Stats Assignment 3 - R Notebook

Code

Hide

```
# Install package
install.packages("mgcv")
```

Libraries

Hide

```
library(tidyverse)
library(ggthemes)
library(GGally)
library(forecast)
library(olsrr)
library(caret)
library(MASS)
library(TTR)
library(zoo)
library(DHARMa)
library(flextable)
library(gridExtra)
library(pROC)
library(logistf)
library(mgcv)
```

```
Loading required package: nlme

Attaching package: ‘nlme’

The following object is masked from ‘package:dplyr’:

    collapse

The following object is masked from ‘package:forecast’:

    getResponse

This is mgcv 1.9-1. For overview type 'help("mgcv-package")'.
```

Set Custom Graph Theme

Hide

```
# Define the customized Stata theme for ggplot unity in presentation

custom_stata_theme <- theme_stata() +
  theme(
    legend.title = element_text(size = 10, face = "bold"),
    legend.text = element_text(size = 9))

# Set the customized theme as the default

theme_set(custom_stata_theme)
```

Explore Customers.csv Data Set

Hide

```
# read data set

cust <- read.csv("Customers.csv")
```

The variables included are:

- **Month** - Month
- **Customers** - Number of customers during the month

Hide

```
str(cust)
```

Hide

```
str(cust)
```

```
'data.frame':   72 obs. of  2 variables:
 $ Month    : chr  "2017-01-01" "2017-02-01" "2017-03-01" "2017-04-01" ...
 $ Customers: int  2550 2102 1494 2458 2705 2072 2042 2799 2100 2965 ...
```

Hide

```
# convert date character string to date using as.date

cust$Monthly <- as.Date(cust$Month, format = "%Y-%m-%d")
```

Plot 0 - Time series of customers across 72 months

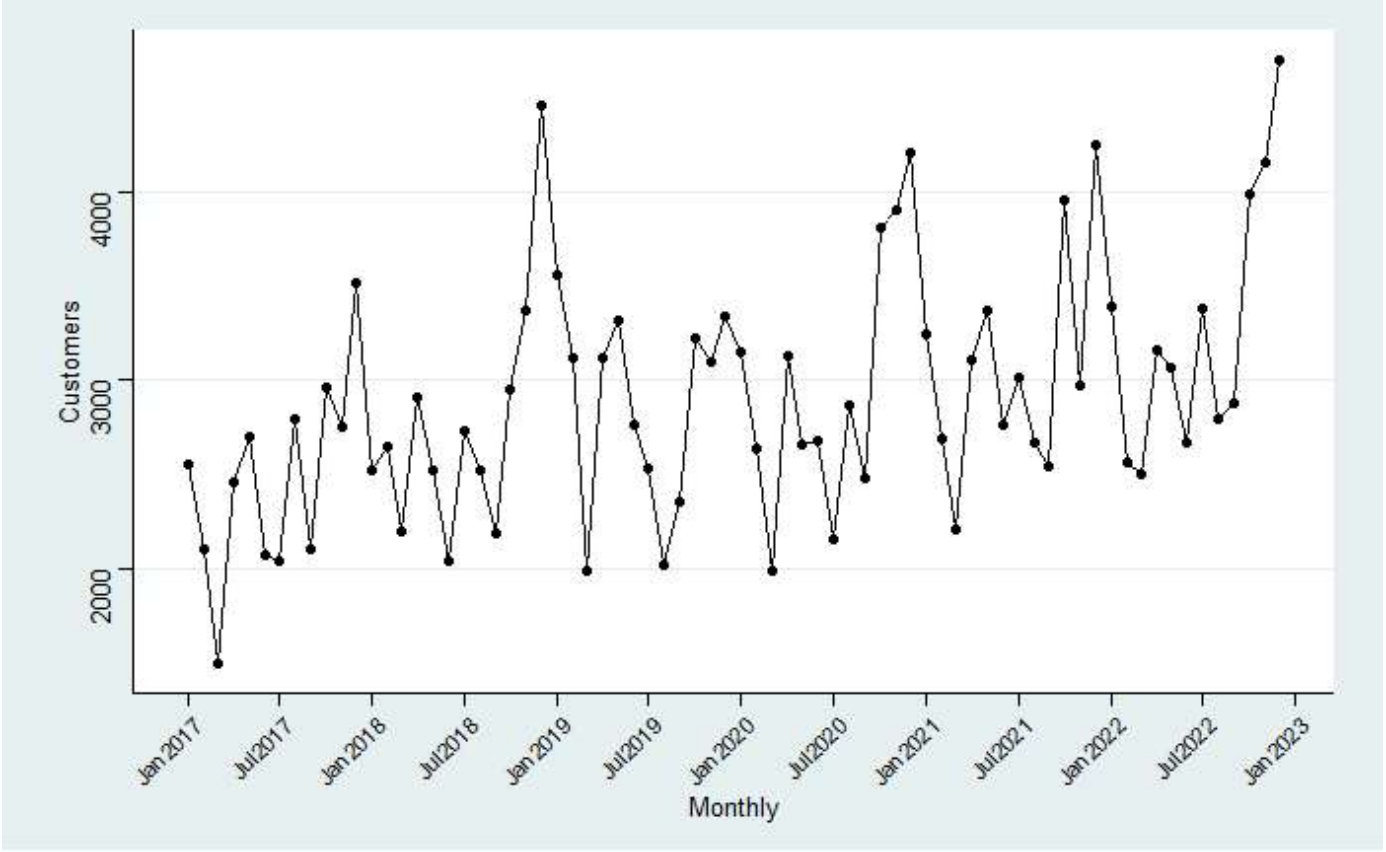
Hide

```
# Define the start date and interval for breaks in the time series for plotting

start_date <- as.Date("2017-01-01")
end_date <- as.Date("2023-01-01")
breaks <- seq.Date(start_date, end_date, by = "6 months")
```

Hide

```
# Plot 0 - plot time series data across full 72 months for visualisation

p0 <- ggplot(cust,aes(Monthly, Customers)) + geom_point() + geom_line()+
  scale_x_date(breaks=breaks, date_labels ="%b%Y") +
  theme(axis.text.x=element_text(size=8,angle=45,hjust=1))
print(p0)
ggsave("p0.png")
```

```
Saving 6.89 x 4.25 in image
```



Hide

```
# time series table for ease of viewing

series_ts <- ts(cust$Customers, frequency=12)
series_ts
```

```
   Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
1 2550 2102 1494 2458 2705 2072 2042 2799 2100 2965 2756 3516
2 2520 2647 2197 2906 2525 2042 2728 2521 2183 2950 3374 4458
3 3556 3120 1992 3124 3320 2758 2530 2019 2354 3226 3101 3337
4 3148 2633 1991 3134 2660 2680 2158 2870 2481 3805 3909 4203
5 3240 2691 2210 3107 3367 2765 3012 2671 2544 3961 2971 4254
6 3389 2562 2501 3160 3067 2666 3380 2792 2877 3989 4157 4701
```

Hide

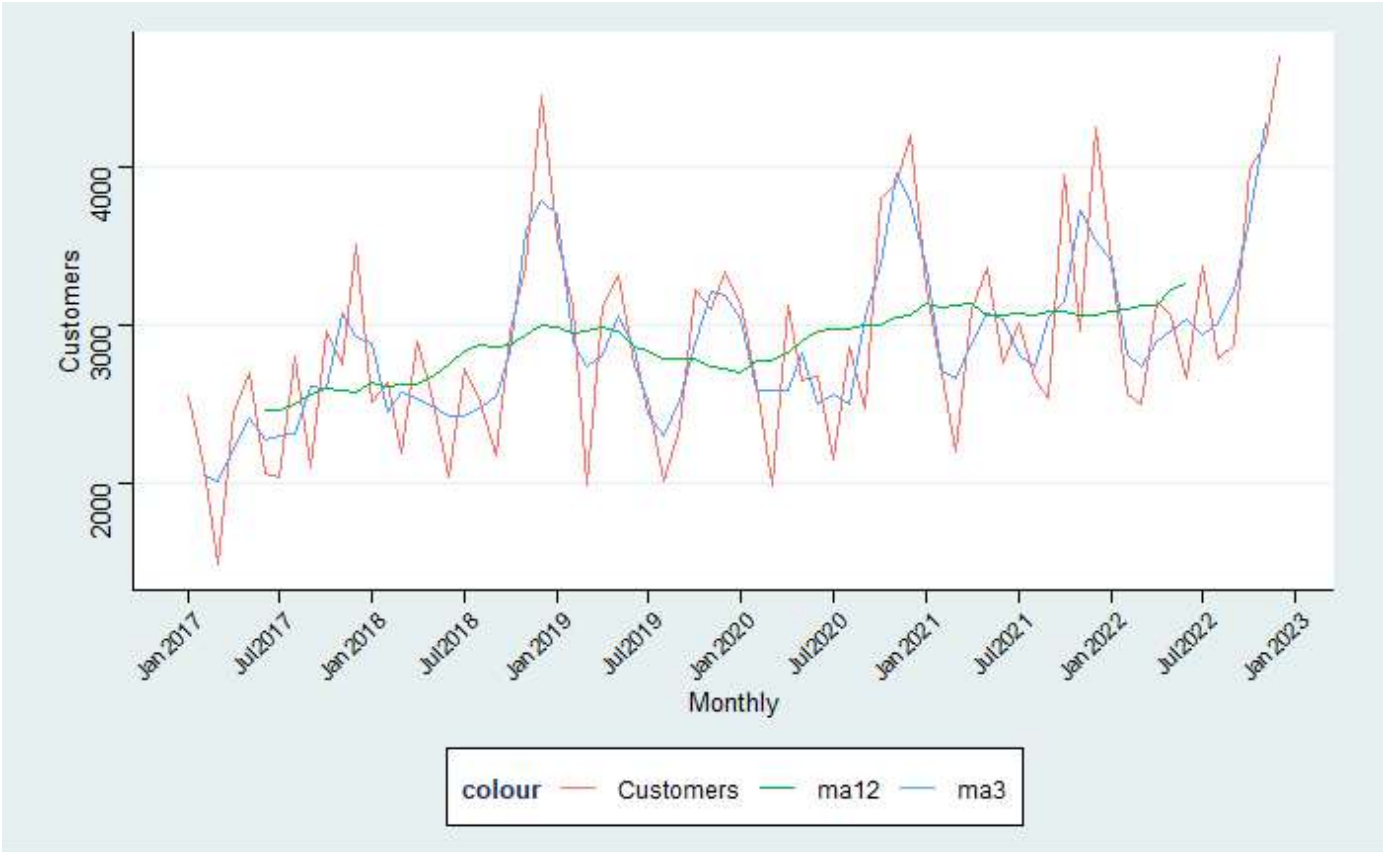
```
# create a model for moving averages of window sizes 3, 4, 6 and 12
#NOTE that this is NOT a dplyr filter. Turn dplyr OFF.

cust$ma3 <- filter(cust$Customers, rep(1/3,3))
cust$ma4 <- filter(cust$Customers, rep(1/4,4))
cust$ma6 <- filter(cust$Customers, rep(1/6,6))
cust$ma12<- filter(cust$Customers, rep(1/12,12))
```

Plot 1 Time Series - Customers with Moving Averages for 3 months and 12 months

Hide

```
# p1
# Plot 1 - plot data with moving averages of 3 & 12 months across 72 months

p1=ggplot(cust,aes(x=Monthly))
p1=p1 + geom_line(aes(y=Customers, color='Customers'), group=1)
p1=p1 + geom_line(aes(y=ma3, color='ma3'), group=1)
p1=p1 + geom_line(aes(y=ma12, color='ma12'), group=1)
p1=p1 + scale_x_date(breaks=breaks, date_labels ="%b%Y") +
  theme(axis.text.x=element_text(size=8,angle=45,hjust=1))
print(p1)
ggsave("plots/p1.png")
```

```
Saving 6.89 x 4.25 in image
```



Exponential Smoothing

1. **Calculation**: Applies exponentially decreasing weights to past data points, meaning more recent observations have a higher influence.
2. **Sensitivity**: More responsive to recent changes than both Moving Averages.

Hide

```
# Exponential Smoothing with a weighting of alpha=0.25

cust$ESW.25 <- fitted(ses(cust$Customers, alpha=0.25, initial='simple'))
View (cust)
```

Hide

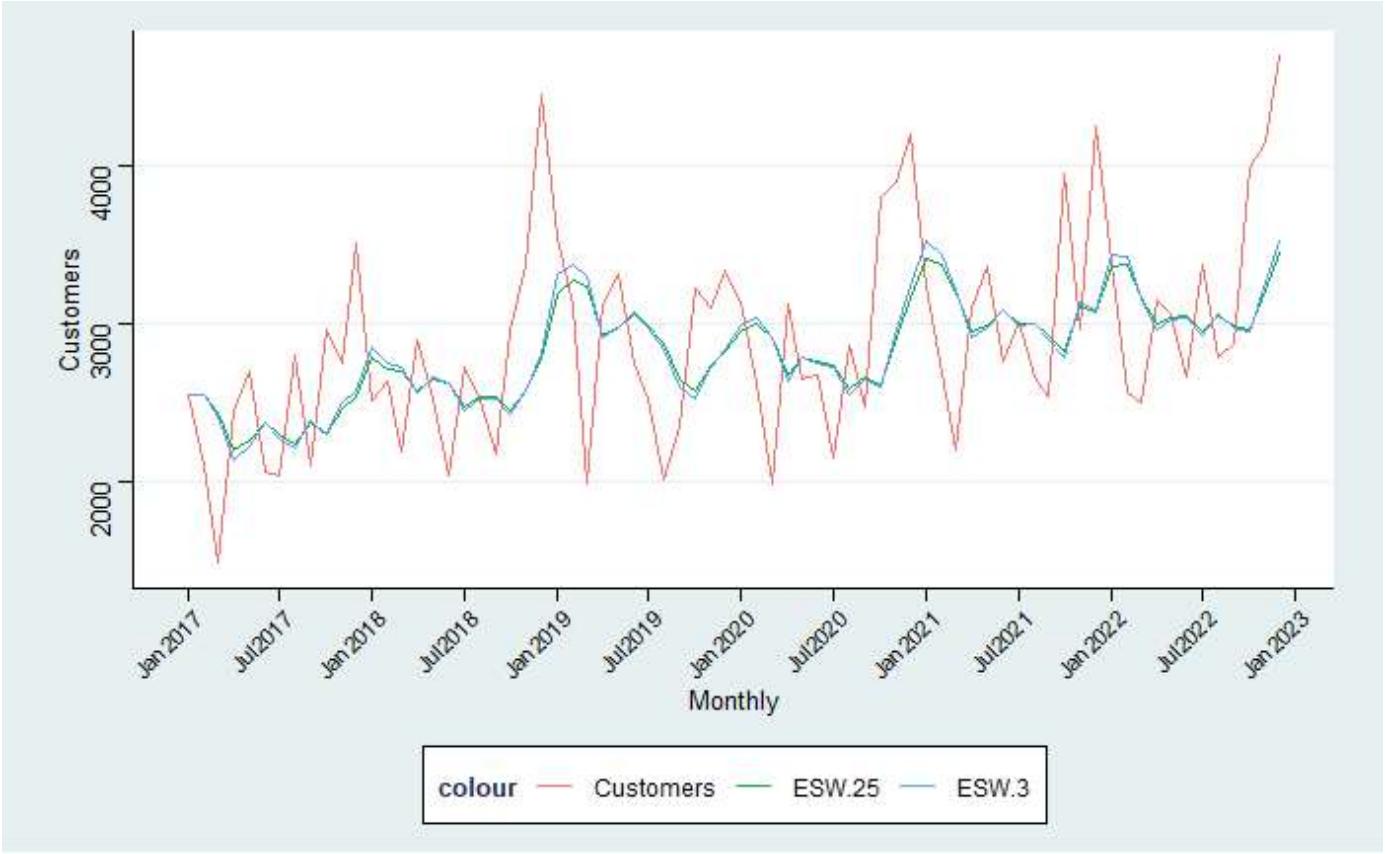
```
# Exponential Smoothing with a weighting of alpha=0.3

cust$ESW.3 <- fitted(ses(cust$Customers, alpha=0.3, initial='simple'))
View (cust)
```

Plot 2 Exponential Smoothing and Time series of customers

Hide

```
# p2
# Plot 2 - Plotting the Exponential Smoothing across 72 months

p2=ggplot(cust,aes(x=Monthly))
p2=p2 + geom_line(aes(y=Customers, color='Customers'), group=1)
p2=p2 + geom_line(aes(y=ESW.25, color='ESW.25'), group=1)
p2=p2 + geom_line(aes(y=ESW.3, color='ESW.3'), group=1)
p2=p2 + scale_x_date(date_labels ="%b%Y", breaks=breaks) +
  theme(axis.text.x=element_text(size=8,angle=45,hjust=1))
print(p2)
ggsave("plots/p2.png")
```

```
Saving 6.89 x 4.25 in image
```



