Rossmann Sales Forecasting

Anita Shahrokhan

Project Overview

This project aims to predict sales using historical data from January 2013 to March 2015. I implemented three models, Extreme Gradient Boosting (XGBoost), Random Forest, and Linear Regression using features such as store ID, day of the week, promotions, school holidays, store type, competition distance, and date components (year, month, day). After exploring the data through visualizations, I compared model performances to identify the most effective approach.

1. Data

The training dataset includes daily records from January 2013 to March 2015 across 20 stores, with the following columns:

- **Store**: Unique store ID
- **DayOfWeek**: Integer from 1 (Monday) to 7 (Sunday)
- **Sales**: Daily sales (target variable)
- **Customers**: Number of customers per day
- **Open**: Store status (1 = open, 0 = closed)
- **Promo**: Indicates if a promotion was active on that day
- **StateHoliday**: Holiday type (`a` = public, `b` = Easter, `c` = Christmas, `0` / `d` = none)
- **SchoolHoliday**: Indicates if schools were closed on that date
- **Date**: Date of the observation (from Jan 2013 to Mar 2015)

First We load the packages we need for this study:

```r
#------------------------------------------------------------------------------
#load packages
#------------------------------------------------------------------------------
library(readr)
library(randomForest)
library(xgboost)
library(dplyr)
library(lubridate)
library(gridExtra)
library(grid)
library(ggplot2)
library(patchwork)
library(ggpubr)
```

Next, we load the datasets, clean the data, and handle missing values. We remove records with closed stores (as sales are zero), replace all remaining missing values with zero, and decompose the date column into **day**, **month**, and **year**.

To evaluate model performance locally, we split the cleaned training data into an **80% training set** and a **20% validation set**.

```r
#------------------------------------------------------------------------------
# Load Train, Test, and Store data
#------------------------------------------------------------------------------
trainDat=read.csv("W21_train.csv", header = T)
storeDat =read.csv("W21_store_info.csv", header = T)

#merge store information and train and test
train_total=left_join(trainDat,storeDat, by = "Store")
train_total=train_total %>% filter(Sales > 0, Open == 1)

#check if we have missing values
train_total[is.na(train_total)]=0


#------------------------------------------------------------------------------
# Decompose date on training data
#------------------------------------------------------------------------------
train_total$month=as.integer(month(ymd(train_total$Date)))
train_total$year=as.integer(year(ymd(train_total$Date)))
train_total$day=as.integer(day(ymd(train_total$Date)))

#Remove the date column (after decomposing) and also StateHoliday(many zeros)
train_total=train_total[,-c(3,8)]

set.seed(42)  # for reproducibility

# Split the data to make preditcion
n <- nrow(train_total)
index <- sample(1:n, size = 0.8 * n)
train <- train_total[index, ]
test <- train_total[-index, ]


#------------------------------------------------------------------------------
#pick the variables
#------------------------------------------------------------------------------

Variables=names(train)[c(1,2,6:9,11:13)]
Variables
```

```
## [1] "Store"            "DayOfWeek"          "Promo"
## [4] "SchoolHoliday"    "StoreType"          "CompetitionDistance"
## [7] "month"            "year"               "day"
```

```r
#change the categorical variables to integer
for (i in Variables) {
  if (class(train[[i]])=="character") {
    levels <- unique(c(train[[i]], test[[i]]))
    train[[i]]=as.integer(factor(train[[i]], levels=levels))
    test[[i]]=as.integer(factor(test[[i]],  levels=levels))
  }
}
```

