Problem 1

AUTHOR
Putri Nisrina Az-Zahra - M0501241050

Demand Forecasting

It is a meal delivery company which operates in multiple cities. They have various fulfillment centers in these cities for dispatching meal orders to their customers. The client wants you to help these centers with demand forecasting for upcoming weeks so that these centers will plan the stock of raw materials accordingly.

Goals : **Predict the demand for the next 10 weeks** (Weeks: 136-145) for the center-meal combinations in the test set ; Forecasting accurately number of order (num_order)

```
# Mengatur working directory
setwd("E:\\S2 IPB\\SEMESTER 2\\Pembelajaran Mesin Statistika\\Praktikum\\Problem 1")
```

Load Library

```
library(readxl)      # Membaca file Excel
library(skimr)       # Ringkasan statistik eksploratif
library(ggplot2)     # Visualisasi data berbasis grammar of graphics
library(tidyverse)   # Kumpulan paket untuk manipulasi & analisis data
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✔ dplyr     1.1.4     ✔ readr     2.1.5
✔ forcats   1.0.0     ✔ stringr   1.5.1
✔ lubridate 1.9.3     ✔ tibble    3.2.1
✔ purrr     1.0.2     ✔ tidyr     1.3.1
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

```
library(reshape2)    # Transformasi format data (wide ↔ long)
```

```
Attaching package: 'reshape2'

The following object is masked from 'package:tidyr':

    smiths
```

```
library(corrplot)    # Visualisasi matriks korelasi
```

```
corrplot 0.94 loaded
```

```r
library(gridExtra)   # Menggabungkan beberapa plot dalam satu grid
```

```
Attaching package: 'gridExtra'

The following object is masked from 'package:dplyr':

    combine
```

```r
library(caret)       # Pemrosesan data & validasi model
```

```
Loading required package: lattice

Attaching package: 'caret'

The following object is masked from 'package:purrr':

    lift
```

```r
library(data.table)  # Manipulasi data skala besar
```

```
Attaching package: 'data.table'

The following objects are masked from 'package:reshape2':

    dcast, melt

The following objects are masked from 'package:lubridate':

    hour, isoweek, mday, minute, month, quarter, second, wday, week,
    yday, year

The following objects are masked from 'package:dplyr':

    between, first, last

The following object is masked from 'package:purrr':

    transpose
```

```r
library(lightgbm)    # Model Gradient Boosting yang cepat
library(xgboost)     # Model Extreme Gradient Boosting
```

```
Attaching package: 'xgboost'

The following object is masked from 'package:dplyr':

    slice
```

```r
library(Matrix)      # Operasi matriks efisien
```

```
Attaching package: 'Matrix'

The following objects are masked from 'package:tidyr':

    expand, pack, unpack
```

```r
library(randomForest)# Algoritma Random Forest
```

```
randomForest 4.7-1.2
Type rfNews() to see new features/changes/bug fixes.

Attaching package: 'randomForest'

The following object is masked from 'package:gridExtra':

    combine

The following object is masked from 'package:dplyr':

    combine

The following object is masked from 'package:ggplot2':

    margin
```

```r
library(fastDummies) # Pembuatan variabel dummy cepat
```

```
Warning: package 'fastDummies' was built under R version 4.4.2
```

Load Data

Train

```r
train <- read_xlsx("train.xlsx")
head(train)
```

```
# A tibble: 6 × 9
      id  week center_id meal_id checkout_price base_price emailer_for_promotion
   <dbl> <dbl>     <dbl>   <dbl>          <dbl>      <dbl>                  <dbl>
1 1.38e6     1        55    1885           137.       152.                      0
2 1.47e6     1        55    1993           137.       136.                      0
3 1.35e6     1        55    2539           135.       136.                      0
4 1.34e6     1        55    2139           340.       438.                      0
5 1.45e6     1        55    2631           244.       242.                      0
6 1.27e6     1        55    1248           251.       252.                      0
# ℹ 2 more variables: homepage_featured <dbl>, num_orders <dbl>
```

Test

```
test <- read_xlsx("test.xlsx")
head(test)
```

```
# A tibble: 6 × 8
      id  week center_id meal_id checkout_price base_price emailer_for_promotion
   <dbl> <dbl>     <dbl>   <dbl>          <dbl>      <dbl>                  <dbl>
1 1.02e6   136        55    1885          148.        148.                      0
2 1.40e6   136        55    1993          151.        151.                      0
3 1.01e6   136        55    2539          152.        151.                      0
4 1.04e6   136        55    2631           96.0       166.                      0
5 1.02e6   136        55    1248           97         166.                      0
6 1.19e6   136        55    1778          165.        165.                      0
# i 1 more variable: homepage_featured <dbl>
```

Meal Info

```
meal_info <- read.csv("meal_info.csv")
head(meal_info)
```

```
  meal_id  category cuisine
1    1885 Beverages    Thai
2    1993 Beverages    Thai
3    2539 Beverages    Thai
4    1248 Beverages  Indian
5    2631 Beverages  Indian
6    1311    Extras    Thai
```

Fulfilment Center Info

```
fulfilment_center_info <- read.csv("fulfilment_center_info.csv")
head(fulfilment_center_info)
```

```
  center_id city_code region_code center_type op_area
1        11       679          56      TYPE_A     3.7
2        13       590          56      TYPE_B     6.7
3       124       590          56      TYPE_C     4.0
4        66       648          34      TYPE_A     4.1
5        94       632          34      TYPE_C     3.6
6        64       553          77      TYPE_A     4.4
```

Pre-Processing Data

Added Meal Info