Seasonal Indices

Hide

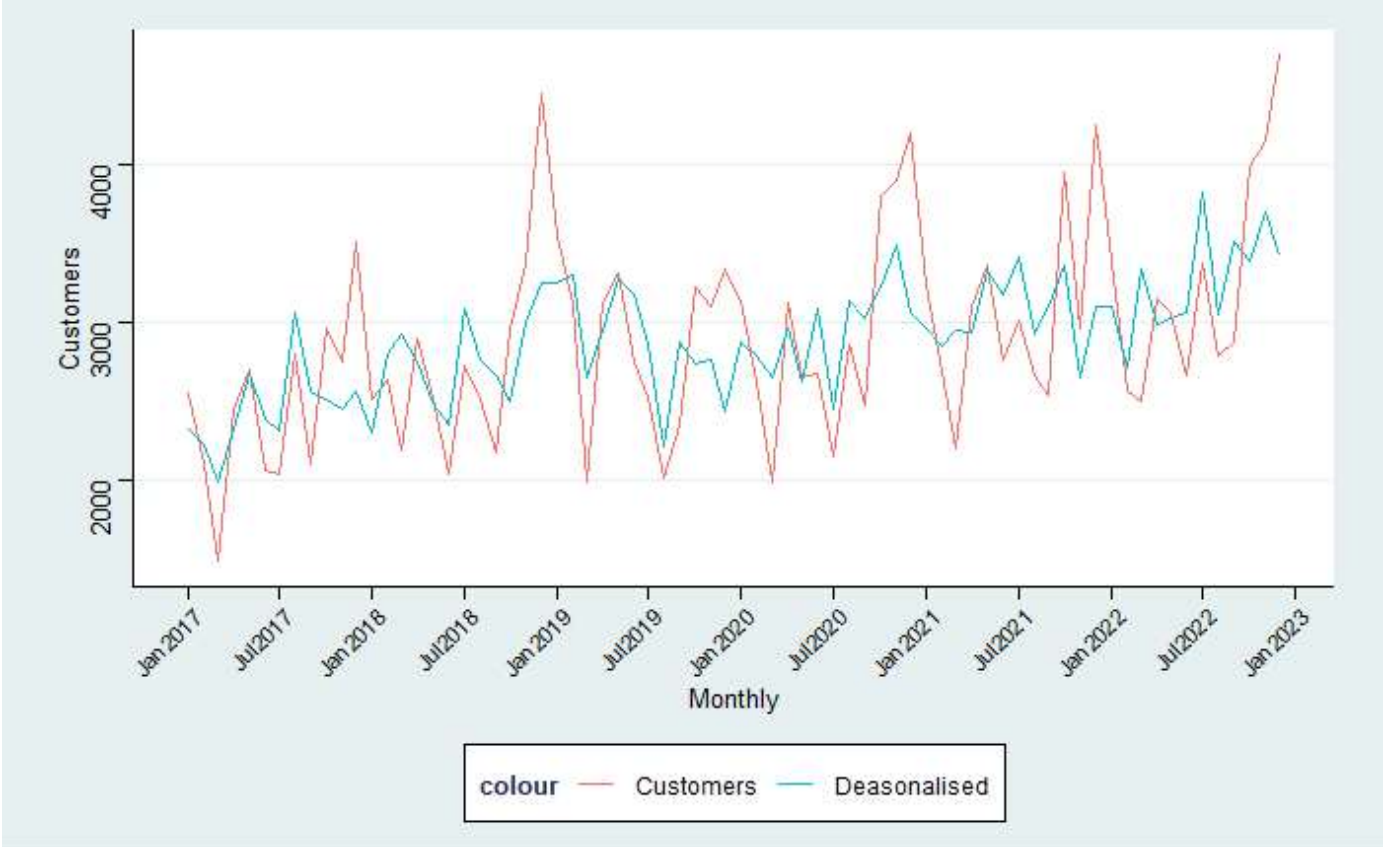
```
# Convert numeric vector to a time series object
CustomerTS = ts(cust$Customers, start=c(2017,1), frequency=12)

# Decompose the time series
Customer.components = decompose(CustomerTS, type="multiplicative")

# Extract seasonal indices and add column to the data file
cust$SeasonalIndex = Customer.components$seasonal

# Add de-seasonalised series column to the data file
cust$Deseasonalised = cust$Customers/cust$SeasonalIndex

View(cust)
```

Plot 3 De-seasonalised time series

Hide

```
# p3
# Plot 3 - Plotting deseasonalised time series across 72 months

p3=ggplot(cust,aes(x=Monthly))
p3=p3 + geom_line(aes(y=Customers, color='Customers'), group=1)
p3=p3 + geom_line(aes(y=Deseasonalised, color='Deasonalised'), group=1)
p3=p3 + scale_x_date(date_labels ="%b%Y", breaks=breaks) +
  theme(axis.text.x=element_text(size=8,angle=45,hjust=1))
p3
ggsave("plots/p3.png")
```

```
Saving 6.89 x 4.25 in image
```



Train/Test Split

Hide

```
# splitting data set with a train set of 60 months + test set of 12 months
# trying to avoid disrupting the temporal structure by splitting at full years
# rather than a pure 80/20 split

test_set_start_date <- "2022-01-01"
train_set <- subset(cust, Monthly < test_set_start_date)
test_set <- subset(cust, Monthly >= test_set_start_date)

dim(train_set)
```

```
[1] 60 11
```

Hide

```
dim(test_set)
```

```
[1] 12 11
```

Plot 4 Visualising the Train and Test Sets

Hide

```
# p4
# Plot 4 Visualising the Train and Test Sets
p4 <- ggplot() +
  geom_line(data = train_set, aes(x = Monthly, y = Customers, color = "Training"), size = .7)
+
  geom_line(data = test_set, aes(x = Monthly, y = Customers, color = "Testing"), size = .7) +
  labs( x = "Date", y = "Customers") +
  scale_color_manual(values = c("Training" = "#12355B", "Testing" = "#D72638"), name = "Custo
mers") +
  scale_x_date(limits=as.Date(c("2017-01-01", NA)), date_labels ="%b%Y", breaks = breaks) +
  theme(axis.text.x=element_text(size=8,angle=45,hjust=1))
p4
ggsave("plots/p4.png")
```

```
Saving 6.89 x 4.25 in image
```

The increasing **trend** and **seasonal** components are clearly visible. December ’19-January ’20 didn’t have the same seasonal spike as other years. Initially I thought it may have been the outcome of the first covid lockdown, but it is too early for that, so without further data, it remains a random instance.

Seasonal Indices for Training Set

Hide

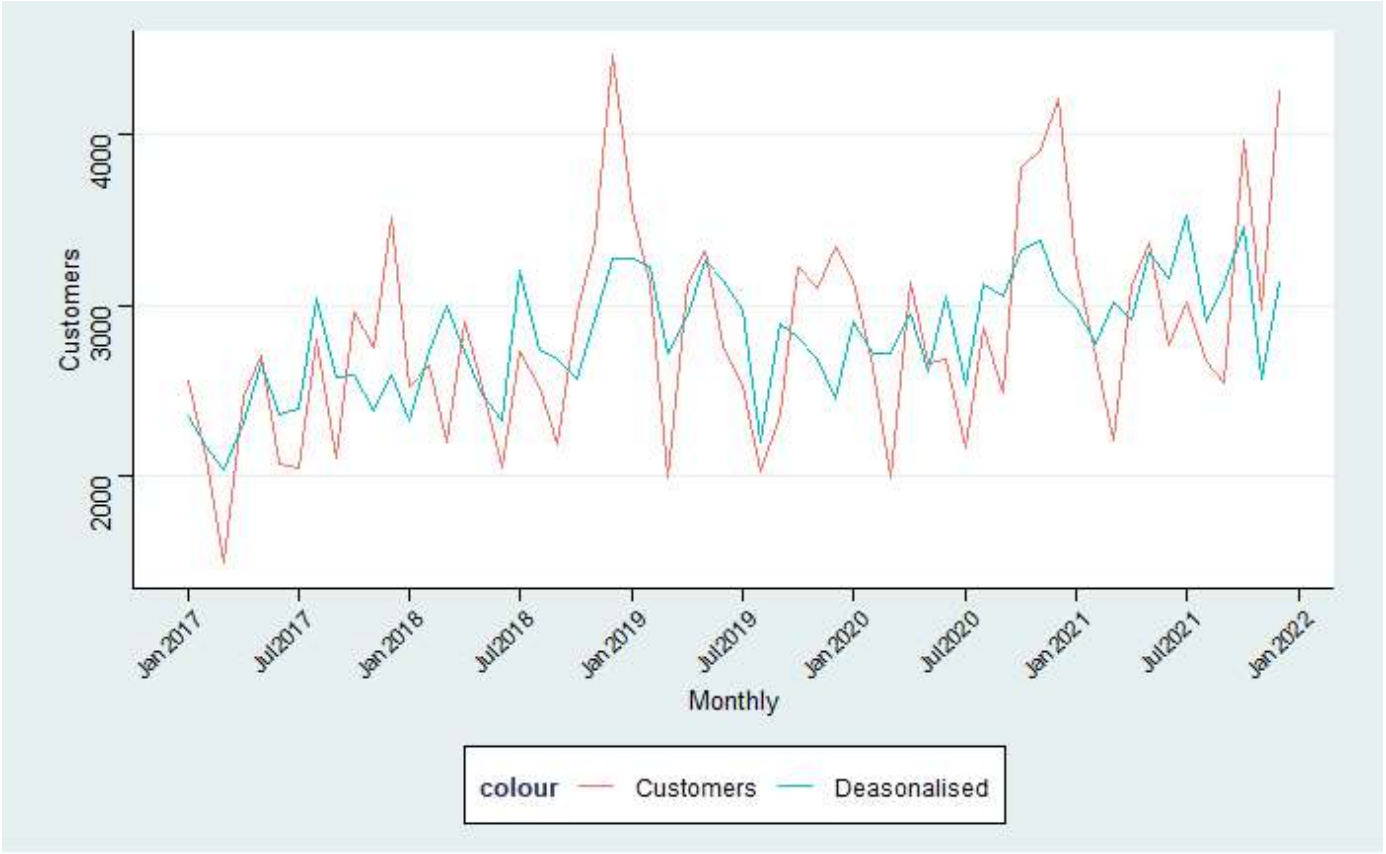
```
# Convert numeric vector to a time series object for training set
cust_train_ts = ts(train_set$Customers, start=c(2017,1), frequency=12)

# Convert numeric vector to a time series object for test set
cust_test_ts = ts(test_set$Customers, start=c(2022,1), frequency=12)

# Decompose the time series
cust_ts.components = decompose(cust_train_ts, type="multiplicative")

# Extract seasonal indices and add column to the data file
train_set$SeasonalIndex = cust_ts.components$seasonal

# Add de-seasonalised series column to the data file
train_set$Deseasonalised = train_set$Customers/train_set$SeasonalIndex

View(train_set)
```

Plot 5 - De-seasonalised time series for the Training Set

Hide

```
# p5
# Plot 5 - Plotting deseasonalised time series for the Training Set

p5=ggplot(train_set,aes(x=Monthly))
p5=p5 + geom_line(aes(y=Customers, color='Customers'), group=1)
p5=p5 + geom_line(aes(y=Deseasonalised, color='Deasonalised'), group=1)
p5=p5 + scale_x_date(date_labels ="%b%Y", breaks=breaks) +
  theme(axis.text.x=element_text(size=8,angle=45,hjust=1))
print(p5)
ggsave("plots/p5.png")
```

```
Saving 6.89 x 4.25 in image
```



Fit an ARIMA Model for forecasting to the Test data subset

The auto.arima function from the forecast package can help find the best model for forecasting.

Hide

```
# Trying the auto.arima function to help find best model for forecasting
# Fit the arima model on the traning data

fit_arima <- auto.arima(cust_train_ts)

# Forecast using the arima model on the test data
forecast_arima <- forecast(fit_arima, h = length(cust_test_ts))

# Print the summary of the arima model
summary(fit_arima)
```

```
Series: cust_train_ts
ARIMA(0,0,0)(1,1,0)[12] with drift

Coefficients:
         sar1    drift
      -0.5974  11.7636
s.e.   0.1088   3.2137

sigma^2 = 154354:  log likelihood = -356.46
AIC=718.93   AICc=719.47   BIC=724.54

Training set error measures:
                    ME     RMSE      MAE        MPE     MAPE      MASE      ACF1
Training set 12.44541 344.0025 244.2379 -0.5239984 8.530318 0.6093887 0.1853196
```

Fit a Holt-Winters Model for forecasting to the Test data to compare with ARIMA

Hide

```
# Fit the Holt-Winters model
fit_hw <- HoltWinters(cust_train_ts)

# Forecast using the Holt-Winters model
forecast_hw <- forecast(fit_hw, h = length(cust_test_ts))

# Print the summary of the Holt-Winters model
summary(fit_hw)
```

```
             Length Class  Mode
fitted       192    mts    numeric
x             60    ts     numeric
alpha          1    -none- numeric
beta           1    -none- numeric
gamma          1    -none- numeric
coefficients  14    -none- numeric
seasonal       1    -none- character
SSE            1    -none- numeric
call           2    -none- call
```

Hide

```
# Calculate accuracy metrics for ARIMA model
acc_results_arima <- round(accuracy(forecast_arima, cust_test_ts),2)

# Calculate accuracy metrics for Holt-Winters model
acc_results_hw <- round(accuracy(forecast_hw, cust_test_ts),2)

acc_results_arima
```

```
                ME   RMSE    MAE   MPE MAPE MASE  ACF1 Theil's U
Training set 12.45 344.00 244.24 -0.52 8.53 0.61  0.19        NA
Test set     34.31 287.79 240.07  0.32 7.54 0.60 -0.37      0.52
```

Hide

```
acc_results_hw
```

```
                 ME   RMSE    MAE   MPE  MAPE MASE ACF1 Theil's U
Training set -19.96 395.14 304.05 -2.18 10.78 0.76 0.10        NA
Test set      31.84 300.42 247.99 -0.11  7.60 0.62 0.05       0.5
```

Neither model outshines the other. The ARIMA has better performance in terms of the error metrics like RMSE, MAE and MAPE especially in the test set. Holt-Winters is better at reducing bias (ME), autocorrelation in the residuals (ACF1) and has a slightly better forecast (.02) relative to the naive model (Theil's U).

I am choosing ARIMA as it appears to be the better choice here because of its lower overall error metrics (RMSE, MAE, and MAPE) on both training and test sets.

ARIMA Forecast next 36months and visualise against actual data

Hide

```
# forecast the next 36 months using arima fitted model
# without the confidence interval
forecast_values <- forecast(fit_arima, h = 36, level=c(80,95))
```

Hide

```
# convert forecast to a data frame

forecast_arima_df <- data.frame(
  Month = time(forecast_values$mean),
  Forecast = as.numeric(forecast_values$mean),
  Lower_80 = as.numeric(forecast_values$lower[,1]),
  Upper_80 = as.numeric(forecast_values$upper[,1]),
  Lower_95 = as.numeric(forecast_values$lower[,2]),
  Upper_95 = as.numeric(forecast_values$upper[,2])
)
forecast_arima_df
```

Month <dbl>	Forecast <dbl>	Lower_80 <dbl>	Upper_80 <dbl>	Lower_95 <dbl>	Upper_95 <dbl>
2022.000	3410.534	2907.040	3914.029	2640.507	4180.562
2022.083	2881.847	2378.353	3385.341	2111.819	3651.875
2022.167	2304.660	1801.166	2808.154	1534.632	3074.688
2022.250	3348.629	2845.135	3852.123	2578.601	4118.657
2022.333	3170.112	2666.617	3673.606	2400.084	3940.139

Month <dbl>	Forecast <dbl>	Lower_80 <dbl>	Upper_80 <dbl>	Lower_95 <dbl>	Upper_95 <dbl>
2022.417	2939.716	2436.222	3443.211	2169.689	3709.744
2022.500	2727.289	2223.794	3230.783	1957.261	3497.316
2022.583	3015.388	2511.894	3518.882	2245.360	3785.416
2022.667	2731.860	2228.366	3235.354	1961.832	3501.888
2022.750	4093.299	3589.804	4596.793	3323.271	4863.326