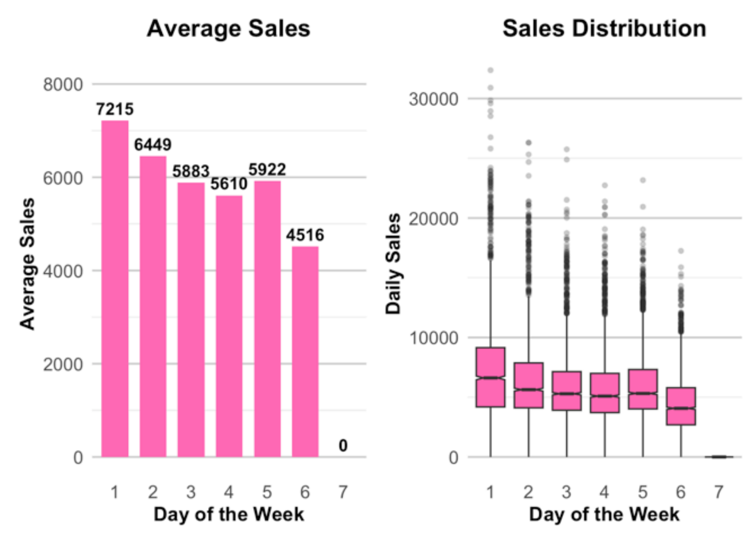
2. Data Exploration

After merging the datasets, I filtered out days when stores were closed, as sales on those days are zero and not useful for prediction. Missing values were replaced with zero, and the date column was decomposed into year, month, and day components.

2.1 Day of Week and Monthly Trends

Sales patterns show that Mondays and Fridays typically have higher sales, while Sundays have little to no sales due to store closures.

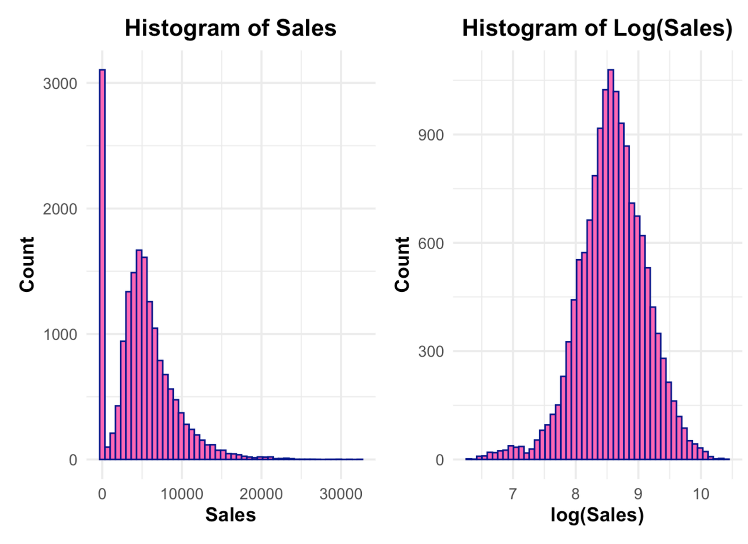
```r
#------------------------------------------------------------------------------
#Data Visualization
#------------------------------------------------------------------------------



#Average sale
Dayofweek_mean_sales = aggregate(trainDat$Sales, by=list(trainDat$DayOfWeek), FUN=mean)
data_dayofweek = data.frame(DayOfWeek=c("1", "2", "3","4","5","6","7"),
                            Sales=Dayofweek_mean_sales$x)
# Enhanced ggplot
plot_dayofweek1 <- ggplot(data_dayofweek, aes(x = DayOfWeek, y = Sales)) +
  geom_bar(stat = "identity", fill = "#FF69B4", width = 0.7) +  # soft pink tone
  geom_text(aes(label = round(Sales, 0)),
            vjust = -0.5, size = 4, color = "black", fontface = "bold") +
  labs(
    title = "Average Sales",
    x = "Day of the Week",
    y = "Average Sales"
  ) +
  theme_minimal(base_size = 13) +
  theme(
    plot.title = element_text(face = "bold", size = 16, hjust = 0.5),
    axis.title = element_text(face = "bold"),
    axis.text = element_text(size = 12),
    panel.grid.major.y = element_line(color = "grey80"),
    panel.grid.major.x = element_blank()
  ) +
  ylim(0, max(data_dayofweek$Sales) * 1.15)



# boxplot of sales
Data_box=data.frame(DayOfWeek=as.factor(trainDat$DayOfWeek),Sales=trainDat$Sales)
plot_dayofweek2 <- ggplot(Data_box, aes(x = DayOfWeek, y = Sales, fill = DayOfWeek)) +
  geom_boxplot(notch = TRUE, outlier.shape = 16, outlier.alpha = 0.3) +
  scale_fill_manual(values = rep("#FF69B4", 7)) +  # consistent pink fill
  labs(
    title = "Sales Distribution",
    x = "Day of the Week",
    y = "Daily Sales"
  ) +
  theme_minimal(base_size = 13) +
  theme(
    plot.title = element_text(face = "bold", size = 16, hjust = 0.5),
    axis.title = element_text(face = "bold"),
    axis.text = element_text(size = 12),
    legend.position = "none",  # remove legend for DayOfWeek
    panel.grid.major.y = element_line(color = "grey80"),
    panel.grid.major.x = element_blank()
  )

#1
plot_dayofweek1+plot_dayofweek2
```