```
# Join Data Meal Info by meal_id
train <- train %>%
```

```
  left_join(meal_info, by = "meal_id")

head(train)
```

```
# A tibble: 6 × 11
      id  week center_id meal_id checkout_price base_price emailer_for_promotion
   <dbl> <dbl>     <dbl>   <dbl>          <dbl>      <dbl>                  <dbl>
1 1.38e6     1        55    1885           137.       152.                      0
2 1.47e6     1        55    1993           137.       136.                      0
3 1.35e6     1        55    2539           135.       136.                      0
4 1.34e6     1        55    2139           340.       438.                      0
5 1.45e6     1        55    2631           244.       242.                      0
6 1.27e6     1        55    1248           251.       252.                      0
# i 4 more variables: homepage_featured <dbl>, num_orders <dbl>,
#   category <chr>, cuisine <chr>
```

Added Fulfillment Center Info

```
# Join Data Fulfillment Center Info by center_id
train <- train %>%
  left_join(fulfilment_center_info, by = "center_id")

head(train)
```

```
# A tibble: 6 × 15
      id  week center_id meal_id checkout_price base_price emailer_for_promotion
   <dbl> <dbl>     <dbl>   <dbl>          <dbl>      <dbl>                  <dbl>
1 1.38e6     1        55    1885           137.       152.                      0
2 1.47e6     1        55    1993           137.       136.                      0
3 1.35e6     1        55    2539           135.       136.                      0
4 1.34e6     1        55    2139           340.       438.                      0
5 1.45e6     1        55    2631           244.       242.                      0
6 1.27e6     1        55    1248           251.       252.                      0
# i 8 more variables: homepage_featured <dbl>, num_orders <dbl>,
#   category <chr>, cuisine <chr>, city_code <int>, region_code <int>,
#   center_type <chr>, op_area <dbl>
```

Updated Data

```
# Melihat Struktur Data
str(train)
```

```
tibble [456,548 × 15] (S3: tbl_df/tbl/data.frame)
 $ id                   : num [1:456548] 1379560 1466964 1346989 1338232 1448490 ...
 $ week                 : num [1:456548] 1 1 1 1 1 1 1 1 1 1 ...
 $ center_id            : num [1:456548] 55 55 55 55 55 55 55 55 55 55 ...
 $ meal_id              : num [1:456548] 1885 1993 2539 2139 2631 ...
 $ checkout_price       : num [1:456548] 137 137 135 340 244 ...
 $ base_price           : num [1:456548] 152 136 136 438 242 ...
 $ emailer_for_promotion: num [1:456548] 0 0 0 0 0 0 0 0 0 0 ...
 $ homepage_featured    : num [1:456548] 0 0 0 0 0 0 0 0 0 1 ...
```

```
 $ num_orders           : num [1:456548] 177 270 189 54 40 28 190 391 472 676 ...
 $ category             : chr [1:456548] "Beverages" "Beverages" "Beverages"
"Beverages" ...
 $ cuisine              : chr [1:456548] "Thai" "Thai" "Thai" "Indian" ...
 $ city_code            : int [1:456548] 647 647 647 647 647 647 647 647 647 647 ...
 $ region_code          : int [1:456548] 56 56 56 56 56 56 56 56 56 56 ...
 $ center_type          : chr [1:456548] "TYPE_C" "TYPE_C" "TYPE_C" "TYPE_C" ...
 $ op_area              : num [1:456548] 2 2 2 2 2 2 2 2 2 ...
```

Dataset ini berisi 456,548 baris dan 15 kolom yang terdiri atas 11 variabel numerik dan 4 variabel kategorik.

- `id` : ID unik untuk setiap data.
- `week` : Periode waktu pemesanan makanan.
- `center_id` : ID pusat distribusi makanan.
- `meal_id` : ID makanan yang ditawarkan.
- `checkout_price` : Harga akhir makanan yang dibayar pelanggan setelah diskon.
- `base_price` : Harga awal makanan sebelum diskon.
- `emailer_for_promotion` : Indikator apakah makanan dipromosikan melalui email (0 = tidak, 1 = ya).
- `homepage_featured` : Indikator apakah makanan ditampilkan di halaman utama untuk promosi (0 = tidak, 1 = ya).
- `num_orders` : Jumlah pesanan makanan dan berperan sebagai variabel target.
- `category` : Kategori makanan atau minuman.
- `cuisine` : Jenis masakan berdasarkan gaya kuliner.
- `city_code` : Kode kota tempat pusat distribusi berada.
- `region_code` : Kode wilayah tempat pusat distribusi berada.
- `center_type` : Jenis pusat distribusi makanan.
- `op_area` : Luas operasional pusat distribusi dalam satuan tertentu.

```r
# Mengecek Missing Value pada data train
any(is.na(train))
```

```
[1] FALSE
```

Tidak ada variabel yang memiliki missing value (NA) pada data train sehingga tidak perlu dilakukan penanganan seperti imputasi atau penghapusan baris/kolom dengan nilai NA.