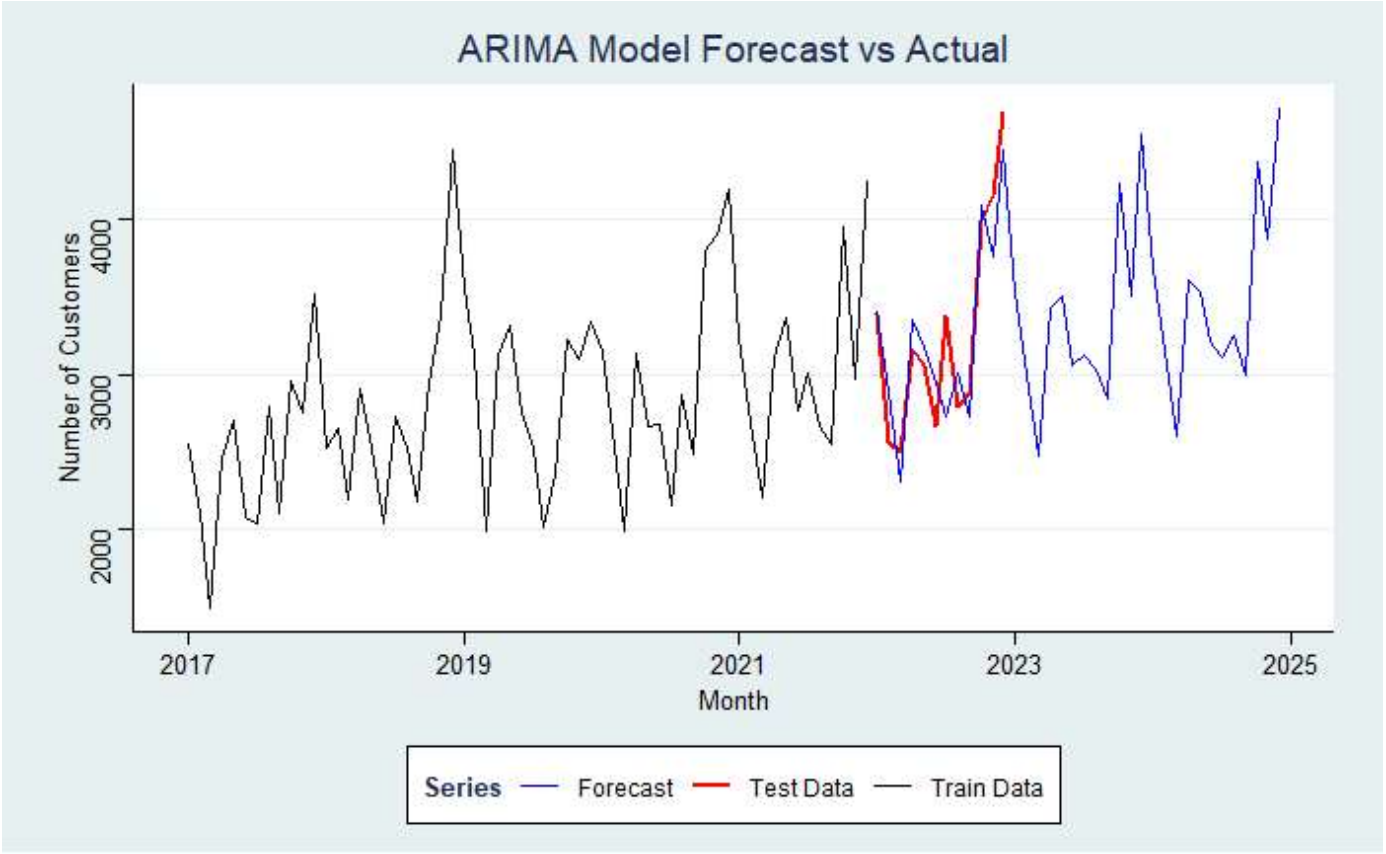
1-10 of 36 rows Previous 1 2 3 4 Next

Plot 6 ARIMA Model Actual & Forecast

Hide

```
# p6
# Plot the 5yrs training data, 1 yr test data, and 2yr forecast with 1 yr overlap
# Trying autoplot for the first time
p6=autoplot(forecast_values, PI=FALSE) +
  autolayer(cust_test_ts, series = "Test Data",  size=1) +
  autolayer(cust_train_ts, series = "Train Data") +
  autolayer(forecast_values$mean, series = "Forecast") +
  scale_color_manual(name = "Series", values = c("Train Data" = "black", "Test Data" = "red",
"Forecast" = "blue")) +
  ggtitle("ARIMA Model Forecast vs Actual") +
  xlab("Month") +
  ylab("Number of Customers") +
  theme(legend.position = "bottom")
p6
ggsave("plots/p6.png")
```

```
Saving 6.89 x 4.25 in image
```



Plot 7 - Residuals from ARIMA model

Hide

```
# p7
# plots checking residuals for the fit_arima model
p7=checkresiduals(fit_arima)
```

```

    Ljung-Box test

data:  Residuals from ARIMA(0,0,0)(1,1,0)[12] with drift
Q* = 15.978, df = 11, p-value = 0.142

Model df: 1.   Total lags used: 12
```

Hide

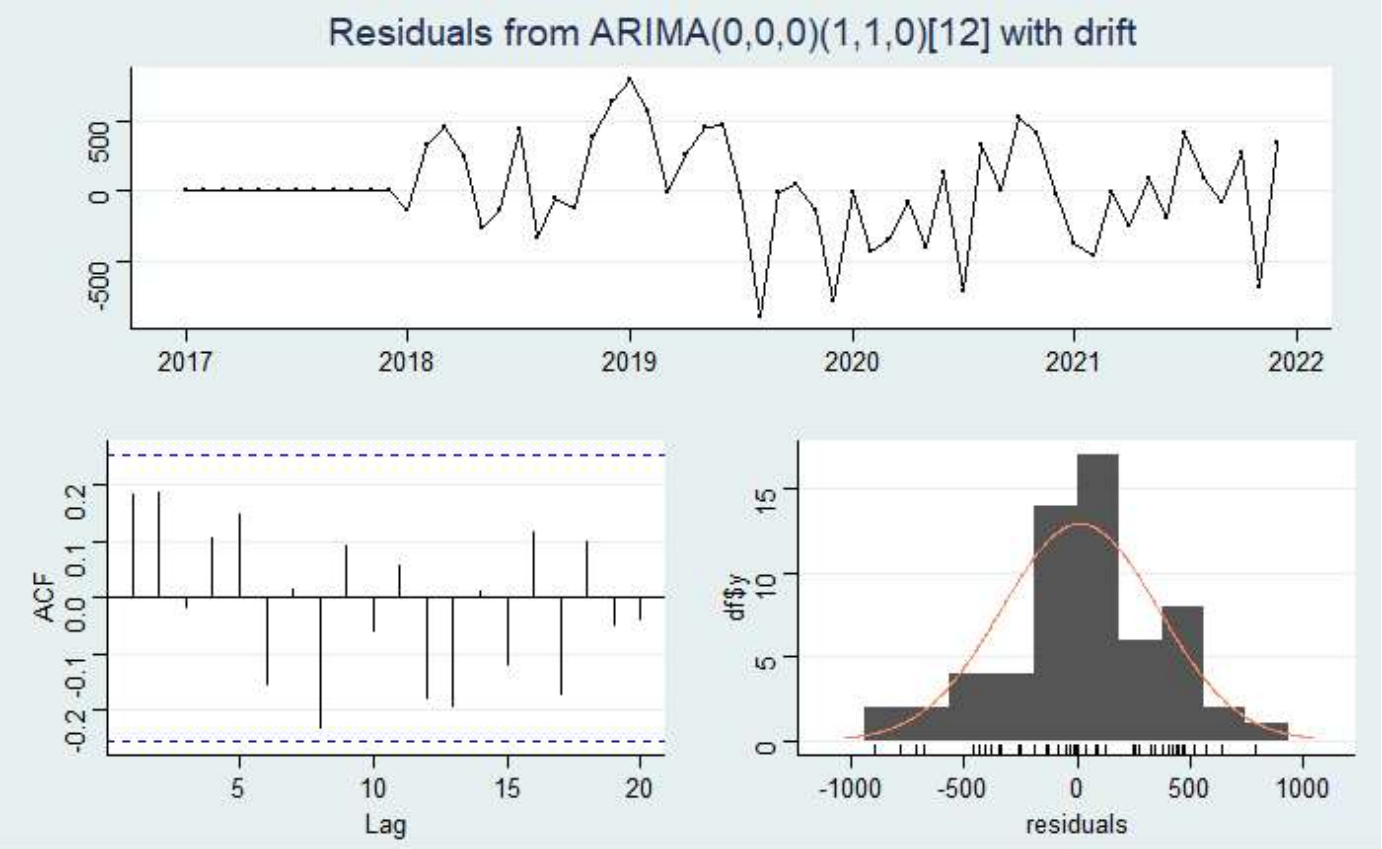
```
p7
```

```

    Ljung-Box test

data:  Residuals from ARIMA(0,0,0)(1,1,0)[12] with drift
Q* = 15.978, df = 11, p-value = 0.142
```

Hide

```
ggsave("plots/p7.png")
```

```
Saving 6.89 x 4.25 in image
```



Hide

```
# rounding the forecast values to integers

forecast_arima_df$Forecast <- round(forecast_arima_df$Forecast)

# then creating a timeseries table

fit_arima_ts <- ts(forecast_arima_df$Forecast, frequency=12)
fit_arima_ts
```

```
   Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
1 3411 2882 2305 3349 3170 2940 2727 3015 2732 4093 3757 4449
2 3534 2993 2474 3430 3513 3061 3123 3035 2845 4240 3513 4558
3 3686 3152 2598 3607 3534 3214 3112 3249 3003 4378 3884 4718
```

Hide

```
# converting the decimal year format to conventional date and saving to .csv

forecast_arima_df$Month <- as.yearmon(forecast_arima_df$Month)
write.csv(forecast_arima_df, file="forecast_arima.csv")
```

Hide

```
# formatting the dataframe as a table using flextable and save as image

flextable(forecast_arima_df) %>%
  theme_vanilla() %>%
save_as_image(path = "images/forecast_customers.png")
```

```
[1] "images/forecast_customers.png"
```

Hide

```


# Plot the original data
p1 <- ggplot(plot_data, aes(x = Time, y = Original)) +
  geom_line(color = "blue") +
  ggtitle("Original Data") +
  scale_y_continuous(limits = c(min(plot_data$Original), max(plot_data$Original)))
```

```
Error: object 'plot_data' not found
```

Exploring Restaurant.csv

Hide

```
rests <- read.csv("Restaurants.csv")
```

Hide

```
str(rests)
```

```
'data.frame':   500 obs. of  20 variables:
 $ ID           : int  1002 1005 1006 1024 1028 1033 1052 1054 1071 1080 ...
 $ Location     : chr  "Suburbs" "City" "City" "City" ...
 $ Menu         : chr  "Mexican" "Mexican" "Chinese" "Indian" ...
 $ Days         : int  5 6 6 6 7 7 6 7 6 6 ...
 $ Sunday       : num  0 0 0 1 1 1 1 1 1 0 ...
 $ Monday       : chr  "No" "Yes" "Yes" "No" ...
 $ Distance     : num  1 0.5 0.7 0.7 0.5 1 3.1 1.3 0.5 1 ...
 $ Price        : num  23.6 21.2 24.1 94 56.5 ...
 $ Customers    : int  98 150 121 32 68 33 46 170 67 64 ...
 $ Food         : num  3.27 2.53 3.8 3.77 4.04 4.11 3.17 3.41 3.41 4.2 ...
 $ Atmosphere   : num  2.67 2.95 3.34 3.9 2.7 4.25 3.35 3.22 3.48 4.14 ...
 $ Service      : num  1.41 2.91 2.79 3.42 3.49 3.48 3.78 3.49 4.76 3.27 ...
 $ Sold         : int  0 0 1 0 0 0 0 0 0 1 ...
 $ SoldStatus   : chr  "retained" "retained" "sold" "retained" ...
 $ Rating       : num  7.35 8.39 9.93 11.09 10.23 ...
 $ Price_sqrt   : num  4.86 4.6 4.91 9.7 7.52 ...
 $ Price_log    : num  3.16 3.05 3.18 4.54 4.03 ...
 $ TopRating    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ EstDayTakings: num  2313 3180 2916 3008 3842 ...
 $ Sunday.y     : chr  "No" "No" "No" "Yes" ...
```

The variables included are:

- **Location** - City/Suburban
- **Menu** - Predominant style of food
- **Days** - Days open per week
- **Sunday** - Open Sunday?
- **Monday** - Open Monday?
- **Distance** - Distance (km) to nearest restaurant with same menu
- **Price** - Average price ($) per customer
- **Customers** - Average number of customers per day
- **Food** - Average rating of food
- **Atmosphere** - Average rating of atmosphere
- **Service** - Average rating of service
- **Sold** - Changed owners in the last 12 month

Hide

```
head(rests)
```

Hide

```
summary(rests)
```

Exploring bar charts of categorical variables

Hide

```
# Plot r2 - Exploring distribution of Location
r2 = ggplot(rests, aes(x=Location)) +
geom_bar(fill="deepskyblue4") + geom_text(stat="prop", aes(label=sprintf("%.1f%%", after_stat
((prop)*100))),vjust=1, colour="white")+
  theme(axis.title.y = element_blank())
r2
ggsave("plots/r2.png")
```

Hide

```
# Plot r3 - Exploring distribution of Menu
r3 = ggplot(rests, aes(x=Menu)) +
geom_bar(fill="deepskyblue4") +
geom_text(stat="prop", aes(label=sprintf("%.1f%%",after_stat((prop)*100))),
          vjust=1, colour="white", size = 3) +
  theme(axis.text.x=element_text(size=8,angle=45,hjust=1),axis.title.y = element_blank())
r3
ggsave("plots/r3.png")
```

Hide

```
menu.table <- table(rests$Menu)
prop.table(menu.table)
```

Hide

```
# Plot r4 - Exploring distribution of Sold(1) in last 12 months
r4 = ggplot(rests, aes(x=factor(Sold, levels=c(0,1),
                                labels=c("Retained", "Sold"))))+
  geom_bar(fill="deepskyblue4") +
  geom_text(stat="prop", aes(label=sprintf("%.1f%%", after_stat((prop)*100))),
            vjust=1, colour="white")+
  scale_x_discrete(name="Sales Status")+
  theme(axis.title.y = element_blank())
r4
ggsave("plots/r4.png")
```

Exploring bivariate relationships

Plot R1 - Scatterplot Matrix Price: Customers: Distance

Hide

```
# R1 Examine bivariate relationships
# Scatterplot Matrix
r1 = data.frame(Price = rests$Price, Customers = rests$Customers,
                 Distance = rests$Distance)
ggpairs(r1[,1:3])
ggsave("plots/r1.png")

# Note that all three have skews to the right and will require transforming
```

These variable are all positively skewed and, depending on statistical testing planned, may require transformation. The relationship between price and customers is strongly correlated but look at that shape!

Hide

```
# taking an exploratory look at the correlation between Customers and Price
r1e = ggplot(rests, aes(Price, Customers)) +
  geom_point() +
  geom_smooth(method=lm, se=TRUE, colour= "deepskyblue4")
r1e
ggsave("plots/r1e.png")
```

Plot R1a Scatterplot Matrix Price: Service: Atmosphere: Food

Hide

```
# Plot R1a Examine bivariate relationships
# Scatterplot Matrix
r1a = data.frame(Price = rests$Price, FoodRating = rests$Food,
                 ServiceRating = rests$Service,
                 AtmosphereRating = rests$Atmosphere)
ggpairs(r1a[,1:4])
ggsave("plots/r1a.png")
```

Plot R1a Scatterplot Matrix Customers: Service: Atmosphere: Food

Hide

```
# Plot R1b Examine bivariate relationships
# Scatterplot Matrix
r1b = data.frame(Customers = rests$Customers,
                  FoodRating = rests$Food,
                 ServiceRating = rests$Service,
                 AtmosphereRating = rests$Atmosphere)
ggpairs(r1b[,1:4])
ggsave("plots/r1b.png")
```

If thinking of using ratings to predict customers…. then the above shows a problem. Highly correlated predictors (Food and atmosphere) can lead to collinearity issues and this can greatly increase the model variance, especially in the context of regression. In some cases, there could be relationships between multiple predictor variables and this is called multicollinearity. Having correlated variables will result in unnecessarily complex models with more than necessary predictor variables.

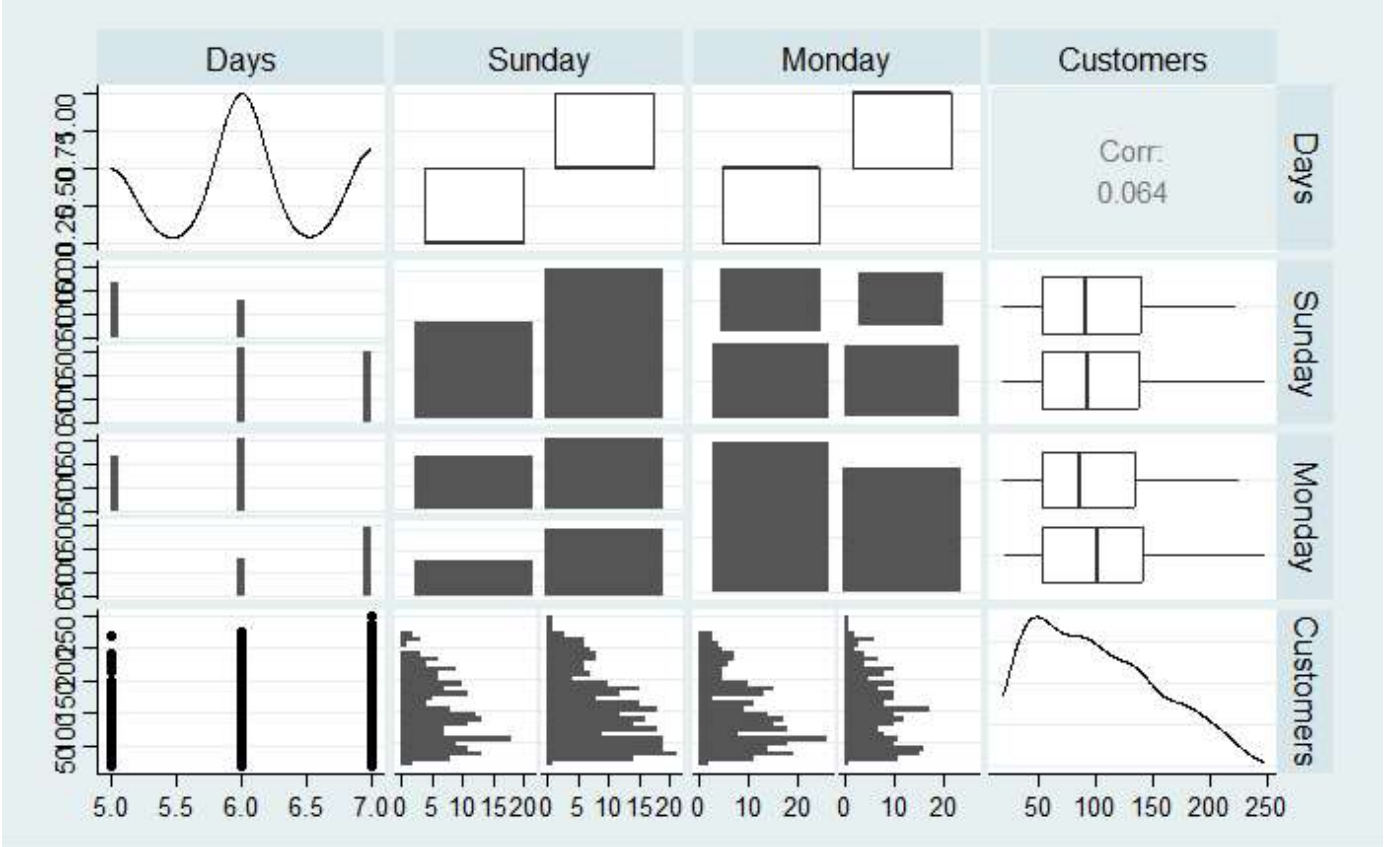
The potential solutions include the following:

1. Remove some of the highly correlated independent variables.
2. Linearly combine the independent variables, such as adding them together.
3. Partial least squares regression uses principal component analysis to create a set of uncorrelated components to include in the model.