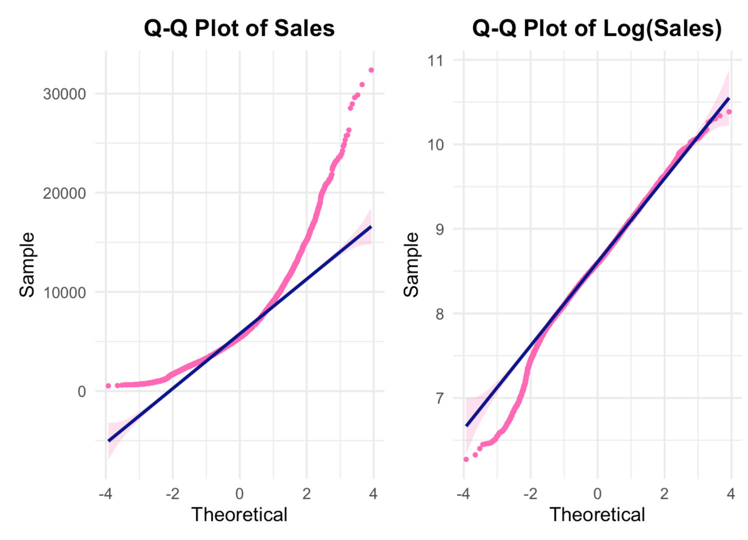
2.2 Sales Distribution

Sales data is right-skewed and non-normal. Applying a log transformation brings the distribution closer to a normal (bell) shape, which supports using log-sales as the response variable for modeling.

```r
# Histogram of raw sales
plot_sale <- ggplot(trainDat, aes(x = Sales)) +
  geom_histogram(color = "darkblue", fill = "#FF69B4", bins = 50) +
  labs(title = "Histogram of Sales",
       x = "Sales",
       y = "Count") +
  theme_minimal(base_size = 13) +
  theme(
    plot.title = element_text(face = "bold", hjust = 0.5),
    axis.title = element_text(face = "bold")
  )

# Histogram of log-transformed sales
plot_logsale <- ggplot(trainDat, aes(x = log(Sales))) +
  geom_histogram(color = "darkblue", fill = "#FF69B4", bins = 50) +
  labs(title = "Histogram of Log(Sales)",
       x = "log(Sales)",
       y = "Count") +
  theme_minimal(base_size = 13) +
  theme(
    plot.title = element_text(face = "bold", hjust = 0.5),
    axis.title = element_text(face = "bold")
  )

# 2
plot_sale + plot_logsale
```




```r
# Q-Q for raw Sales
qq_sales <- ggqqplot(train$Sales, color = "#FF69B4", size = 1) +
  stat_qq_line(color = "darkblue", size = 1) +  # Baseline color here
  labs(title = "Q-Q Plot of Sales") +
  theme_minimal(base_size = 13) +
  theme(plot.title = element_text(face = "bold", hjust = 0.5))

# Q-Q for log(Sales)
qq_logsales <- ggqqplot(log(train$Sales), color = "#FF69B4", size = 1) +
  stat_qq_line(color = "darkblue", size = 1) +
  labs(title = "Q-Q Plot of Log(Sales)") +
  theme_minimal(base_size = 13) +
  theme(plot.title = element_text(face = "bold", hjust = 0.5))

#3
qq_sales + qq_logsales
```



2.3 Summary

- Closed days and certain holidays are removed from training data.
- Log transformation improves the distribution of the target variable.
- Final features selected include: **Store**, **Day of the Week**, **Promo**, **School Holiday**, **Store Type**, **Competition Distance**, **Year**, **Month**, and **Day**. The response variable is **log(Sales)**.

3. Prediction

In this section, I implement three different models to predict daily sales: **XGBoost**, **Random Forest**, and **Linear Regression**.

These models were selected to represent a mix of predictive approaches:

- **XGBoost** is a powerful gradient boosting technique known for its accuracy and efficiency in handling structured data and non-linear relationships.
- **Random Forest** is a robust ensemble method that helps reduce overfitting and captures variable interactions well.
- **Linear Regression** provides a simple,baseline model to compare against more complex approaches.