```r
# Split Data as Validation
test_validation <- train %>% filter((week >= 136 & week <= 145))
```

```
glimpse(test_validation)
```

```
Rows: 32,821
Columns: 15
$ id                    <dbl> 1017495, 1395634, 1007493, 1042952, 1022147, 119…
$ week                  <dbl> 136, 136, 136, 136, 136, 136, 136, 136, 136, 136…
$ center_id             <dbl> 55, 55, 55, 55, 55, 55, 55, 55, 55, 55, 55, 55, …
$ meal_id               <dbl> 1885, 1993, 2539, 2631, 1248, 1778, 1062, 2707, …
$ checkout_price        <dbl> 148.44, 151.38, 152.35, 96.03, 97.00, 164.90, 17…
$ base_price            <dbl> 148.44, 151.38, 151.35, 165.93, 165.93, 164.90, …
$ emailer_for_promotion <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, …
$ homepage_featured     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, …
$ num_orders            <dbl> 134, 216, 163, 95, 14, 136, 310, 95, 230, 134, 1…
$ category              <chr> "Beverages", "Beverages", "Beverages", "Beverage…
$ cuisine               <chr> "Thai", "Thai", "Thai", "Indian", "Indian", "Ita…
$ city_code             <int> 647, 647, 647, 647, 647, 647, 647, 647, 647, 647…
$ region_code           <int> 56, 56, 56, 56, 56, 56, 56, 56, 56, 56, 56, 56, …
$ center_type           <chr> "TYPE_C", "TYPE_C", "TYPE_C", "TYPE_C", "TYPE_C"…
$ op_area               <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, …
```

Data train dengan minggu ke-136 sampai minggu ke-145 digunakan untuk validasi jumlah num_orders dari hasil prediksi model yaitu sebanyak 32.821 baris data.

```
# Data Week 1 - 135
train <- train %>% filter(!(week >= 136 & week <= 145))
glimpse(train)
```

```
Rows: 423,727
Columns: 15
$ id                    <dbl> 1379560, 1466964, 1346989, 1338232, 1448490, 127…
$ week                  <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
$ center_id             <dbl> 55, 55, 55, 55, 55, 55, 55, 55, 55, 55, 55, 55, …
$ meal_id               <dbl> 1885, 1993, 2539, 2139, 2631, 1248, 1778, 1062, …
$ checkout_price        <dbl> 136.83, 136.83, 134.86, 339.50, 243.50, 251.23, …
$ base_price            <dbl> 152.29, 135.83, 135.86, 437.53, 242.50, 252.23, …
$ emailer_for_promotion <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, …
$ homepage_featured     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, …
$ num_orders            <dbl> 177, 270, 189, 54, 40, 28, 190, 391, 472, 676, 8…
$ category              <chr> "Beverages", "Beverages", "Beverages", "Beverage…
$ cuisine               <chr> "Thai", "Thai", "Thai", "Indian", "Indian", "Ind…
$ city_code             <int> 647, 647, 647, 647, 647, 647, 647, 647, 647, 647…
$ region_code           <int> 56, 56, 56, 56, 56, 56, 56, 56, 56, 56, 56, 56, …
$ center_type           <chr> "TYPE_C", "TYPE_C", "TYPE_C", "TYPE_C", "TYPE_C"…
$ op_area               <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, …
```

Data train dengan minggu ke-1 sampai minggu ke-135 sebanyak 423.727 baris dan 15 kolom digunakan untuk melatih model yang dapat memprediksi variabel num_orders atau jumlah pesanan makanan.

Exploration Data

Summary Data

```
skim_without_charts(train)
```

Data summary

Name	train
Number of rows	423727
Number of columns	15
_____	
Column type frequency:	
character	3
numeric	12
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
category	0	1	4	12	0	14	0
cuisine	0	1	4	11	0	4	0
center_type	0	1	6	6	0	3	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50
id	0	1	1249964.44	144343.12	1.000e+06	1124878.50	1249955.00
week	0	1	69.68	38.69	1.000e+00	37.00	70.00
center_id	0	1	82.11	45.98	1.000e+01	43.00	76.00
meal_id	0	1	2023.96	547.37	1.062e+03	1558.00	1971.00
checkout_price	0	1	331.53	153.09	2.970e+00	228.01	292.03
base_price	0	1	353.73	160.92	5.535e+01	243.50	309.49
emailer_for_promotion	0	1	0.08	0.28	0.000e+00	0.00	0.00
homepage_featured	0	1	0.11	0.31	0.000e+00	0.00	0.00
num_orders	0	1	264.55	403.57	1.300e+01	54.00	136.00
city_code	0	1	601.56	66.21	4.560e+02	553.00	596.00
region_code	0	1	56.61	17.64	2.300e+01	34.00	56.00
op_area	0	1	4.08	1.09	9.000e-01	3.60	4.00

Check Duplicated Data