Hide

```
# Plot R1a Examine bivariate relationships
# Scatterplot Matrix
r1e = data.frame(Days = rests$Days, Sunday = rests$Sunday,
                 Monday = rests$Monday,
                 Customers = rests$Customers)
ggpairs(r1e[,1:4])
ggsave("plots/r1e.png")
```

```
Saving 7.29 x 4.5 in image
```



Hide

```
# Plot R1c Examine bivariate relationships
# Scatterplot Matrix Customers-Food-Service-Atmosphere
r1c = data.frame(Days = rests$Days,
                 FoodRating = rests$Food,
                 ServiceRating = rests$Service,
                 AtmosphereRating = rests$Atmosphere)
ggpairs(r1c[,1:4])
ggsave("plots/r1c.png")
```

Hide

```
# Plot R1d Examine bivariate relationships
# Scatterplot Matrix Customers-Food-Service-Atmosphere
r1d = data.frame(Location = rests$Location,
                 Customer = rests$Customers,
                 Price = rests$Price,
                 Days = rests$Days)
ggpairs(r1d[,1:4])
ggsave("plots/r1d.png")
```

Bivariate categorical Visualisations

Hide

```
# Plot r5 Open on Sunday with days open per Week
r5 = ggplot(rests, aes(x=Sunday))+
geom_bar(aes(fill=Sunday), alpha = 1)+
facet_grid(~ Days)+
scale_fill_brewer(palette="Paired")+
geom_text(stat="prop", aes(label=sprintf("%.1f%%",after_stat((prop)*100))),vjust=1, colour="w
hite")+
theme(axis.title.y = element_blank(), legend.position="none")+
  xlab("Open Sunday?")+
  labs(title="Days per week Open", subtitle = "with a focus on Sundays")

r5
ggsave("plots/r5.png")
```

Hide

```
#r5a Open on Monday
# with days open per Week
r5a = ggplot(rests, aes(x=Monday))+
geom_bar(aes(fill=Monday), alpha = 1)+
facet_grid(~ Days)+
scale_fill_brewer(palette="Paired")+
geom_text(stat="prop", aes(label=sprintf("%.1f%%",after_stat((prop)*100))),vjust=1, colour="w
hite")+
theme(axis.title.y = element_blank(), legend.position="none")+
  xlab("Open Monday?")+
  labs(title="Days per week Open", subtitle = "with a focus on Mondays")

r5a
ggsave("plots/r5a.png")
```

Hide

```
#r5b Days per Week
# with Suburb vs City
# and proportion of restaurants open 5 days vs 6 vs 7.

r5b = ggplot(rests, aes(x=Location))+
geom_bar(aes(fill=Location), alpha=1)+
facet_grid(~ Days)+
scale_fill_brewer(palette="Paired")+
geom_text(stat="count", aes(label= after_stat(count)),vjust=1, colour="white")+
theme(axis.title.y = element_blank(), legend.position = "none")+
  labs(title="Days per week Open", subtitle = "with a focus on Location")

r5b
ggsave("plots/r5b.png")
```

Hide

```
#r5b Sundays
# with Suburb vs City
# and count of restaurants open Sundays.

r5b = ggplot(rests, aes(x=Location))+
geom_bar(aes(fill=Location), alpha=1)+
facet_grid(~ Sunday)+
scale_fill_brewer(palette="Paired")+
geom_text(stat="count", aes(label= after_stat(count)),vjust=1, colour="white")+
theme(axis.title.y = element_blank(), legend.position = "none")+
  labs(title="Open Sunday", subtitle = "with a focus on Location")

r5b
ggsave("plots/r5b.png")
```

Hide

```
#r5c Menu Locations
# with Suburb vs City
r5c = ggplot(rests, aes(x=Location))+
geom_bar(aes(fill=Location), alpha = 1)+
facet_wrap(~ Menu)+
scale_fill_brewer(palette="Paired")+
geom_text(stat="count", aes(label= after_stat(count)),vjust=1,
                            colour="white", size=3)+
theme(axis.title.y = element_blank(), legend.position = "none")+
  labs(title="Menu Cuisine", subtitle = "with a focus on Location")

r5c
ggsave("plots/r5c.png")
```

Hide

```
#r5d Sold or Retained
# with Suburb vs City
r5d = ggplot(rests, aes(x=factor(Sold, levels=c(0,1),
                                 labels=c("Retained", "Sold"))))+
geom_bar(aes(fill=factor(Sold,levels=c(0,1))), alpha = 1)+
facet_grid(~ Location)+
scale_fill_brewer(palette="Paired")+
geom_text(stat="prop", aes(label=sprintf("%.1f%%",after_stat((prop)*100))),vjust=1, colour="w
hite")+
scale_x_discrete(name="Sales Status")+
theme(axis.title.y = element_blank(), legend.position = "none")+
  labs(title="Sold or Retained in past 12 months", subtitle = "with a focus on Location")

r5d
ggsave("plots/r5d.png")
```

Exploring Ratings and Categorical variables

Hide

```
#create new variable SoldStatus from Sold where 0 is retained, 1 is sold
rests <- rests %>%
  mutate(SoldStatus = ifelse(Sold == 1, "sold","retained"))
```

Hide

```
#r5e Sold or Retained
# with Mean Ratings across Food, Service & Atmosphere
r5e = ggplot(rests, aes(x=Food, fill=SoldStatus)) +
geom_density(alpha=0.4, show.legend = TRUE) +
labs(x= "Food Rating", fill="Sales Status", title="Sold or Retained in past 12 months", subti
tle = "with a focus on Food Rating")

r5e
ggsave("plots/r5e.png")
```

Hide

```
#r5f Sold or Retained
# with Mean Ratings across Service
r5f = ggplot(rests, aes(x=Service, fill=SoldStatus)) +
geom_density(alpha=0.4, show.legend = TRUE) +
labs(x= "Service Rating", fill="Sales Status", title="Sold or Retained in past 12 months", su
btitle = "with a focus on Service Rating")

r5f
ggsave("plots/r5f.png")
```

Hide

```
#r5g Sold or Retained
# with Mean Ratings across Atmosphere
r5g = ggplot(rests, aes(x=Atmosphere, fill=SoldStatus)) +
geom_density(alpha=0.4, show.legend = TRUE) +
labs(x= "Atmosphere Rating", fill="Sales Status", title="Sold or Retained in past 12 months",
subtitle = "with a focus on Atmosphere Rating")

r5g
ggsave("plots/r5g.png")
```

Hide

```
#r5g Days open
#  across Food
r5g = ggplot(rests, aes(x=Food, fill=factor(Days), group=(Days))) +
geom_density(alpha=0.4, show.legend = TRUE) +
labs(x= "Food Rating", fill="Days Open", title="Food Rating", subtitle = "with a focus on day
s opened")

r5g
ggsave("plots/r5g.png")
```

Hide

```
#r5h Days open
#  across Service
r5h = ggplot(rests, aes(x=Service, fill=factor(Days), group=(Days))) +
geom_density(alpha=0.4, show.legend = TRUE) +
labs(x= "Service Rating", fill="Days Open", title="Food Rating", subtitle = "with a focus on
days opened")

r5h
ggsave("plots/r5h.png")
```

Hide

```
#r5i Days open
#  across Food
r5i = ggplot(rests, aes(x=Atmosphere, fill=factor(Days), group=(Days))) +
geom_density(alpha=0.4, show.legend = TRUE) +
labs(x= "Atmosphere Rating", fill="Days Open", title="Atmosphere Rating", subtitle = "with a
focus on days opened")

r5i
ggsave("plots/r5i.png")
```

Hide

```
#r5j Location across Food Rating
r5j = ggplot(rests, aes(x=Food, fill=Location)) +
geom_density(alpha=0.4, show.legend = TRUE) +
labs(x= "Food Rating", fill="Location", title="Food Rating", subtitle = "with a focus on loca
tion")

r5j
ggsave("plots/r5j.png")
```

Hide

```
#r5k Location across Food Rating
r5k = ggplot(rests, aes(x=Service, fill=Location)) +
geom_density(alpha=0.4, show.legend = TRUE) +
labs(x= "Service Rating", fill="Location", title="Service Rating", subtitle = "with a focus o
n location")

r5k
ggsave("plots/r5k.png")
```

Hide

```
#r5l Location across Atmosphere Rating
r5l = ggplot(rests, aes(x=Atmosphere, fill=Location)) +
geom_density(alpha=0.4, show.legend = TRUE) +
labs(x= "Atmosphere Rating", fill="Location", title="Atmosphere Rating", subtitle = "with a f
ocus on location")

r5l
ggsave("plots/r5l.png")
```

Mutate Ratings into a single variable

Hide

```
# mutation of all ratings into a single variable on a scale of 0-15
rests <- rests %>%
  mutate(Rating = Food + Atmosphere + Service)
# Density Plot of Rating with Mean
mean_rating <- mean(rests$Rating)

#Density Plot of the new Rating variable
r5r = ggplot(rests, aes(x=Rating))+
    geom_density(fill="deepskyblue4", alpha = 0.4) +
    geom_vline(aes(xintercept=mean_rating),linetype="dashed",
                   color="darkorchid4", size=0.9) +
  annotate("label", x=(mean_rating), y=0.3, label= (paste(round(mean_rating,2))),vjust=2)

r5r
ggsave("plots/r5r.png")
```

Hide

```
# bivariate examinations of Rating and Location
#r5m Location across Rating
r5m = ggplot(rests, aes(x=Rating, fill=Location)) +
geom_density(alpha=0.4, show.legend = TRUE) +
labs(x= "Rating", fill="Location", title="Rating", subtitle = "with a focus on location")

r5m
ggsave("plots/r5m.png")
```

Hide

```
# Bivariate examingation of Sold and Rating
#r5n Sold across Rating
r5n = ggplot(rests, aes(x=Rating, fill=SoldStatus)) +
geom_density(alpha=0.4, show.legend = TRUE) +
labs(x= "Rating", fill="Sold Status", title="Rating", subtitle = "with a focus on Sold vs Ret
ained")

r5n
ggsave("plots/r5n.png")
```

Mutate Sunday and Monday into single categorical variable

Hide

```
# Create a new combined variable from Sunday and Monday called SunMon.

rests$SunMon <- with(rests, ifelse(Sunday == "Yes" & Monday == "Yes", "SunAndMon",
                      ifelse(Sunday == "Yes" & Monday == "No", "SunNotMon",
                      ifelse(Sunday == "No" & Monday == "Yes", "MonNotSun", "NotSunMon"))))
```

Hide

```
# Convert SunMon to a factor variable with specified levels
rests$SunMon <- factor(rests$SunMon, levels = c("SunNotMon", "SunAndMon", "MonNotSun", "NotSu
nMon"))
```

Hide

```
# Check the distribution of the new SunMon variable
table(rests$SunMon)
```

```

SunNotMon SunAndMon MonNotSun NotSunMon
      156       148        79       117
```

Mutate Days, Sunday and Monday into single categorical variable

Hide

```
# Create variable called Open

# Create the combined categorical variable based on Days, Sunday, and Monday

rests$Open <- with(rests, ifelse(Days == 7, "open7",
                  ifelse(Days == 6 & Sunday == "Yes" & Monday == "Yes", "open6incSunMon",
                  ifelse(Days == 6 & Sunday == "Yes", "open6incSun",
                  ifelse(Days == 6 & Monday == "Yes", "open6incMon",
                  ifelse(Days == 5 & Sunday == "Yes" & Monday == "Yes", "open5incSunMon",
                  ifelse(Days == 5 & Sunday == "Yes", "open5incSun",
                  ifelse(Days == 5 & Monday == "Yes", "open5incMon",
                  "open5notSunMon")))))))
```

Hide

```
# Convert Open to a factor and specify the levels for clarity

rests$Open <- factor(rests$Open, levels = c("open7", "open6incSun", "open6incMon",
                                            "open6incSunMon", "open5incSun",
                                            "open5incMon", "open5incSunMon",
                                            "open5notSunMon"))
```

Hide

```
# View the distribution of the new categorical variable
table(rests$Open)
```

```

        open7    open6incSun    open6incMon open6incSunMon    open5incSun
          148            156             79              0              0
   open5incMon open5incSunMon open5notSunMon
             0              0            117
```

Logistic Regression - Sunday

Hide

```
# logistic regression Sunday and Customers
# Do the number of customers predict being open on Sundays?

lr.Sun.Cust <- glm(Sunday ~ Customers, data = rests, family = binomial)
summary(lr.Sun.Cust)
```

Hide

```
# Does the average price per customer predict being open on Sundays?

lr.Sun.Price <- glm(Sunday ~ Price, rests, family=binomial)
summary(lr.Sun.Price)
```

Hide

```
# Does Location predict being open on Sundays?
lr.Sun.Location <- glm(Sunday ~ Location, data = rests, family = binomial)
summary(lr.Sun.Location)
```

Hide

```
# Does Rating predict being open on Sundays?

lr.Sun.Rating <- glm(Sunday ~ Rating, data = rests, family = binomial)
summary(lr.Sun.Rating)
```

Create TopRating Variable

Hide

```
# engineer new variable called TopRating.
# TopRating  = (service + food + atmosphere) when > 12
rests <- rests%>%
  mutate(TopRating = ifelse(Rating>= 12, 1, 0))
```

Hide

```
str(rests)
```

Hide

```
# Table of counts for TopRating
table(rests$TopRating)
```

Hide

```
# Plot r10 - Exploring distribution of TopRating
r10 = ggplot(rests, aes(x=TopRating)) +
geom_bar(fill="deepskyblue4") + geom_text(stat="prop", aes(label=sprintf("%.1f%%", after_stat
((prop)*100))),vjust=1, colour="white")+
  theme(axis.title.y = element_blank())
r10
ggsave("plots/r10.png")
```

Class Imbalance Note

Key Points to Consider:

1. **Accuracy Paradox**:
    - High accuracy can be misleading in the presence of class imbalance because the model might simply predict the majority class most of the time.
2. **Evaluation Metrics**:
    - Instead of relying solely on accuracy, consider using other metrics that provide a better picture of model performance for imbalanced data:
        - **Precision**: The proportion of true positives among the predicted positives.
        - **Recall (Sensitivity)**: The proportion of true positives among the actual positives.
        - **F1 Score**: The harmonic mean of precision and recall, providing a balance between the two.
        - **ROC-AUC**: The area under the Receiver Operating Characteristic curve, which evaluates the model's ability to distinguish between classes.
3. **Handling Class Imbalance**:
    - **Resampling Techniques**: Use oversampling (e.g., SMOTE) or undersampling to balance the classes.
    - **Class Weights**: Adjust the weights of the classes in the logistic regression model to give more importance to the minority class.

Logistic Regression with dependent variable as Top Rating

Top Rating ~ Customers

Hide

```
lr_tr_cust <- glm(TopRating ~ Customers, data = rests, family = binomial)
summary(lr_tr_cust)
```

Hide

```
# Coefficient log-odds value
log_odds <- -0.019026

# Convert log-odds to probability
probability <- plogis(log_odds)
print(probability)
```

Hide

```
# Extract the intercept
intercept_log_odds <- -0.271767

# Convert log-odds to probability
baseline_probability <- plogis(intercept_log_odds)
print(baseline_probability)
```

Hide

```
# Define the baseline probability when there are 0 customers
baseline_probability <- 0.4479329

# Convert the baseline probability to log-odds
baseline_log_odds <- qlogis(baseline_probability)

# Coefficient for Customers
coef_customers <- -0.019

# Define a range of customer values
customers <- seq(0, 250, by = 1)

# Calculate the corresponding probabilities
probabilities <- plogis(baseline_log_odds + coef_customers * customers)

# Create a data frame for plotting
data <- data.frame(customers, probabilities)

# Plot the relationship
lrm1= ggplot(data, aes(customers, probabilities)) +
    geom_line(color = "blue")  +
    geom_point(aes(x=0, y=baseline_probability), color="red", size=3) +
    annotate("text", x = 10, y = baseline_probability,
             label = round(baseline_probability, 3), vjust = 0, color = "red") +
    labs(title ="Probability of Top Rating vs. Number of Customers",
         x = "Number of Customers", y = "Probability of Top Rating") +
  theme_stata()
lrm1
ggsave("plots/lrm1.png")
```

Intercept not significant at -0.27. This means that when the number of customers is zero, the log-odds of having a top rating is not not significantly different from zero.

