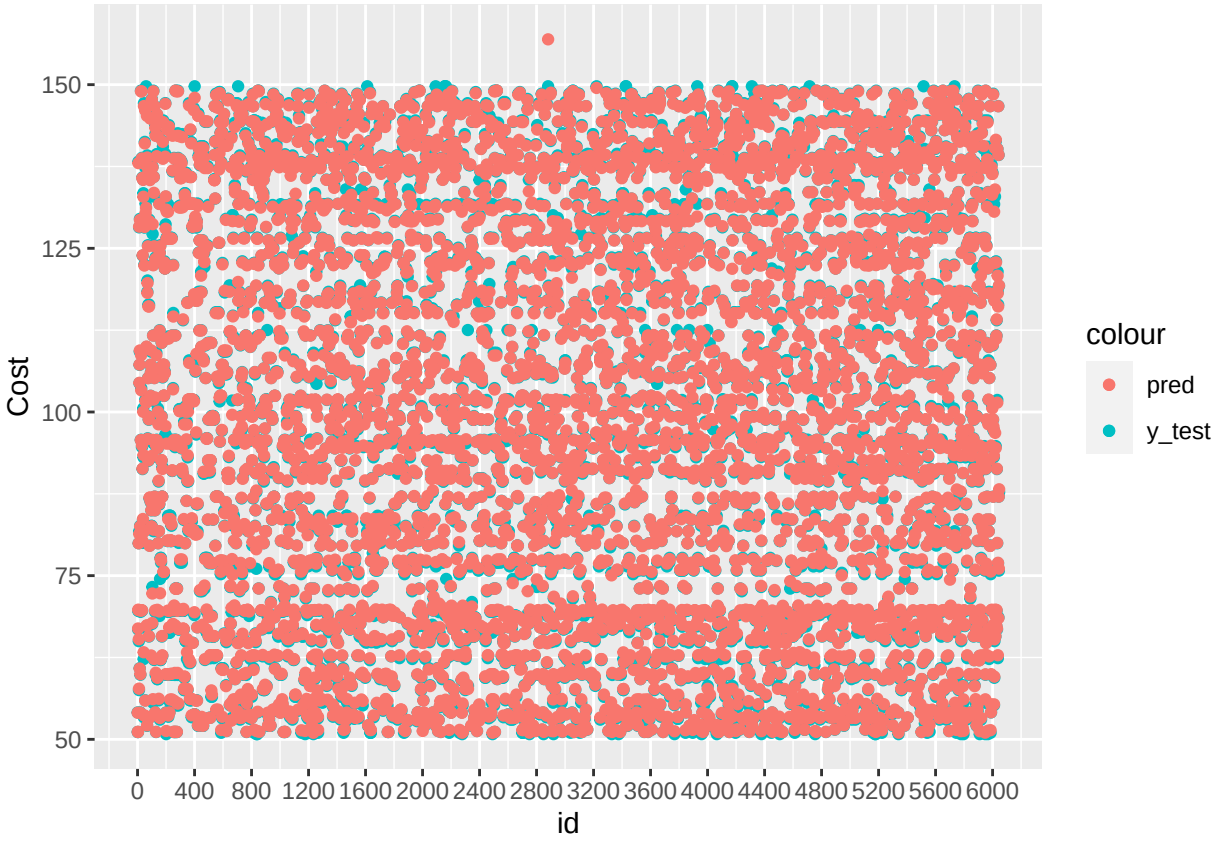
```
mse = mean((y_test - pred)^2)
mae = caret::MAE(y_test, pred)
rmse = caret::RMSE(y_test, pred)
cat("MSE: ", mse, "\nMAE: ", mae, "\nRMSE: ", rmse)
```

```
## MSE:  1.445622
## MAE:  0.2964298
## RMSE:  1.20234
```

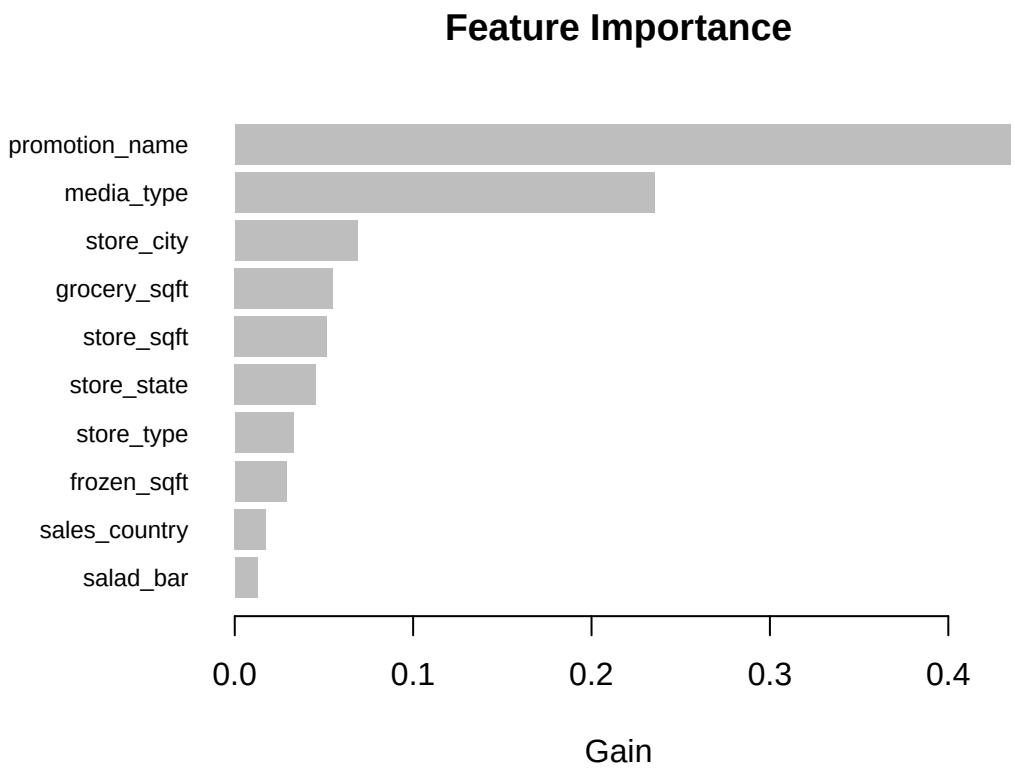
```
postResample(y_test,pred)
```

```
##      RMSE  Rsquared       MAE
## 1.2023400 0.9984284 0.2964298
```

```
df = data.frame(y_test, pred)
df$id = 1:nrow(df)
options(repr.plot.width = 20, repr.plot.height =2)
ggplot() + geom_point(data = df, aes(x = id, y = y_test, color = 'y_test')) +
  geom_point(data = df, aes(x=id, y = pred, color = 'pred')) + scale_x_continuous(breaks = seq(0, 6000,
  theme(plot.title = element_text(hjust = 0.5)) +
  ylab('Cost')
```




```
# feature importance
tree_imp = lgb.importance(model, percentage = TRUE)
lgb.plot.importance(tree_imp, top_n = 10L, measure = "Gain")
```


Feature Importance
promotion_name
media_type
store_city
grocery_sqft
store_sqft
store_state
store_type
frozen_sqft
sales_country
salad_bar
0.0
0.1
0.2
0.3
0.4
Gain