```r
# Mengecek duplikasi data
sum(duplicated(train))
```

```
[1] 0
```

Tidak ada data yang duplikat pada data train sehingga tidak perlu dilakukan penghapusan baris data yang duplikat.

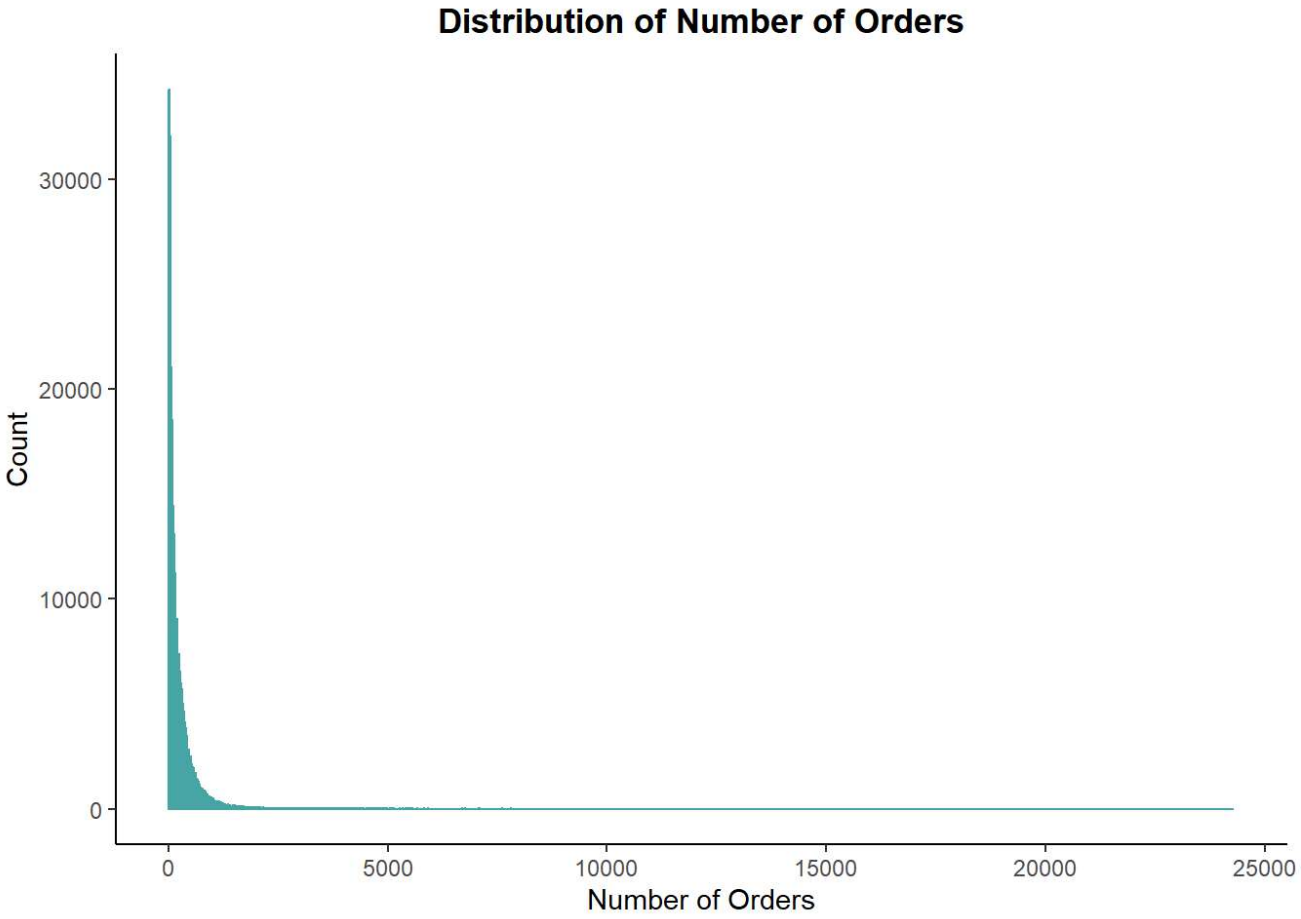
Num of Order

```r
train_filtered <- train %>%
  group_by(week) %>%
  summarize(avg_orders = mean(num_orders, na.rm = TRUE))

ggplot(train_filtered, aes(x = week, y = avg_orders)) +
  geom_line(color = "#48A6A7", linewidth = 1) +
  labs(title = "Average Number of Orders", x = "Week", y = "Avg Orders") +
  theme_classic() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold")
  )
```

Plot deret waktu dari `num_orders` menunjukkan **fluktuasi yang signifikan** dalam jumlah pesanan makanan per minggu, dengan beberapa lonjakan tajam yang kemungkinan dipengaruhi oleh promosi atau faktor musiman. Sebaliknya, terdapat pula penurunan drastis pada beberapa titik. Pola

ini menunjukkan kemungkinan **adanya siklus permintaan yang berulang**, meskipun di bagian akhir grafik tren tampak lebih stabil.