The significant but negative coefficient for Customers at -0.019 indicates that for each additional customer, the log-odds of having a top rating decrease by approximately 0.019. So as customer numbers increase, there is a 49.5% probability of not receiving a top rating

Top Rating ~ Price

Hide

```
lr_tr_price <- glm(TopRating ~ Price, data = rests, family = binomial)
summary(lr_tr_price)
```

Hide

```
# Define the function
plot_logistic_regression <- function(intercept, coefficient, range = 0:170) {
  # Convert the intercept to baseline log-odds
  baseline_log_odds <- -3.189044

  # Calculate the probabilities for the given range
  probabilities <- plogis(baseline_log_odds + 0.024538 * range)

  # Create a data frame for plotting
  data <- data.frame(predictor = range, probabilities = probabilities)

  # Create the plot using ggplot2
  lrm2 <- ggplot(data, aes(x = predictor, y = probabilities)) +
    geom_line(color = "blue") +
    geom_point(aes(x = 0, y = plogis(baseline_log_odds)), color= "red", size =3) +
    annotate("text", x = 10, y = plogis(baseline_log_odds),
             label = round(plogis(baseline_log_odds), 3), vjust=0, color= "red") +
    labs(title = "Probability of Top Rating vs. Average Price per Customer",
         x = "Average Price per Customer",
         y = "Probability of Top Rating") +
      theme_stata()

  # Return the plot
  return(lrm2)
}

# Generate the plot
lrm2 <- plot_logistic_regression(intercept, positive_coefficient)
ggsave("plots/lrm2.png")
lrm2
```

Intercept is significant at -3.18. This means that when the price is zero, the log-odds of having a top rating is significantly different from zero.

The significant and positive coefficient for Price at 0.024 indicates that for each additional dollar in price, the log-odds of having a top rating increases by approximately 0.024. So as price increases, the likelihood of having a top rating also increases.

Hide

```
# Does Location predict a TopRating?
# The intercept will be City
lr_tr_location <- glm(TopRating ~ Location, data = rests, family = binomial)
summary(lr_tr_location)
```

Hide

```
# Will a specific menu cuisine predict a TopRating?
lr_tr_menu <- glm(TopRating ~ Menu, data = rests, family = binomial)
summary(lr_tr_menu)
```

Very little of significance within the impact of menu on a TopRating.

Hide

```
# Does opening Sunday predict a TopRating?

lr_tr_Sunday <- glm(TopRating ~ Sunday, data = rests, family = binomial)
summary(lr_tr_Sunday)

#While Yes is significant at .06, No (intercept) is signficant AND negative
```

Hide

```
lr_tr_Monday <- glm(TopRating ~ Monday, data = rests, family = binomial)
summary(lr_tr_Monday)
```

Both Sunday and Monday have significant and negative intercepts, indicating that being closed on those days reduces the log-odds of a top rating by -2.175 for Sunday and -1.76 for Monday.

Hide

```
lr_tr_Days <- glm(TopRating ~ Days, data = rests, family = binomial)
summary(lr_tr_Days)
```

The coefficient for days at 0.1518, is not statistically significant (p-value = 0.3991). This indicates that the number of days a restaurant is open does not have a significant effect on the likelihood of receiving a top rating. The log-odds of having a top rating increase by 0.1518 for each additional day the restaurant is open, but this effect is not statistically significant. Being open for 0 days (hypothetically) reduces the log-odds of having a top rating by a significant -2.76.

Hide

```
lr_tr_Distance <- glm(TopRating ~ Distance, data = rests, family = binomial)
summary(lr_tr_Distance)
```

When the distance between restaurants is zero, then the log-odds of having a top rating significantly decreases by -1.81.

Hide

```
lr_tr_Sold <- glm(TopRating ~ Sold, data = rests, family = binomial)
summary(lr_tr_Sold)
```

Intercept of -1.495 and significant. This is the log-odds of a restaurant receiving a top rating when it has not been sold in the past 12 months. The negative value indicates that the log-odds are less than zero, meaning the probability of receiving a top rating is less than 50% when the restaurant has not been sold. Given the nature of Top rating, this makes sense.

The coefficient indicates that the log-odds of receiving a top rating decrease by 0.9646 when the restaurant has been sold in the past 12 months compared to when it has not been sold. This is also a negative value, meaning that being sold *further* decreases the likelihood of receiving a top rating.

Example Calculation:

To make it clearer, let's calculate the probabilities:

- **Not Sold**:
  - Log-odds: -1.4955
  - Odds: (e^{-1.4955} \approx 0.224)
  - Probability: (\frac{0.224}{1 + 0.224} \approx 0.183) (18.3%)
- **Sold**:
  - Log-odds: -1.4955 - 0.9646 = -2.4601
  - Odds: (e^{-2.4601} \approx 0.085)
  - Probability: (\frac{0.085}{1 + 0.085} \approx 0.078) (7.8%)

So, the probability of receiving a top rating is lower for restaurants that have been sold compared to those that have not been sold.

Multiple Logistic Regression

Stepwise Logistic Regression with Top Rating as Dependent Variable

Hide

```
# New Subset called rest6

rest6 <- subset(rests, select= c(TopRating, Price, Customers, Location, Menu, Days, Distance,
Sold))
```

Hide

```
str(rest6)
```

Hide

```
# Build a null model for TopRating

TopRating.null=glm(TopRating ~ 1, rest6, family=binomial)
```

Hide

```
# Build a full model for TopRating

TopRating.full=glm(TopRating ~ ., rest6, family=binomial)
```

Hide

```
# Stepwise GLM for TopRating

glm_model6 <- stepAIC(TopRating.null, scope= list(lower= TopRating.null, upper=TopRating.ful
l), direction = "both", family=binomial)
```

Hide

```
summary(glm_model6)
```

Hide

```


# Fit the multiple logistic regression model
model_TopRating <- glm(TopRating ~ Price + Sold, rest6, family = binomial)

# Create a data frame for predictions
price_range <- seq(min(rest6$Price), max(rest6$Price), length.out = 100)
constant_sold <- 0

prediction_data <- data.frame(Price = price_range, Sold = constant_sold)

# Predict probabilities
prediction_data$probability <- predict(model_TopRating, newdata = prediction_data, type = "re
sponse")

# Plot the results using ggplot2
ggplot(prediction_data, aes(x = Price, y = probability)) +
  geom_line(color = "blue") +
  labs(title = "Probability of Top Rating vs. Average Price per Customer",
       subtitle = "With restaurant unsold in past 12 months",
       x = "Average Price per Customer",
       y = "Probability of Top Rating")
```

Hide

```
# Example Price values ~ 0, Q1, mean, Q3
price_values <- c(0, 21, 45, 60)

# Calculate log-odds for each Price value
log_odds_price <- -2.862715 + 0.024404 * price_values

# Convert log-odds to probabilities
probabilities_price <- plogis(log_odds_price)
print(probabilities_price)
```

Interpretation:

- **Baseline Probability**: When Price and Sold are both zero, the probability of receiving a top rating is approximately 5.41%.
- **Effect of Price:** As Price increases, and the business is retained, the probability of receiving a top rating increases.
- **Effect of Sold:** If the restaurant has been sold in the past 12 months (Sold = 1), the probability of receiving a top rating decreases compared to when it hasn't been sold (Sold = 0).

Confusion Matrix for TopRating

Hide

```
# Fit the logistic regression model
model_TopRating <- glm(TopRating ~ Price + Sold, data = rest6, family = binomial)

# Make predictions
predicted_probabilities <- predict(model_TopRating, type = "response")
predicted_classes <- ifelse(predicted_probabilities > 0.5, 1, 0)

# Create the confusion matrix
confusion_matrix <- confusionMatrix(as.factor(predicted_classes), as.factor(rest6$TopRating))
print(confusion_matrix)
```

Let's break down the confusion matrix and the associated statistics to understand the performance of your logistic regression model:

Confusion Matrix

- **True Negatives (TN)**: 425
- **False Positives (FP)**: 6
- **False Negatives (FN)**: 57
- **True Positives (TP)**: 12

Accuracy: 0.874 This means that 87.4% of the predictions made by the model are correct.

```
-   95% Confidence Interval: (0.8417, 0.9018)
```

- **Sensitivity (Recall)**: 0.9861 This means that the model correctly identifies 98.61% of the actual positives (class 0).
- **Specificity**: 0.1739 This means that the model correctly identifies 17.39% of the actual negatives (class 1). ### Predictive Values
- **Positive Predictive Value (Precision)**: 0.8817 This means that 88.17% of the predicted positives (class 0) are actually positive.
- **Negative Predictive Value**: 0.6667 This means that 66.67% of the predicted negatives (class 1) are actually negative.

Prevalence and Detection Rates

- **Prevalence**: 0.8620 This is the proportion of actual positives (class 0) in the dataset.
- **Detection Rate**: 0.8500 This is the proportion of actual positives correctly identified by the model.
- **Detection Prevalence**: 0.9640 This is the proportion of predicted positives in the dataset.

Balanced Accuracy

- **Balanced Accuracy**: 0.5800 This is the average of sensitivity and specificity, providing a balanced measure of model performance.

Interpretation

- **High Sensitivity**: The model is very good at identifying actual positives (class 0), but…

- **Low Specificity**: The model struggles to correctly identify actual negatives (class 1).
- **Accuracy**: The overall accuracy is high, but given the imbalance in sensitivity and specificity, this might be misleading.
- **Kappa and Mcnemar's Test**: Indicate that while the model performs better than random chance, there is room for improvement, especially in correctly identifying negatives.

Hide

```
# Fit the logistic regression model
model_TopRating <- glm(TopRating ~ Price + Sold, data = rest6, family = binomial)

# Make predictions
predicted_probabilities <- predict(model_TopRating, type = "response")

# Calculate ROC curve
roc_curve <- roc(rest6$TopRating, predicted_probabilities)

# Plot ROC curve
ggroc(roc_curve) +
  ggtitle("ROC Curve for Logistic Regression Model") +
  xlab("False Positive Rate") +
  ylab("True Positive Rate")

# Calculate AUC
auc_value <- auc(roc_curve)
print(auc_value)
```

Weighting the model with Class Weights

Explanation:

- **Class Weights Calculation**: The ifelse function assigns a weight of 0.86 to the minority class (TopRating = 1) and 0.14 to the majority class (TopRating = 0). Adjust these weights based on the actual class distribution in your dataset.
- **Stepwise Model Selection**: The stepAIC function performs stepwise selection to find the best model based on AIC, incorporating the class weights to handle the imbalance.

Practical Considerations:

- **Choosing Weights**: The weights should ideally reflect the inverse of the class frequencies. In this case, with 86% zeros and 14% ones, the weights are set accordingly.
- **Model Evaluation**: After fitting the model, evaluate its performance using metrics like precision, recall, F1 score, and ROC-AUC to ensure it performs well on the minority class.

Hide

```
# Weighting the Model to mitigate class imbalance
# Calculate class weights
class_weights <- ifelse(rest6$TopRating == 1, 6, 1)
```

Hide

```
#Create null model with class weights to use in stepwise
fit.null <- glm(TopRating ~ 1, family = "binomial", data = rest6,
                weights = class_weights)
```

Hide

```
# Create full model with class weights to use in stepwise
fit.full <- glm(TopRating ~ Price + Sold + Customers + Days + Distance +
                  Location,
                 data = rest6,
                family = binomial,
                weights = class_weights)
```

Hide

```
# Perform stepwise selection with the weighted null and weighted full model.

fit.final <- step(fit.null, formula(fit.full), direction = "both", family="binomial")

# Display the summary of the selected model
summary(fit.final)

# Distance is only significant at p=0.0943
# Distance only improves the model AIC by a small fraction
```

Hide

```
# Running the model without distance

fit.final1 <- glm(TopRating ~ Price + Sold,
                data = rest6,
                family = binomial,
                weights = class_weights)
summary(fit.final1)
```

Hide

```


# Make predictions
predicted_probabilities_w <- predict(fit.final1, type = "response")
predicted_classes <- ifelse(predicted_probabilities_w > 0.5, 1, 0)

# Create the confusion matrix
confusion_matrix <- confusionMatrix(as.factor(predicted_classes), as.factor(rest6$TopRating))
print(confusion_matrix)

# Note that while model isn't as accurate, the balanced accuracy has improved
# The specificity has also dramatically improved
```

Summary on Weighted Results for TopRating Logistic Regression

1. **Class Imbalance Handling**:

- The weighted model improves the **specificity** (recall for class 1) significantly (from 0.1739 to 0.6812), indicating better performance in predicting the minority class (TopRating = 1).
- The **balanced accuracy** is higher in the weighted model (0.7257 vs. 0.5800), suggesting a more balanced performance across both classes.

2. **Overall Accuracy**:
   - The unweighted model has higher overall accuracy (0.874 vs. 0.758), but this is largely driven by the majority class (TopRating = 0).
3. **Precision and Recall**:
   - The weighted model has a higher **positive predictive value (PPV)** (0.9379 vs. 0.8817), meaning it is more reliable when it predicts the majority class (TopRating = 0), though a lower **negative predictive value (NPV)** (0.3219 vs. 0.6667), indicates it is less reliable when predicting the minority class (TopRating = 1).

As the goal is to improve the detection of the minority class (TopRating = 1), the weighted model is more appropriate because it significantly improves specificity and balanced accuracy.

Given the class imbalance and the importance of correctly identifying the minority class, the weighted model seems more valid and appropriate for this scenario. It provides a better balance between sensitivity and specificity, which is crucial when dealing with imbalanced data sets.

Cross-checking with the olsrr all possible subsets approach

Hide

```
# New Subset called rest7

rest7 <- subset(rests, select= c(TopRating, Price, Customers, Location, Menu, Sunday, Monday,
Distance, Sold))
```

Hide

```
# Build a null model for TopRating

TopRating7.null=glm(TopRating ~ 1, rest7, family=binomial, weights = class_weights)
```

Hide

```
# Build a full model for TopRating

TopRating7.full=glm(TopRating ~ ., rest7, family=binomial, weights = class_weights)
```

Hide

```
# Stepwise GLM in olsrr for TopRating

allposs_models <- ols_step_all_possible(lm(TopRating7.full, rest7))
allposs_models
```

Hide

```
#troubleshooting the stepallmodels results - added $result into the string
# Extract AIC values from the result
aic_values <- allposs_models$result$aic

# Find the minimum AIC value
min_aic <- min(aic_values)

# Find the model with the minimum AIC
best_model_index <- which.min(aic_values)
best_model <- allposs_models$result$models[[best_model_index]]

# Print the minimum AIC value and the corresponding model
print(paste("Minimum AIC:", min_aic))
print(best_model)

# while $result worked for AIC. $models is not working for the model itself.
```

Hide

```
# troubleshooting the stepallmodels results - added $result into the string
# finding the index number of best AIC

allposs_models$result$mindex[allposs_models$result$aic == min(allposs_models$result$aic)]
```

Hide

```
# finding the predictors for the best AIC
allposs_models$result$predictors[allposs_models$result$aic == min(allposs_models$result$aic)]
```

Hide

```
# still troubleshooting to find full result for best model

# Check the structure of the result object
str(allposs_models$result)

# Extract AIC values from the result
aic_values <- allposs_models$result$aic

# Find the model with the minimum AIC
best_model_index <- which.min(aic_values)

# Extract the best model
best_model <- allposs_models$result[best_model_index, ]

# Print the minimum AIC value and the corresponding model
print(paste("Minimum AIC:", min_aic))
print(best_model)
```

Hide

```
summary(allposs_models)
```

Looking at skew in continuous variables

Customers

Hide

```
# Density Plot of Customers with Mean
mean_customers <- mean(rests$Customers)

#Density Plot
r6 = ggplot(rests, aes(x=Customers))+
    geom_density(fill="deepskyblue4", alpha = 0.4) +
    geom_vline(aes(xintercept=mean_customers),linetype="dashed",
                   color="darkorchid4", size=0.9) +
  annotate("label", x=(mean_customers), y=0.008,
           label= (paste(round(mean_customers,2))),vjust=2)
r6
ggsave("plots/r6.png")
```

Density Plot of Price with Mean

Hide

```
# Density Plot of Price with Mean
mean_price <- mean(rests$Price)

#Density Plot
r7 = ggplot(rests, aes(x=Price))+
    geom_density(fill="deepskyblue4", alpha = 0.4) +
    geom_vline(aes(xintercept=mean_price),linetype="dashed",
                   color="darkorchid4", size=0.9) +
  annotate("label", x=(mean_price), y=0.02, label= (paste(round(mean_price,2))),vjust=2)
ggsave("plots/r7.png")
r7
```

Exploring Transformation of continuous variables

With the intention to perform parametric tests, the results will be more valid if the distribution of the dependent variable approximates a normal distribution and the homoscedasticity assumption is met. Depending upon the degree of skewness and whether the direction of skewness is positive or negative, a different approach to transformation is often required. Uni-modal distributions can be roughly classified into the following transformation categories:

Skew	Moderate	High	(Higher)	Extreme
Positive (right tail)	Square root transformation	Natural log transformation	Log base 10 transformation	Inverse transformation
Negative (left tail)	Reflect then square root transformation	Reflect then natural log transformation	Reflect then log base 10 transformation	Reflect then inverse transformation

Hide

```
# Exploring Transformation of continuous variables

# Working through the positive right tail skew approaches
# applying Square root transformation to Price
rests$Price_sqrt <- sqrt(rests$Price)
```

Hide

```
# Density Plot of Square Root Transformation of Price with Mean
mean_price_sqrt <- mean(rests$Price_sqrt)

#Density Plot
r7a = ggplot(rests, aes(x=Price_sqrt))+
    geom_density(fill="deepskyblue4", alpha = 0.4) +
    geom_vline(aes(xintercept=mean_price_sqrt),linetype="dashed",
                   color="darkorchid4", size=0.9) +
  annotate("label", x=(mean_price_sqrt), y=0.2, label= (paste(round(mean_price_sqrt,2))),vjus
t=2)
r7a
ggsave("plots/r7a.png")
```