All models use the same set of predictors selected during the data exploration phase

In the following subsections, I describe how each model was implemented and evaluated by **Root Mean Square Percentage Error (RMSPE)**.

$$\text{RMSPE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{\hat{y}_i - y_i}{y_i}\right)^2}$$

where $\hat{y}_i$ is the predicted sales and y_i is the actual sales.

```r
#------------------------------------------------------------------------------
#model Linear Regression
#------------------------------------------------------------------------------

#Fit model on 80% training split
lm_model <- lm(log(Sales + 1) ~ Store + DayOfWeek + Promo + SchoolHoliday +
                 StoreType + CompetitionDistance + month + year + day,
               data = train)

#Predict on validation set (excluding the Sales column)
valid_features <- test[, !(names(test) %in% c("Sales"))]
preds_regression <- as.numeric(exp(predict(lm_model, newdata = valid_features)) - 1)




#true values
actual_sales <- test$Sales

rmpse_regression <- sqrt(mean(((preds_regression / actual_sales) - 1)^2))



#------------------------------------------------------------------------------
#Random Forest
#------------------------------------------------------------------------------
##parameters
mtry=5
ntree=50
samplesize=nrow(train)
##Random Forest
RandomForest =randomForest(train[,Variables],
                           log(train$Sales+1),
                           mtry=mtry,
                           ntree=ntree,
                           sampsize=samplesize,
                           do.trace=FALSE)


importance(RandomForest, type = 1)
```

```
##
## Store
## DayOfWeek
## Promo
## SchoolHoliday
## StoreType
## CompetitionDistance
## month
## year
## day
```

```r
#prediction
preds_randomforest<- as.numeric(exp(predict(RandomForest, newdata = valid_features)) - 1)


rmpse_randomforest <- sqrt(mean(((preds_randomforest / actual_sales) - 1)^2))


#------------------------------------------------------------------------------
#Xgboost
#------------------------------------------------------------------------------
###spliting training data  20% in test and 80% training
train_integer=train[,Variables]
###set.seed(2)
Index=sample(1:nrow(train), 0.8 * (nrow(train)))
###Matrix from
Mat_val=xgb.DMatrix(data=data.matrix(train_integer[Index,]),label=log(train$Sales+1)[Index])
Mat_train=xgb.DMatrix(data=data.matrix(train_integer[-Index,]),label=log(train$Sales+1)[-Index])
watchlist=list(val=Mat_val,train=Mat_train)
###Parameters
eta                  = 0.02
max_depth            = 10
subsample            = 0.9
colsample_bytree     = 0.7
###parameter for the best tune
Parameter = list(objective= "reg:linear",
                 booster = "gbtree",
                 eta                  = eta,
                 max_depth            = max_depth,
                 subsample            = subsample,
                 colsample_bytree     = colsample_bytree)
###
nrounds=500
###estimate
XGBoost = xgb.train(params               = Parameter,
                    data                 = Mat_train,
                    nrounds              = nrounds,
                    verbose              = 0,
                    watchlist            = watchlist,
                    maximize             = FALSE)
```

```
## [21:05:22] WARNING: src/objective/regression_obj.cu:213: reg:linear is now deprecated in favor of reg:squarede
rror.
```

```r
preds_XGBoost<- exp(predict(XGBoost,  data.matrix(test[,Variables]))) - 1


rmpse_XGBoost <- sqrt(mean(((preds_XGBoost / actual_sales) - 1)^2))
```

Model Performance Comparison (RMSPE)

Model	RMSPE
Linear Regression	0.587
Random Forest	0.139
XGBoost	0.125

4. Summary and Results

This study aimed to predict store sales using nearly three years of historical data. After cleaning and exploring the dataset, several key decisions and findings were made:

- Days with zero sales were excluded, and the **StateHoliday** and **Customers** columns were dropped from modeling.
- Visualizations revealed that the **sales distribution is right-skewed**, and a **log transformation** on sales improved normality.
- Three models were applied: **XGBoost**, **Random Forest**, and **Linear Regression**.
- Based on model evaluation, **XGBoost** achieved the best predictive performance on the validation set.

These findings suggest that boosting models, combined with proper feature engineering and transformation, are highly effective for structured retail forecasting tasks.