```
ggplot(train, aes(x = num_orders)) +
  geom_histogram(binwidth = 10, fill = "#48A6A7", color = "#48A6A7", alpha = 0.8) +
  labs(title = "Distribution of Number of Orders", x = "Number of Orders", y = "Cour
  theme_classic() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold")
  )
```



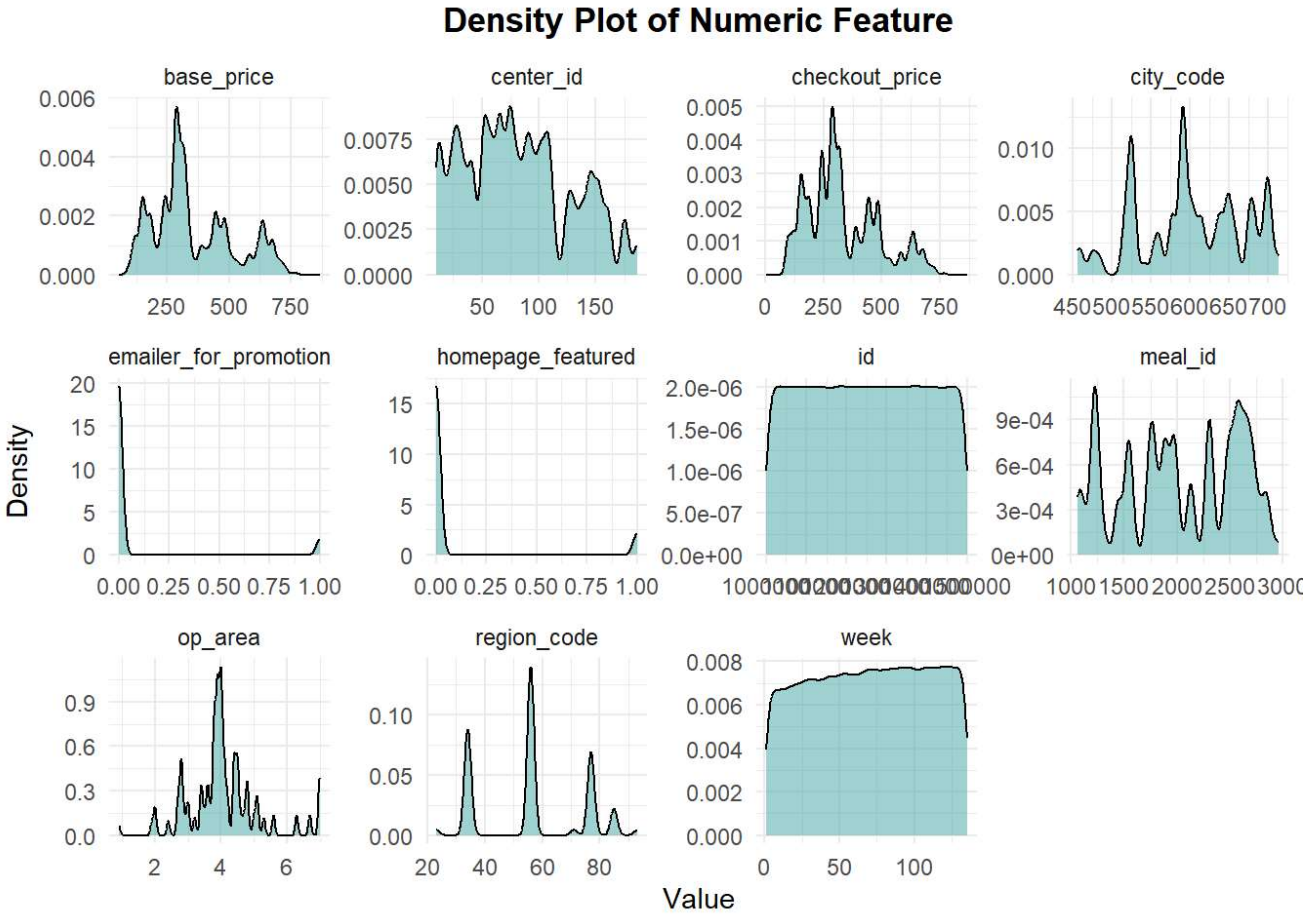
Plot ini menunjukkan distribusi jumlah pesanan makanan dengan sebagian besar pesanan berada pada nilai yang relatif kecil. Sebagian besar **jumlah pesanan terkonsentrasi di sekitar nol** hingga beberapa ratus pesanan, sementara ada beberapa outlier dengan jumlah pesanan yang jauh lebih tinggi, bahkan mendekati 25.000. Distribusi ini mengindikasikan bahwa sebagian besar makanan memiliki permintaan yang rendah hingga sedang, sementara hanya sedikit yang memiliki permintaan sangat tinggi.

Feature

Density Plot of Numeric Feature

```
train %>%
  select(-num_orders) %>%
```

```
  select(where(is.numeric)) %>%
  pivot_longer(cols = everything(), names_to = "variable", values_to = "value") %>%
  ggplot(aes(x = value)) +
  facet_wrap(~ variable, scales = "free") +
  geom_density(fill = "#48A6A7", alpha = 0.5) +
  labs(
    title = "Density Plot of Numeric Feature",
    x = "Value",
    y = "Density"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold")
  )
```

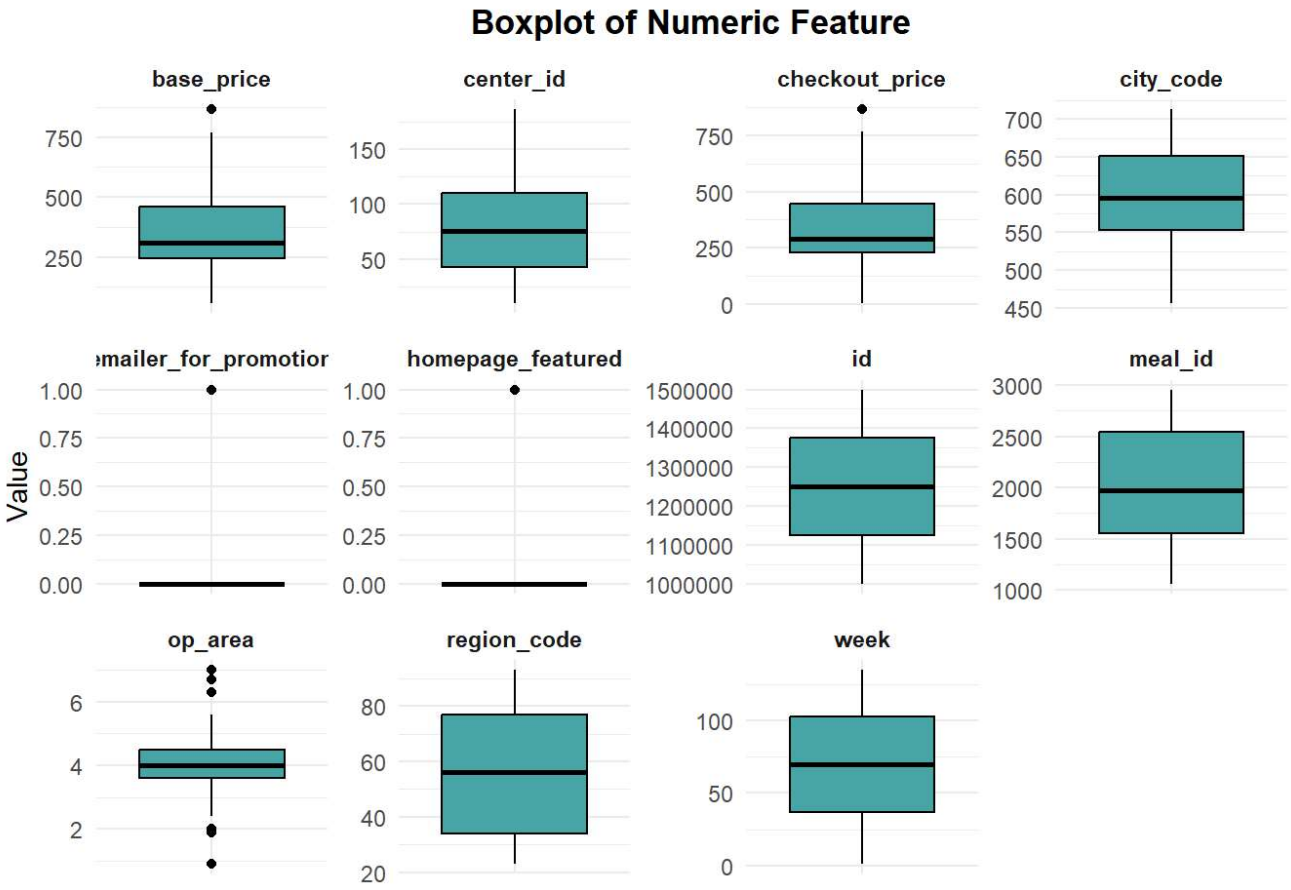


- `base_price` dan `checkout_price` : menunjukkan distribusi multimodal, menunjukkan adanya beberapa kelompok harga yang dominan.
- `center_id` dan `city_code` : menunjukkan pola multimodal, mengindikasikan adanya beberapa pusat distribusi dan kode kota dengan jumlah yang cukup banyak.
- `emailer_for_promotion` dan `homepage_featured` : memiliki distribusi yang sangat miring ke kiri, menunjukkan bahwa sebagian besar nilai adalah nol, menandakan hanya sedikit item yang dipromosikan melalui email atau ditampilkan di halaman utama.
- `id` dan `meal_id` : memiliki distribusi yang hampir uniform, mengindikasikan bahwa nilai-nilainya tersebar merata dalam rentang tertentu.

- `op_area` dan `region_code` : menunjukkan distribusi multimodal, mengindikasikan adanya beberapa kategori operasional dan wilayah dengan karakteristik berbeda.
- `week` : memiliki distribusi yang cukup merata, yang mungkin menunjukkan bahwa data mencakup periode waktu yang cukup panjang tanpa banyak variasi dalam cakupan minggu.

Boxplot of Numeric Feature