Hide

```
# applying Log transformation to Price
rests$Price_log <- log(rests$Price)
```

Hide

```
# Density Plot of Price with Mean
mean_price_log <- mean(rests$Price_log)

#Density Plot
r7b = ggplot(rests, aes(x=Price_log))+
    geom_density(fill="deepskyblue4", alpha = 0.4) +
    geom_vline(aes(xintercept=mean_price_log),linetype="dashed",
                   color="darkorchid4", size=0.9) +
  annotate("label", x=(mean_price_log), y=0.5, label= (paste(round(mean_price_log,2))),vjust=
2)
r7b
ggsave("plots/r7b.png")
```

Hide

```
# Kolmogorov-Smirnov test for original data
ks.test(rests$Price, "pnorm", mean = mean(rests$Price), sd = sd(rests$Price))

# Kolmogorov-Smirnov test for transformed data
ks.test(rests$Price_log, "pnorm", mean = mean(rests$Price_log), sd = sd(rests$Price_log))

# Still siginificantly skewed
```

Q-Q Plots of original, sqrt and log transformations of Price

Hide

```
# Q-Q Plot of original distribution of Price
r7c = ggplot(rests, aes(sample=Price)) + stat_qq() + stat_qq_line()
r7c
ggsave("plots/r7c.png")
```

Hide

```
#
r7d = ggplot(rests, aes(sample=Price_sqrt)) + stat_qq() + stat_qq_line()
r7d
ggsave("plots/r7d.png")
```

Binning Price into Categories

Hide

```
# Binning Price into Categories - Terciles
# Using the cut() function with quantile breaks

rests$Price_Binned <- cut(rests$Price,
                      breaks = quantile(rests$Price, probs =seq(0, 1,length.out=4),
                      na.rm = TRUE),
                      include.lowest = TRUE,
                      labels = c("Low", "Mid", "High"))

# View the binned data
str(rests)
```

Hide

```
# Rename hi_price_bin
rests <- rests %>%
  dplyr::rename(HighEndPrice = hi_price_bin)
str(rests)
```

Hide

```
str(rests)
```

Hide

```
# Create a ggplot of Price grouped by Price_Binned Categories

r7e=ggplot(rests, aes(x = Price)) +
  geom_density(aes(x = Price, fill = Price_Binned),
            position = "identity", alpha = 0.9, , binwidth = 1) +
  scale_x_continuous(breaks=seq(0, max(rests$Price), by=25)) +
  labs(title = "Density Plot of Price grouped by Pricing Categories",
       x = "$Price per Customer",
       y = "Density")
ggsave("plots/r7e.png")
r7e
```

Creating a High-End Price Variable

Hide

```
# Creating a High-End Price Variable
rests$HighEndPrice <- ifelse(rests$Price_Binned == "High", 1, 0)
```

Hide

```
summary(rests$HighEndPrice)
```

Hide

```
# Plotting number of customers by High Price Bin with Mean

# Calculate the mean Number of Customers where HighEndPrice = 1
mean_high_price <- rests %>%
  dplyr::filter(HighEndPrice == 1) %>%
  summarize(mean_customers = mean(Customers, na.rm = TRUE)) %>%
  pull(mean_customers)

# Plot the data with Mean annotated on vertical line
r7f=ggplot(rests, aes(x = Customers, fill = as.factor(HighEndPrice))) +
  geom_density(alpha = .6) +
  scale_fill_manual(values = c("1" = "red", "0" = "skyblue3"))+
  labs(title = "Customers Per Day categorized by High-End Price",
       x = "Number of Customers",
       fill = "1 = High End Price") +
  geom_vline(aes(xintercept = mean_high_price), color = "black",
             linetype = "dashed", size = 1) +
  annotate("text", x = mean_high_price,
           y = 0.01, label = paste("Mean =", round(mean_high_price, 2)),
           color = "black", angle = 90, vjust = -1)

r7f
ggsave("plots/r7f.png")
```

High variance and U shape peaks of Customers

This is an interesting distribution. It is “tall” or “peaked” but the peak has has a U shape. The number of customers reduces after a certain $ amount and then increases again before a higher peak. Capturing this U-shape may require a quadratic term to be added to the predictors.

To help determine best GLM family

Hide

```
# To help determine best GLM family for high peaked data
# Calculate mean and variance of the count data
mean_customers <- mean(rests$Customers, na.rm = TRUE)
var_customers <- var(rests$Customers, na.rm = TRUE)

mean_customers
var_customers

# Where variance ≈ mean: Poisson distribution might be suitable
# Where variance > mean: Either negative binomial or quasi-Poisson families
```

Hide

```
# Box Plot of customers by Price Categories
r7g=ggplot(rests, aes(x = as.factor(Price_Binned) , y = Customers)) +
  geom_boxplot(fill = "DeepSkyBlue4", alpha = 0.7) +
  labs(title = "Customers per Day across Price Categories",
       x = "Price Category",
       y = "Number of Customers Per Day")
ggsave("plots/r7g.png")
r7g
```

Hide

```
# Violin Plot of customers by Price Categories
r7g=ggplot(rests, aes(x = as.factor(Price_Binned) , y = Customers,
                      fill = as.factor(Price_Binned))) +
  geom_violin(alpha = 0.7) +
  theme(legend.position = "bottom")+
  labs(title = "Customers per Day across Price Categories",
       x = "Price Category",
       y = "Number of Customers Per Day",
       fill = "Price Categories")
r7g + geom_boxplot(width = 0.1, color = "black", alpha = 0.5,
                   position = position_dodge(width = 0.75))

ggsave("plots/r7g.png")
r7g
```

Hide

```
# scatter plot of customers and high-end price

r7h=ggplot(rests, aes(x = Price, y = Customers, color = as.factor(HighEndPrice))) +
  geom_point() +
  scale_x_continuous(breaks = seq(0, max(rests$Price, na.rm = TRUE), by = 25)) +
  scale_color_manual(values = c("0" = "skyblue4", "1" = "red"),
                     labels = c("0", "1")) +
  labs(title = "High-End Price vs Customers per Day",
       x = "$Price per Customer",
       y = "Number of Customers",
       color = "1 = High End Price")
ggsave("plots/r7h.png")
r7h
```

Hide

```
# scatter plot of customers and high-end price

r7i=ggplot(rests, aes(x = Price, y = Customers, color = Price_Binned)) +
  geom_point() +
  scale_x_continuous(breaks = seq(0, max(rests$Price, na.rm = TRUE), by = 25)) +
  labs(title = "Scatter Plot of Price Categories vs Customer Numbers",
       x = "$Price per Customer",
       y = "Number of Customers",
       color = "High End Price (0 = No, 1 = Yes)")
ggsave("plots/r7i.png")
r7i
```

Stepwise Log Linear Regression using High-End Price to create the subset

Hide

```
# Create a subset of the data where HighEndPrice = 1

high_price_subset <- subset(rests, HighEndPrice == 1)

# View the first few rows of the subset
head(high_price_subset)
```

If variance ≈ mean: The Poisson distribution might be suitable. If variance > mean: Consider using the negative binomial or quasi-Poisson families.

Hide

```
# Create a null model first
HighEnd.null <- lm(log(Customers) ~ 1, data=high_price_subset)

# AIC measures for the null model
AIC(HighEnd.null)
extractAIC(HighEnd.null)
```

Hide

```


# Create a full model next
HighEnd.full <- lm(log(Customers) ~ Price + Location + Service + Atmosphere + Food + Days + M
enu + Distance + Sunday + Days*Sunday + Atmosphere*Food, data=high_price_subset)
summary(HighEnd.full)

# AIC measures for the full model
AIC(HighEnd.full)
extractAIC(HighEnd.full)
```

Hide

```
# Stepwise model
HighEnd.fit <- stepAIC(HighEnd.null, direction = "both", list(lower=HighEnd.null, upper=HighE
nd.full))
```

Hide

```
summary(HighEnd.fit)
```

- Price: exp(-0.0111)-1=0.011 -> 1.11% This means for each dollar increase in price, the expected number of customrs decreases by 1.11%, all else constant.
- Days: exp(0.0439)-1 = 0.0449 -> 4.49% Each additional day increases customers numbers by aprox 4.49%, all else constant.p=0.002
- Sunday: When Yes, exp(0.0428)-1=0.0437 -> 4.37% increase in customers when open for Sundays, all else constant. p=0.05
- Atmosphere: exp(0,0208)-1=0.021 -> 2.1% increase in customers for each rating point increase, all else constant. p=0.09

- The minimum and maximum residuals indicate that there is some variation between the observed and predicted values, but the residuals are relatively small, given the scale of the log-transformed response variable. The median residual is close to zero, indicating that the model predictions are not systematically biased.

Check residuals

Hide

```
# plotting fitted vs residuals
# Create a data frame with fitted values and residuals
residual_data_highend <- data.frame(
  Fitted = fitted(HighEnd.full),
  Residuals = residuals(HighEnd.full))

# Create residuals vs. fitted values plot using ggplot2
ggplot(residual_data_highend, aes(x = Fitted, y = Residuals)) +
  geom_point(color = "deepskyblue4", alpha = 0.6) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red", size = 1) +  # Reference lin
e at zero
  labs(title = "Residuals vs. Fitted Values",
       x = "Fitted Values",
       y = "Residuals") +
   theme(plot.title = element_text(hjust = 0.5))  # Center the plot title
```

Hide

```
# Create a Q-Q plot using stat_qq() and stat_qq_line()
ggplot(data = data.frame(residuals = residuals(HighEnd.full)), aes(sample = residuals)) +
  stat_qq(color = "deepskyblue4", alpha = 0.6) +  # Q-Q plot points
  stat_qq_line(color = "red", linetype = "dashed", size = 1) +  # Reference line
  labs(title = "Q-Q Plot of Residuals",
       x = "Theoretical Quantiles",
       y = "Sample Quantiles") +
  theme(plot.title = element_text(hjust = 0.5))
```

Logistic Reg- High-End Price ~ Top Rating

Hide

```


Price.bin1 <- glm(HighEndPrice ~ TopRating,
                  data = rests,family = binomial)
summary(Price.bin1)
```

Hide

```
predictions <- predict(Price.bin1, type = "response")
predicted_classes <- ifelse(predictions > 0.5, 1, 0)
actual_classes <- rests$HighEndPrice
confusion_matrix <- confusionMatrix(as.factor(predicted_classes), as.factor(actual_classes))
print(confusion_matrix)
```

Trying this with weights to see if it improves balanced accuracy and specificity

Hide

```
hi_class_weights <- ifelse(rests$HighEndPrice == 1, 3, 1)
```

Hide

```
Price.bin.w <- glm(HighEndPrice ~ TopRating, data=rests, family=binomial, weights = hi_class_
weights)
```

Hide

```
summary(Price.bin.w)
```

Hide

```
# Confusion Matrix
predictions1 <- predict(Price.bin.w, type = "response")
# Ensure factor levels are consistent
predicted_classes1 <- ifelse(predictions1 > 0.5,1,0)
actual_classes1 <- rests$HighEndPrice
confusion_matrix1 <- confusionMatrix(as.factor(predicted_classes1),
                                     as.factor(actual_classes1))
print(confusion_matrix1)
```

Hide

```
str(rests)
```

Hide

```
# Weighting the Model to mitigate class imbalance
# Calculate class weights
price_class_weights <- ifelse(rests$HighEndPrice == 1, 3, 1)
```

Hide

```
# Creating a null set for High-End Price
hiprice.null=glm(HighEndPrice~1, rests, family=binomial,
                 weights= price_class_weights)
```

Hide

```
# Creating a full set for High-End Price
hiprice.full=glm(HighEndPrice~ Locate + Days+ Food+
                Atmosphere + Service + Sold + Menu,
                rests, family=binomial)
```

Hide

```
# Check correlation matrix
cor(rests[, c("Locate", "Days", "Food", "Atmosphere", "Service", "Sold", "Customers")])

# Check variance inflation factor (VIF)
library(car)
vif(hiprice.full)
```

Hide

```
# Fit Firth's logistic regression
firth_model <- logistf(HighEndPrice ~ Customers, data = rests)
summary(firth_model)
```

Hide

```
# Perform stepwise selection with the null and full model.

fit.hiprice <- step(hiprice.null, formula(hiprice.full), direction = "both", family="binomia
l")

# Display the summary of the selected model
summary(fit.hiprice)
```

Hide

```
# Confusion Matricx
predictions <- predict(fit.hiprice, type = "response")
predicted_classes <- ifelse(predictions > 0.5, 1, 0)
actual_classes <- rests$HighEndPrice
confusion_matrix <- confusionMatrix(as.factor(predicted_classes), as.factor(actual_classes))
print(confusion_matrix)
```

Since the intercept is highly negative, this indicates that when all predictors are at their baseline value, the probability of having a high price is extremely low.

- Food rating: odds ratio: exp(1.1916) =3.29 This means that for each additional unit increas e the food rating, the odds of having a high price are 3.29 times higher, assuming all other variables are constant.
- Serivce rating: odds ration:exp(0.7179) = 2.05 This means that for each additional unit increase the service rating, the odds of having a high price are 2.05 times higher, assuming all other variables are constant.
- Atmosphere: odds ratio: exp(0.6994) = 2.01. This means that for each additional unit increase in the atmosphere rating, the odds of having a high price are 2.01 times higher, assuming all other variables are constant.
- Locate: odds ration: exp(0.3856)=1.47. The p-value of 0.096 suggests that this effect is only marginally significant, meaning it has some influence but is not as strong as other predictors. However being in the city increases the odds of having a high-end price by 47%.

Linear Regression and Normality Assumptions

Multiple Regression doesn't assume that your variables are normally distributed, **only that your model residuals** are. ***This is why we need to plot the residuals…***

The normality assumption for linear regression applies to the errors, not the outcome variable per se (and most certainly not to the explanatory variables). The usual statement is that the errors are i.i.d. (i.e., independently and identically distributed). Independence and homoscedasticity are more important assumptions than normality.

"The t-test and least-squares linear regression do not require any assumption of Normal distribution in sufficiently large samples. Previous simulations studies show that "sufficiently large" is often under 100, and even for our extremely non Normal medical cost data it is less than 500" (Lumley etal, 2002:166).

Linear Regression

Stepwise Regression

Hide

```
#create a modified data frame called rest1 with selected and mutated variables
#don't include Monday as it is not independent of Days and Sunday
#specify dyplr::select because of error

rest1 <- rests%>%
  dplyr::select(Price, Customers, Rating, Location, Menu, Days, Distance, Sold, Sunday, Servi
ce, Atmosphere, Food, TopRating)
str(rest1)
```

Hide

```
#need to change Location and Menu to factors

rest1$Menu <- factor(rest1$Menu)
rest1$Location <- factor(rest1$Location)
```

Hide

```
# change location and menu to factors for rests too
rest1$Menu <- factor(rest1$Menu)
rest1$Location <- factor(rest1$Location)
```

Hide

```
#determing the levels for Menu as the first will be the reference as a factor
levels(rest1$Menu)
```

Hide

```
#change this to "Other" since our client's restaurant falls in that category
rest1$Menu <- relevel(rest1$Menu, ref = "Other")
levels(rest1$Menu)
```

Hide

```
# Estimate the Null Model first
customers.null=lm(Customers~1, rest1)
```

Hide

```
# AIC measures for the null model
AIC(customers.null)
extractAIC(customers.null)
```

Hide

```
# Buid the full model
# Include quadratic term for Price and Interaction Term for Days~Sunday
customers.full=lm(Customers ~ Price + exp(Price) + Location + Menu + Days + Distance + Sold +
Sunday + Service + Atmosphere + Food + Days*Sunday, rest1)
summary(customers.full)
```

Hide

```
# AIC measures for the full model
AIC(customers.full)
extractAIC(customers.full)
```

Hide

```
# Stepwise consideration of the model
step(customers.null, scope = formula(customers.full), direction="forward")
```

Hide

```
# now conducting it with MASS stepAIC in both directions to cross-check

stepwise_lmodel <- stepAIC(customers.null, direction = "both", list(lower=customers.null, upp
er=customers.full))
```

Hide

```
summary(stepwise_lmodel)
```

Interpreting these results: - p-Value: the p-value for the F-statistic is very small and <0.05 indicating a statistically significant model.

- **Model Fit**: The adjusted R-squared indicates that approximately 89.37% of the variance in Customers is explained by the predictors Price, Days, Atmosphere, Sunday and Food.
- **Estimate**: The estimated effect of each predictor on the response variable.
- **Intercept**: Represents the expected value of Customers when all predictors are zero.
- **Linear term for Price**: A negative linear relationship with Customers, indicating that *initially* higher prices are associated with fewer customers.For each unit increase in Price the number of customers decreases by 3.72, holding all other variables constant.
- **Quadratic term for Price^2**: This positive quadratic term suggests a U-Shaped relationship between Price and Customers.While customers initially decrease as Price increases (due to the negative linear coefficient), at higher levels of Price, the number of customers starts increaseing again.