```
train %>%
  select(-num_orders) %>%
  select(where(is.numeric)) %>%
  pivot_longer(cols = everything(), names_to = "variable", values_to = "value") %>%
  ggplot(aes(x = "", y = value)) +
  geom_boxplot(fill = "#48A6A7", color = "black") +
  theme_minimal() +
  labs(title = "Boxplot of Numeric Feature",
       x = "",
       y = "Value") +
  facet_wrap(~ variable, scales = "free", ncol = 4) +
  theme(
    axis.text.x = element_blank(),
    axis.ticks.x = element_blank(),
    strip.text = element_text(face = "bold"),
    plot.title = element_text(hjust = 0.5, face = "bold")
  )
```



- `base_price` dan `checkout_price` : memiliki distribusi dengan beberapa pencilan (outlier) di bagian atas, menunjukkan adanya beberapa harga yang jauh lebih tinggi dibandingkan mayoritas data.
- `emailer_for_promotion` dan `homepage_featured` : memiliki distribusi yang sangat tidak merata dengan dominasi nilai nol, tetapi ada beberapa pencilan yang menunjukkan bahwa hanya sebagian kecil data yang memiliki nilai 1.
- `op_area` : menunjukkan beberapa outlier, baik di bagian bawah maupun atas, menandakan variasi yang cukup besar dalam area operasional.
- `id`, `meal_id`, `region_code`, dan `week` : memiliki distribusi yang lebih merata tanpa outlier signifikan, menunjukkan bahwa data pada fitur ini tersebar secara konsisten dalam rentangnya.
- `center_id` dan `city_code` : menunjukkan distribusi yang cukup lebar tetapi tidak memiliki pencilan signifikan, menandakan bahwa pusat distribusi dan kode kota tersebar dalam kisaran yang cukup stabil.

Correlation between num_orders and Numeric Features

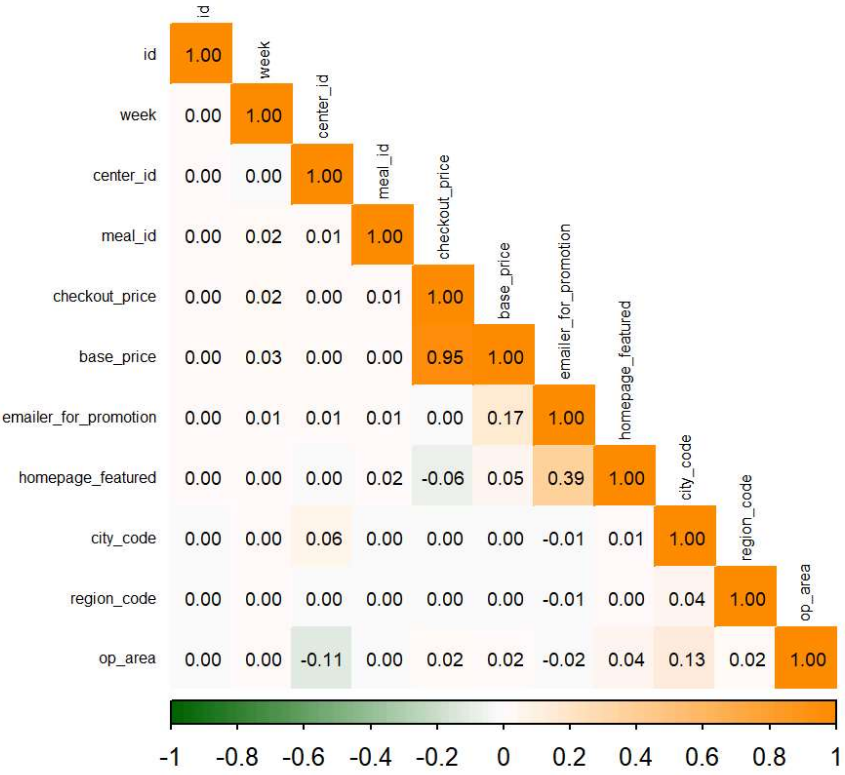
```r
correlations <- train %>%
  select(num_orders, where(is.numeric)) %>%
  cor(use = "complete.obs")

correlations_num_orders <- correlations["num_orders", -1]

correlations_table <- data.frame(
  Variable = names(correlations_num_orders),
  Correlation = round(as.numeric(correlations_num_orders), 3) # Pembulatan 3 desimal
)

print(correlations_table, row.names = FALSE)
```

```
             Variable Correlation
                   id       0.001
                 week      -0.007
            center_id      -0.051
              meal_id       0.012
       checkout_price      -0.283
           base_price      -0.221
emailer_for_promotion       0.283
    homepage_featured       0.299
            city_code       0.041
          region_code       0.030
              op_area       0.174
```

Korelasi antara variabel-variabel ini dengan `num_order` relatif rendah. **Tidak ada variabel yang memiliki korelasi yang sangat kuat** (> 0.5 atau < -0.5), yang menunjukkan bahwa tidak ada satu faktor dominan yang secara langsung menentukan jumlah pesanan.

Correlation Between Numeric Feature

```
numeric_vars <-
  train %>%
  select(-num_orders) %>%
  select(where(is.numeric))
correlation_matrix_pearson <- cor(numeric_vars, method = "pearson")
corrplot(correlation_matrix_pearson, method = "color", type = "lower",
         col = colorRampPalette(c("darkgreen", "white", "darkorange"))(200),
         tl.col = "black", tl.cex = 0.5,
         addCoef.col = "black", number.cex = 0.6, number.font = 0.5)
```

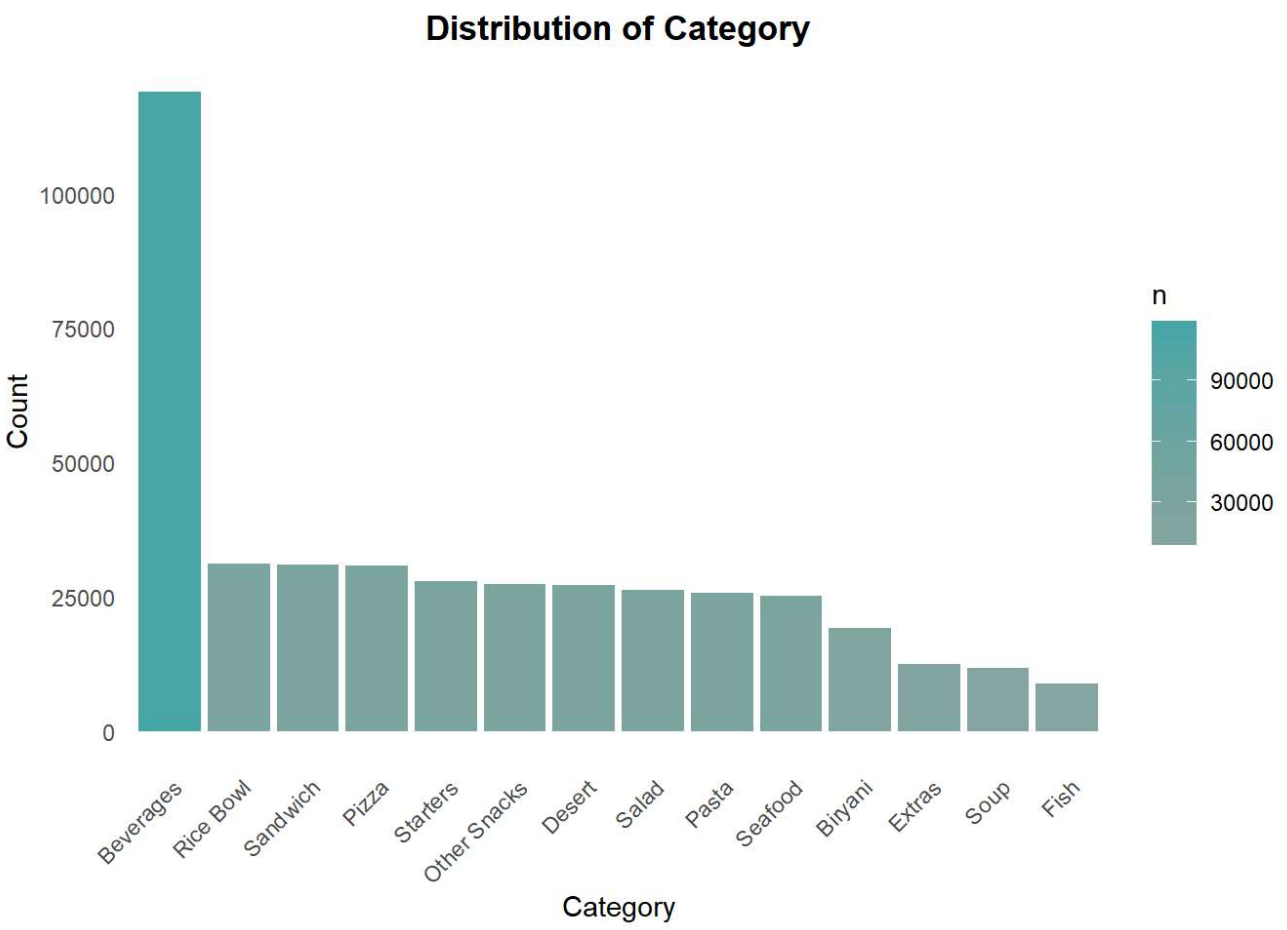


Plot heatmap menunjukkan bahwa s**ebagian besar variabel tidak memiliki korelasi kuat satu sama lain.** Variabel harga (`checkout_price` dan `base_price`) memiliki korelasi tinggi, yang menunjukkan **keterkaitan langsung antara harga dasar dan harga jual.**

Distribution of Category Feature