- **Turning Point for Price**: To find the price point where the relationship changes direction (minimum point), set the derivative of Customers with respect to Price equal to zero. d(customers)/d(Price) = B1 + 2B2*Price = 0 So (-3.72+2.01676*Price = 0) ~ Turning Point is at Price = 111 This means that the minimum number of customers occurs when Price is around $111 and then slightly increases again. This is typical of a prestige pricing curve.
- **Days**: A positive relationship with Customers, indicating that more days open is associated with more customers.For each unit increase in Days, Customers increase by 4.93 units, holding all else constant.
- **Atmosphere** has a small negative effect, indicating that higher atmosphere scores are associated with fewer customers.For each point rating increase, the number of customers decreases by about 2.953. Captures a quality that is not positively associated with customer volume.
- **Sunday** Being open Sundays increases customers by 4.75.
- **Food** A one-point increase in the Food score is associated with an increase of 2.63 customers. Note that this one is only marginally significant.
- **Residuals**: The summary stats of the residuals. Ideally these should be symmetrically distributed around zero with the median close to zero. Woohoo!! These are now that we've added and captured a quadratic term for price.
- **Residual Standard Error**: Measures the average amount that the response variable deviates from the fitted values. Here, it's 17.79.Lower is better!
- **F-Statistic**: The F-statistic tests whether at least one of the coefficients is significantly different from zero. A high F-statistic of 700.2 with a very small p-value (< 2.2e-16) indicates that the model is statistically significant.

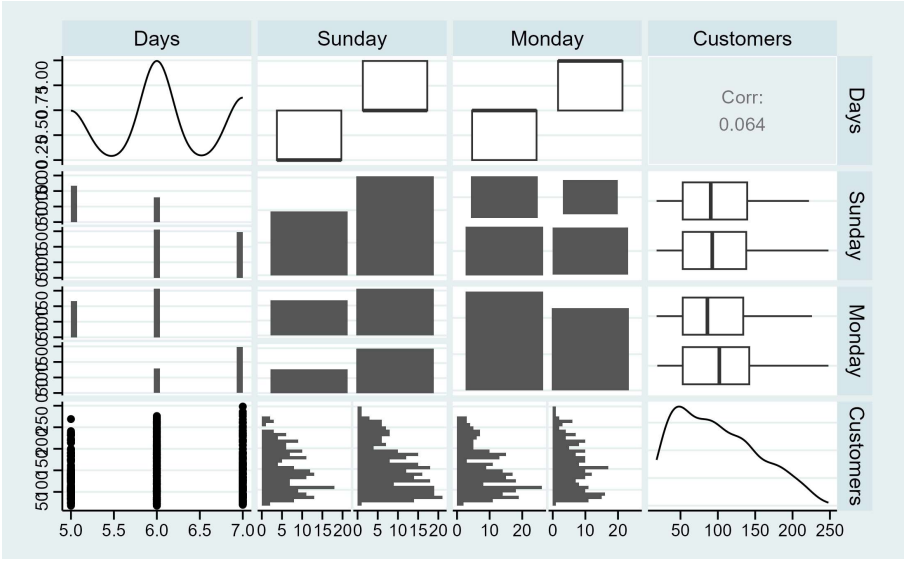
Plotting the Residuals

Hide

```
# Residuals vs. Fitted Values

residuals.lm2 <- resid(stepwise_lmodel)
fitted.lm2 <- fitted(stepwise_lmodel)

lm1= ggplot(data.frame(Fitted = fitted.lm2, Residuals = residuals.lm2),
            aes(x=Fitted, y=Residuals)) +
  geom_point() +
  geom_hline(yintercept=0, colour = "red") +
  labs(title = "Residuals vs Fitted Values Linear Regression Model",
       x = "Fitted Values", y= "Residuals")
ggsave("plots/lm1.png")
lm1

# uh-huh... definitely NOT randomly distributed around zero!!
# curved pattern in residuals suggests not all variables accounted for and that there is hete
roscedasticity. The can affect the validity of tests and confidence intervals.
```

The residuals show a very definite pattern. Not even close to random.

This curved pattern in the residuals suggests the relationship between the predictor and the response variables is not linear. Not surprising given the scatterplot between customers and price in exploratory analysis in plot/r1e



Hide

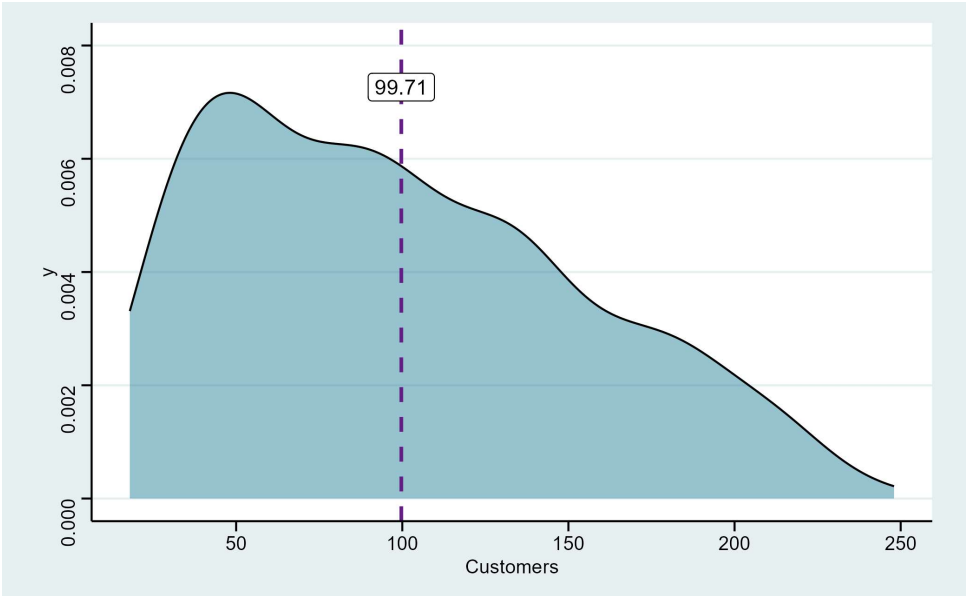
```
# Normal Q-Q Plot
lm2=ggplot(stepwise_lmodel, aes(sample = residuals.lm1)) +
stat_qq() +
stat_qq_line(color="red")
ggsave("plots/lm2.png")
lm2

# Residuals deviate from normality
# S-shaped curve indicates right skewness
```

A Q-Q (quantile-quantile) plot is used to assess whether the residuals of a regression model follow a normal distribution. When the points in a Q-Q plot form a curved pattern around the diagonal line, it indicates that the residuals deviate from normality. **S-shaped Curve**: This pattern suggests that the residuals are **skewed**. If the curve is S-shaped with the ends above the line and the middle below, it indicates **right skewness**.

Generalized Linear Model Analysis

Before model selection, need to scrutinize the distribution of my response variable: Customers.

Plot r6.png shows the shape of Customers and the mean of 99.71.



Hide

```
summary(rest$Customers)
```

Hide

```
# boxplot of distribution of Customers with summary data
r6a = ggplot(rests, aes(Customers)) +
geom_boxplot() +
  annotate("text", x=18,y=0.02, label= "Min=18.0", size=3) +
  annotate("text", x=245,y=0.02, label= "Max=248.0", size=3) +
  annotate("text", x=92,y=0.4, label= "Median=92.0", size=3) +
  annotate("text", x=53,y=-0.4, label= "1stQ=53.0", size=3) +
  annotate("text", x=138,y=-0.4, label= "3rdQ=138.0", size=3)
r6a
ggsave("plots/r6a.png")
```

Being count of customers Poisson could be a good choice, or gamma if considering it as a skewed continuous variable.

Hide

```
#change reference level to City since client's restaurant interest is that
rest1$Location <- relevel(rest1$Location, ref = "City")
levels(rest1$Location)
```

Hide

```
str(rest1)
```

Log-Linear attempt to explain Customers-Price

Hide

```
# Log linear attempt to fix residual issues

model_log_linear <- lm(log(Customers) ~ Price + Days, data = rest1)
summary(model_log_linear)
```

Hide

```
# AIC measures on model but can't directly compare to other non-log models.
AIC(model_log_linear)
extractAIC(model_log_linear)
```

- AIC
- Intercept - exp(4.736)= 113.68. When price and days are zero we still expect 113 customers. Theoretically.
- Price - exp(-0.0179)-1 = -0.0177 or -1.77% So for all else being constant, the number of customers decreases by approximately 1.77% for each unit increase in Price.
- Days - exp(0.0845)-1 = 0.0881 = 8.81%. For every extra day open, expected customers increase by 8.81%, holding all others constant.

Hide

```
# plotting observed vs predicted values plot

# Create a dataframe with observed and predicted values
# exp (on fitted) to undo log transformation
predicted_log_values <- data.frame(Observed = rest1$Customers,
                      Predicted = exp(fitted(model_log_linear)))

# Plot observed vs. predicted values
ggplot(predicted_log_values, aes(x = Observed, y = Predicted)) +
  geom_point(color = "deepskyblue4", alpha = 0.6) +
  geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed", size = 1) +
  labs(title = "Observed vs. Predicted Customers",
       x = "Observed Customers",
       y = "Predicted Customers") +
  theme(plot.title = element_text(hjust = 0.5))
```

Hide

```
# plotting residuals vs fitted
# Create a dataframe with fitted values and residuals
residual_values_log <- data.frame(
  Fitted = fitted(model_log_linear),
  Residuals = residuals(model_log_linear)
)

# Plot residuals vs. fitted values
ggplot(residual_values_log, aes(x = Fitted, y = Residuals)) +
  geom_point(color = "deepskyblue4", alpha = 0.6) +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed", size = 1) +
  labs(title = "Residuals vs Fitted Values",
       x = "Fitted Values",
       y = "Residuals") +
  theme(plot.title = element_text(hjust = 0.5))
```

Hide

```
# QQ-Plot time
# Create a Q-Q plot using stat_qq() and stat_qq_line()
ggplot(data = data.frame(residuals = residuals(model_log_linear)), aes(sample = residuals)) +
  stat_qq(color = "deepskyblue4", alpha = 0.6) +  # Q-Q plot points
  stat_qq_line(color = "red", linetype = "solid", size = 1) +  # Reference line
  labs(title = "Q-Q Plot of Residuals",
       x = "Theoretical Quantiles",
       y = "Sample Quantiles") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))
```

Hide

```
# using GLM with MASS stepAIC to estimate customers1.null first
# Estimate the Null Model first
customers.null1=glm(Customers~1, rest1, family=poisson)
```

Hide

```
# using GLM with MASS stepAIC to estimate customers1.full
# Estimating the Full Model. Selecting specific variables for the full model
# Including the interactive Sunday*Days

customers.full1=glm(Customers ~ Price + I(Price^2) + Location + Menu + Days + Distance + Sold
+ Service + Atmosphere + Food + Sunday + Sunday*Days + Menu*Price, rest1, family=poisson)
```

Hide

```
# stepwise model using stepAIC

stepwise_glm_model <- stepAIC(customers.null1, scope= list(lower=customers.null1, upper=custo
mers.full1), direction = "both", family=poisson)
```

Hide

```
summary(stepwise_glm_model)
```

Interpreting these results:

- **Coefficients**: These represent the estimated effect of each predictor on the **log** of the expected count of customers. exp(5.391) = approx 143 customers when all predictors are set to zero (menu is other)
- **Intercept**: The intercept of 4.859 is the expected log count of Customers when Price is 0, Days is 0 and the **Menu is Other**.
- **Price**: The coefficient for Price -0.0382 is negative and highly significant. exp(-0.0382)=0.9626 indicating that for every dollar increase in Price, the expected count of customers decreases by approximately 3.74% (1-0.9626)=0.0374.
- **Quadratic Term for Price**: Positive quadratic suggests a U-shaped relationship between Price and Customers.This can be indicative of two types of customer segments: one that is price-sensitive (decreasing customers with price)and one that associates higher price with higher value.
- **Days:** Each additional day open the expected number of customers increases by approximately 5.03%
- **OpenSunday**: Is associated with a 5.22% increase in the expected number of customers compared to other days.
- **Food**: A one point increase in the Food score is associated with a 1.7% increase in expected customers.
- **Atmosphere:** at -0.025, a one point increase in the Atmosphere rating indicates a 2.4% decrease in expected customers.

Hide

```
# Refitting model without non-signficant variable Sold.
Stepwise.glm.final <- glm(Customers ~ Price + I(Price^2) + Days + Sunday +
    Food + Atmosphere, family = poisson, data = rest1)
summary(Stepwise.glm.final)
```

- Need to now plot the residuals to check for linearity, heteroscedasticity or outliers.

Hide

```
# Fit the GLM model
model <- glm(Customers ~ Price + I(Price^2) + Days + Sunday +
    Food + Atmosphere, family = poisson, data = rest1)

# Calculate residuals
rest1$deviance_resid <- residuals(model, type = "deviance")
rest1$pearson_resid <- residuals(model, type = "pearson")
rest1$response_resid <- residuals(model, type = "response")

# Residuals vs. Fitted Values
# This plot helps check for non-linearity and heteroscedasticity.
# Definitely NOT a random scatter here. Not a good fit.
glm1 = ggplot(rest1, aes(x = fitted(model), y = deviance_resid)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE) +
  labs(title = "Deviance Residuals vs. Fitted Values",
       x = "Fitted Values",
       y = "Deviance Residuals")
ggsave("plots/glm1.png")

# Normal Q-Q Plot
# This plot helps assess the normality of residuals.
# Still fairly S-Shaped
glm2 = ggplot(rest1, aes(sample = deviance_resid)) +
  stat_qq() +
  stat_qq_line() +
  labs(title = "Normal Q-Q Plot of Deviance Residuals",
       x = "Theoretical Quantiles",
       y = "Deviance Residuals")
ggsave("plots/glm2.png")

# Scale-Location Plot
# This plot helps check for homoscedasticity (constant variance).
# Horizontal line is desired... and definitely not what we have.
glm3 = ggplot(rest1, aes(x = fitted(model), y = sqrt(abs(deviance_resid)))) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE) +
  labs(title = "Scale-Location Plot",
       x = "Fitted Values",
       y = "Square Root of |Deviance Residuals|")
ggsave("plots/glm3.png")

# Residuals vs. Leverage
# This plot helps identify influential data points.
glm4 = ggplot(rest1, aes(x = hatvalues(model), y = deviance_resid)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE) +
  labs(title = "Deviance Residuals vs. Leverage",
       x = "Leverage",
       y = "Deviance Residuals")
ggsave("plots/glm4.png")
```

Hide

```
glm1
```

Hide

```
# Check for overdispersion
dispersion <- sum(residuals(model, type = "pearson")^2) / df.residual(model)
dispersion
# Greater than 1, so overdispersion is likely and variance>mean
```

Hide

```
glm2
```

Hide

```
glm3
```

Hide

```
glm4
```

Hide

```
# Calculate Cook's distance
cooksd <- cooks.distance(model)

# Plot Cook's distance
plot(cooksd, type = "h", main = "Cook's Distance", ylab = "Cook's Distance")
abline(h = 4/(nrow(rest1)-length(coef(model))), col = "red")
```

Hide

```
lm_model <- lm(Customers ~ Price + I(Price^2) + Days + Atmosphere +
    Sunday + Food, data = rest1)
vif(lm_model)
```

Hide

```
# Trying the Negative Binomial Model that suits Overdispersion in count data

# Fit Negative Binomial model
nb_model <- glm.nb(Customers ~ Price + I(Price^2) + Days + Atmosphere +
    Sunday + Food, data = rest1)

# Summary of the model
summary(nb_model)
```

Hide

```
# Fit an initial GAM model
initial_gam <- gam(Customers ~ s(Price) + Days + Sold, family = poisson, data = rest1)
summary(initial_gam)
```

Hide

```
# Residuals vs. Fitted Values
plot(gam_model, residuals = TRUE, pch = 16, cex = 0.5)
abline(h = 0, col = "red")

# QQ Plot of Residuals
qqnorm(residuals(gam_model))
qqline(residuals(gam_model), col = "red")
```

Patterns in Residuals

Non-Linearity: If there's a clear pattern in the residuals vs. fitted values plot, it suggests that the model may not be capturing some aspect of the relationship between the predictors and the response variable. This could indicate the need for additional terms or a different model. Heteroscedasticity: If the spread of the residuals increases or decreases with the fitted values, it indicates heteroscedasticity. This can affect the validity of statistical tests and confidence intervals.

Variable engineering ->

Customers*Price = EstDayTakings

Transforming customers and price into a new variable called EstDayTakings

Hide

```
# trying a transformation that combines customers and price into income
rests$EstDayTakings <- rests$Customers * rests$Price
str(rests)
```

Hide

```
# remove columns average_income, avTakins, Potent.DayTakings
rest1$average_income <-NULL
rest1$avTakings <- NULL
rest1$Potent.DayTakings <- NULL
str(rest1)
```

Hide

```
r8 = ggplot(rests, aes(EstDayTakings)) +
geom_boxplot()
ggsave("plots/r8.png")
r8
```

Hide

```
# Density Plot of Rating with Mean
mean_avTakings <- mean(rests$EstDayTakings, na.rm = TRUE)

r8a = ggplot(rests, aes(EstDayTakings)) +
  geom_density(fill="deepskyblue4", alpha=0.4) +
  geom_vline(aes(xintercept=mean_avTakings),linetype="dashed",
                   color="darkorchid4", size=0.9) +
  annotate("label", x=(mean_avTakings), y=0.0015, label= (paste(round(mean_avTakings,2))),vju
st=2)+
  labs(title = "Density Plot of Estimated Day Takings",
       subtitle= "Engineered from Price per Customer by Customers Per Day",
       x = "Estimated Takings per Day in $")
r8a
ggsave("plots/r8a.png")

r8a
```

Linear Regression with new dependent variable: EstDayTakings

Hide

```
# New Subset called rest2
rest2 <- subset(rests, select= c(EstDayTakings, Location, Menu, Days, Distance, Service, Atmo
sphere, Food, Sold, Sunday))
```

Hide

```
#change reference level to City since client's restaurant interest is such
rest2$Location <- relevel(rest2$Location, ref = "City")
levels(rest2$Location)
```

Hide

```
# Build a null model for EstDayTakings
EstDayTakings.null=lm(EstDayTakings~1, rest2)
```

Hide

```
# AIC measures for the null model
AIC(EstDayTakings.null)
extractAIC(EstDayTakings.null)
```

Hide

```
# Build a full model for EstDayTakings
EstDayTakings.full=lm(EstDayTakings ~ Location + Menu + Days + Distance + Service + Atmospher
e + Food + Sold + Sunday + Days*Sunday, rest2)
```

Hide

```
# AIC measures for the full model
AIC(EstDayTakings.full)
extractAIC(EstDayTakings.full)
```

Hide

```
step_model <- stepAIC(EstDayTakings.null, scope= list(lower=EstDayTakings.null, upper=EstDayT
akings.full), direction = "both")
```

Hide

```
summary(step_model)
```

Hide

```
# Cross-check with multiple Linear Regression without unsignificant location.

model <- lm(EstDayTakings ~ Days + Sunday + Atmosphere + Sold +
    Food + Days:Sunday, data = rest2)

# Display the summary of the model
summary(model)
```

Interpretation

- The model explains about 37.2% of the variance in EstDayTakings, which is a moderate level of explanatory power. It suggests there are most likely other unmeasured variables in the environment influencing EstDayTakings.
- The residuals suggest that the model's predictions are generally unbiased, with a reasonable spread of errors. The median is close to zero and there is symmetry.
- The adjusted R-squared value confirms that the model is a good fit, considering the number of predictors.
- The significant F-statistic indicates that the model as a whole is statistically significant.

Plotting residuals to be certain…

Hide

```
# Fit the linear model
step_model <- lm(EstDayTakings ~ Days + Sunday + Atmosphere + Sold +
    Food + Days:Sunday, data = rest2)

# Calculate residuals and fitted values
rest2$residuals <- residuals(step_model)
rest2$fitted <- fitted(step_model)

# Residuals vs. Fitted Values
lm3 = ggplot(rest2, aes(x = fitted, y = residuals)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE, color = "blue") +
  labs(title = "Residuals vs. Fitted Values",
       x = "Fitted Values",
       y = "Residuals")
ggsave("plots/lm3.png")

# Normal Q-Q Plot
lm4 = ggplot(rest2, aes(sample = residuals)) +
  stat_qq() +
  stat_qq_line() +
  labs(title = "Normal Q-Q Plot",
       x = "Theoretical Quantiles",
       y = "Residuals")
ggsave("plots/lm4.png")

# Scale-Location Plot
lm5 = ggplot(rest2, aes(x = fitted, y = sqrt(abs(residuals)))) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE, color = "blue") +
  labs(title = "Scale-Location Plot",
       x = "Fitted Values",
       y = "Square Root of |Residuals|")
ggsave("plots/lm5.png")

# Residuals vs. Leverage
lm6 = ggplot(rest2, aes(x = hatvalues(step_model), y = residuals)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE, color = "blue") +
  labs(title = "Residuals vs. Leverage",
       x = "Leverage",
       y = "Residuals")
ggsave("plots/lm6.png")
```

Hide

```
lm3
```

Hide

```
lm4
```

Hide

```
lm5
```

Hide

```
lm6
```

Hide

```
# Exploring whether measuring any interaction terms may improve the model
# Fit the model with specific interaction terms
Step_model_interactions <- lm(EstDayTakings ~ Days + Rating + Sold + Days:Rating + Days:Sold
+ Rating:Sold, data = rest3)

# Display the summary of the model
summary(Step_model_interactions)
```

Switch Days for Monday & Sunday - GLM with gaussian family for cross-check

Hide

```
# New Subset called rest3
rest3 <- subset(rests, select= c(EstDayTakings, Location, Menu, Distance, Rating, Sold, Sunda
y, Monday, Days))
```

Hide

```
# Build a null model for EstDayTakings
EstDayTakings.null=glm(EstDayTakings~1, rest3, family=gaussian)
```

Hide

```
# Build a full model for EstDayTakings
EstDayTakings.full=glm(EstDayTakings ~ ., rest3, family=gaussian)
```

Hide

```
stepwise_glm_model1 <- stepAIC(EstDayTakings.null, scope= list(lower=EstDayTakings.null, uppe
r=EstDayTakings.full), direction = "both", family=gaussian)
```

Hide

```
summary(stepwise_glm_model1)
```

Hide

```
# trying model variations with interaction effects
# Days + Sunday + Sunday*Days + Rating + Sold

model1a <- lm(EstDayTakings ~ Days + Sunday + Rating + Sold + Sunday*Days, data = rest3)
summary(model1a)
```

Hide

```
# checking AIC on model 1a
AIC(model1a)
extractAIC(model1a)
```

Hide

```
# Fit the model with specific interaction effects
model1b <- lm(EstDayTakings ~ Rating + Sold + Sunday + Monday + Sunday:Monday, data = rest3)

# Display the summary of the model
summary(model1b)
```

Hide

```
# checking AIC on model 1b
AIC(model1b)
extractAIC(model1b)
```

Need to exclude Monday OR Days otherwise will have multicollinearity as Days when 7 will equal SundayYes + MondayYes. If using two of these variables, should include the interaction terms too.

Hide

```
# New Subset called rest4
rest4 <- subset(rests, select= c(EstDayTakings, Location, Menu, Distance, Food, Atmosphere, S
ervice, Sold, Sunday, Monday, Days))
```

Hide

```
# Build a null model for EstDayTakings
EstDayTakings.null=glm(EstDayTakings~1, rest4, family=gaussian)
```

Hide

```
# Build a full model for EstDayTakings
EstDayTakings.full=glm(EstDayTakings ~ ., rest4, family=gaussian)
```

Hide

```
# Model 1c with Rating broken back apart into its elements

model1c <- stepAIC(EstDayTakings.null, scope= list(lower=EstDayTakings.null, upper=EstDayTaki
ngs.full), direction = "both", family=gaussian)
```

Hide

```
summary(model1c)
```

Hide

```
# checking AIC on model 1c
AIC(model1c)
extractAIC(model1c)
```

Final Multiple Linear Regression Model - Model 1d

EstDayTakings ~ Days + Sunday + Atmosphere + Sold + Food + Days:Sunday

Model1d: Adj R-squared 0.3716 and AIC 6903 (down from 7129)

Hide

```
# Fit model 1c with specific interaction effects -> model1d
model1d <- lm(EstDayTakings ~ Days + Sunday + Atmosphere + Sold + Food + Days*Sunday, data =
rest4)

# Display the summary of the model
summary(model1d)
```

Hide

```
# checking AIC on model 1d
AIC(model1d)
extractAIC(model1d)
```

Plotting Model 1d Residuals

Hide

```
# Fit the linear model
model1d <- lm(EstDayTakings ~ Days + Sunday + Atmosphere + Sold + Food + Days*Sunday, data =
rest4)

# Calculate residuals and fitted values
rest4$residuals <- residuals(model1d)
rest4$fitted <- fitted(model1d)

# Residuals vs. Fitted Values
model1d_resid = ggplot(rest4, aes(x = fitted, y = residuals)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE, color = "blue") +
  labs(title = "Residuals vs. Fitted Values",
       x = "Fitted Values",
       y = "Residuals")
ggsave("plots/model1d_resid.png")
model1d_resid

# Normal Q-Q Plot
model1d_qq = ggplot(rest2, aes(sample = residuals)) +
  stat_qq() +
  stat_qq_line() +
  labs(title = "Normal Q-Q Plot",
       x = "Theoretical Quantiles",
       y = "Residuals")
ggsave("plots/model1d_qq.png")
model1d_qq
```

Factors to Consider with respect model fit:

1. **Nature of the Data**:
    - Business data often includes many unpredictable elements such as market trends, consumer behavior, and economic conditions. These can make it challenging to achieve very high R-squared values.
2. **Model Complexity**:
    - Simpler models with fewer predictors might have lower R-squared values but can still provide valuable insights. Adding more predictors can increase the R-squared but also risks overfitting.
3. **Purpose of the Model**:
    - If the goal is to identify key drivers and understand relationships, a moderate R-squared might be sufficient. For precise predictions, higher R-squared values are preferable.
4. **Comparative Benchmarks**:
    - Compare your model's R-squared with similar studies or models in your field. This can give you a better sense of what is considered good explanatory power in your specific context.

Practical Example:

In marketing, models predicting sales based on advertising spend, seasonality, and promotions might have R-squared values ranging from 20% to 50%. This is because many other factors (like competitor actions, economic shifts, and consumer preferences) also influence sales but are not included in the model.

Improving Explanatory Power:

- **Add Relevant Predictors**: Incorporate additional variables that might influence the response variable.
- **Interaction Terms**: Consider interactions between predictors.
- **Non-linear Relationships**: Explore non-linear models if relationships are not strictly linear.

In summary, while 37.16% explanatory power might seem modest, it can still be quite useful in providing actionable insights and guiding business decisions

Sold as binomial dependent with transformed variables

Hide

```
# Build a null model for Sold
Sold.null=glm(Sold ~ 1, rest3, family=binomial)
```

Hide

```
# Build a full model for Sold
Sold.full=glm(Sold ~ ., rest3, family=binomial)
```

Hide

```
stepwise_glm_model2 <- stepAIC(Sold.null, scope= list(lower= Sold.null, upper=Sold.full), dir
ection = "both", family=binomial)
```

Hide

```
summary(stepwise_glm_model2)
```

Interpreting the results:

The Intercept is the log-odds of 3.576:1 being sold when all predictors are zero.

Hide

```
# Log-odds value
log_odds <- 3.5764078

# Convert log-odds to probability manually
probability <- exp(log_odds) / (1 + exp(log_odds))
print(probability)
```

- **Interpretation of Intercept**: The log-odds of Sold being 1 (changed owners in the past 12 months) when location is “City”, Estimated Daily Takings are 0, and Combined Ratings is 0. Convert log-odds to probability is 0.9727 or 97.27%
- **Interpretation of Location**: The log-odds of Sold being 1 increase by 0.710 when the location is “Suburbs” compared to the reference category (likely “City”). This means that the odds of changing owners in the past 12 months is approximately 2.034 times higher in the “Suburbs” compared to the City as reference category.
- **Interpretation of EstDayTakings**: The log-odds of Sold being 1 decrease by 0.000915 for each unit increase in EstDayTakings. This means that for each unit increase in Takings, the odds of changing owners in the past 12 months decrease slightly (by a factor of 0.9991).
- **Interpretation of Rating**: The log-odds of Sold being 1 decrease by 0.141 for each unit increase in Rating. This means that for each unit increase in Rating, the odds of changing owners in the past 12 months decrease by a factor of approximately 0.868.

Sold as Binomial dependent with original variables

Hide

```
# Build a null model for Sold
Sold1.null=glm(Sold ~ 1, rests, family=binomial)
```

Hide

```
# Build a full model for Sold
Sold1.full=glm(Sold ~ Customers + Price + Days + Location + Atmosphere + Service + Food + Men
u + Distance, rests, family=binomial)
```

Hide

```
stepwise_glm_model3 <- stepAIC(Sold1.null, scope= list(lower= Sold1.null, upper=Sold1.full),
direction = "both", family=binomial)
```

Hide

```
summary(stepwise_glm_model3)
```

Looking at what factors predict Distance

Hide

```
# New Subset called rest4
rest4 <- subset(rest1, select= c(EstDayTakings, Location, Menu, Days, Distance, Rating, Sol
d))
```

Hide

```
# Build a null model for distance
# Family Gamma as continuous skewed variable
distance.null=glm(Distance ~ 1, rest4, family=Gamma)
```

Hide

```
# Build a full model for distance
distance.full=glm(Distance ~ ., rest4, family=Gamma)
```

Hide

```
stepwise_glm_model4 <- stepAIC(distance.null, scope= list(lower= distance.null, upper=distanc
e.full), direction = "both", family=Gamma)
```

Hide

```
summary(stepwise_glm_model4)
```

Hide

```
# Fit the GLM model
model <- glm(Distance ~ Menu + Location + Rating, family = Gamma, data = rest4)

# Calculate residuals
rest4$deviance_resid <- residuals(model, type = "deviance")
rest4$pearson_resid <- residuals(model, type = "pearson")
rest4$response_resid <- residuals(model, type = "response")

# Residuals vs. Fitted Values
# This plot helps check for non-linearity and heteroscedasticity.
# Definitely NOT a random scatter here. Not a good fit.
glm5 = ggplot(rest4, aes(x = fitted(model), y = deviance_resid)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE) +
  labs(title = "Deviance Residuals vs. Fitted Values",
       x = "Fitted Values",
       y = "Deviance Residuals")
ggsave("plots/glm5.png")

# Normal Q-Q Plot
glm6 = ggplot(rest4, aes(sample = deviance_resid)) +
  stat_qq() +
  stat_qq_line() +
  labs(title = "Normal Q-Q Plot",
       x = "Theoretical Quantiles",
       y = "Residuals")
ggsave("plots/lm6.png")
glm5
glm6
```

Filter data set by EstDayTakings >mean: rest5

Okay, so we need to make a value proposition.

I'd be especially interested in knowing what factors help predict the restaurants who have higher than average EstDayTakings. So filter on EstDayTakings and create a subset.

Hide

```
# New Subset called rest5
# specify dplyr select to avoid errors and conflicts
rest5 <- rest1 %>%
  filter(EstDayTakings >= mean_avTakings) %>%
  dplyr::select(EstDayTakings, Location, Menu, Days, Distance, Rating, Sold, Price, Customer
s)
str(rest5)
```

Density Plots for Price and Customers with the above average filtered dataset

Hide

```
# Density Plot of Price with Mean
mean_abav_Price <- mean(rest5$Price, na.rm = TRUE)

r9 = ggplot(rest5, aes(Price)) +
  geom_density(fill="deepskyblue4", alpha=0.4) +
  geom_vline(aes(xintercept=mean_abav_Price),linetype="dashed",
                   color="darkorchid4", size=0.9) +
  annotate("label", x=(mean_abav_Price), y=0.0015, label= (paste(round(mean_abav_Price,2))),v
just=2)
r9
ggsave=("plots/r9.png")
```

Hide

```
# Density Plot of Rating with Mean
mean_abav_Customer <- mean(rest5$Customers, na.rm = TRUE)

r9a = ggplot(rest5, aes(Customers)) +
  geom_density(fill="deepskyblue4", alpha=0.4) +
  geom_vline(aes(xintercept=mean_abav_Customer),linetype="dashed",
                   color="darkorchid4", size=0.9) +
  annotate("label", x=(mean_abav_Customer), y=0.0015, label= (paste(round(mean_abav_Customer,
2))),vjust=2)
r9a
ggsave=("plots/r9a.png")
```

Hide

```
# Density Plot of Rating with Mean
mean_abav_DayTakings <- mean(rest5$EstDayTakings, na.rm = TRUE)

r9b = ggplot(rest5, aes(EstDayTakings)) +
  geom_density(fill="deepskyblue4", alpha=0.4) +
  geom_vline(aes(xintercept=mean_abav_DayTakings),linetype="dashed",
                   color="darkorchid4", size=0.9) +
  annotate("label", x=(mean_abav_DayTakings), y=0.0015, label= (paste(round(mean_abav_DayTaki
ngs,2))),vjust=2)
r9b
ggsave=("plots/r9b.png")
```

GLM - for EstDayTakings with filtered dataset (258obs)

Using family = Gamma for skewed continuous variable

Hide

```
rest5.filt <- rest5 %>%
  dplyr::select(EstDayTakings, Location, Menu, Days, Distance, Rating, Sold)
```

Hide

```
# Build a null model for EstDayTakings
# Family Gamma as continuous skewed variable
EstDayTakings5.null=glm(EstDayTakings ~ 1, rest5.filt, family=Gamma)
```

Hide

```
# Build a full model for EstDayTakings - remember NOT Price / Customers
EstDayTakings5.full=glm(EstDayTakings ~ ., rest5.filt, family=Gamma)
```

Hide

```
stepwise_glm_model5 <- stepAIC(EstDayTakings5.null, scope= list(lower= EstDayTakings5.null, u
pper=EstDayTakings5.full), direction = "both", family=Gamma)
```

Hide

```
summary(stepwise_glm_model5)
```

Hide

```
# Fit the GLM model
model <- glm(EstDayTakings ~ Days, family = Gamma, data = rest5.filt)

# Calculate residuals
rest5.filt$deviance_resid <- residuals(model, type = "deviance")
rest5.filt$pearson_resid <- residuals(model, type = "pearson")
rest5.filt$response_resid <- residuals(model, type = "response")

# Residuals vs. Fitted Values
# This plot helps check for non-linearity and heteroscedasticity.

glm7 = ggplot(rest5.filt, aes(x = fitted(model), y = deviance_resid)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE) +
  labs(title = "Deviance Residuals vs. Fitted Values",
       x = "Fitted Values",
       y = "Deviance Residuals")
ggsave("plots/glm7.png")

# Days as predictor v takes on a limited number of distinct values, 5, 6 & 7
# this has led to grouped vertical lines of residuals in the residual plot


# Normal Q-Q Plot
glm8 = ggplot(rest5.filt, aes(sample = deviance_resid)) +
  stat_qq() +
  stat_qq_line() +
  labs(title = "Normal Q-Q Plot",
       x = "Theoretical Quantiles",
       y = "Residuals")
ggsave("plots/glm8.png")
glm7
glm8
```

The presence of three vertical lines of plotted points at the minimum, mean, and maximum values in the deviance residuals vs. fitted values plot is reflective of the discrete variable Days as the only predictor variable. Since Days takes on a limited number of distinct values, 5, 6 & 7 this has led to grouped vertical lines of residuals in the residual plot.