```
category_counts <- train %>%
  count(category) %>%
  arrange(desc(n))

ggplot(category_counts, aes(x = reorder(category, -n), y = n, fill = n)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  labs(title = "Distribution of Category",
```

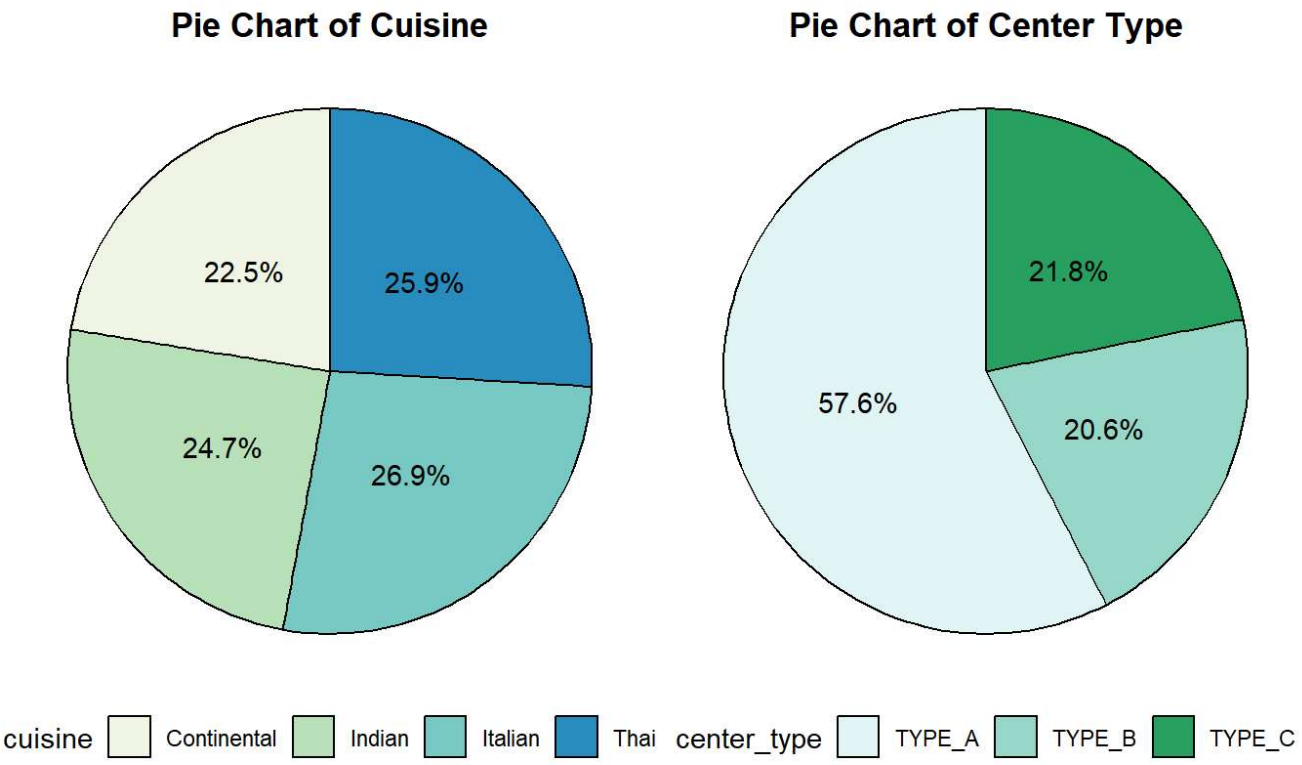
```
      x = "Category",
      y = "Count") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        panel.grid = element_blank(),
        panel.background = element_blank(),
        plot.title = element_text(hjust = 0.5, face = "bold")) +
  scale_fill_gradient(low = "#88A9A1", high = "#48A6A7")
```



Grafik ini menunjukkan distribusi jumlah pesanan berdasarkan kategori makanan dan minuman. Terlihat bahwa **kategori Beverages memiliki jumlah pesanan yang jauh lebih tinggi** dibandingkan kategori lainnya, dengan lebih dari 100.000 pesanan, menjadikannya produk paling populer dalam dataset ini.

```
category_counts <- train %>%
  count(cuisine) %>%
  mutate(percentage = n / sum(n) * 100)

p1 <- ggplot(category_counts, aes(x = "", y = n, fill = cuisine)) +
  geom_bar(stat = "identity", width = 1, color = "black") +  # Added border color
  coord_polar("y") +
  theme_void() +
  labs(title = "Pie Chart of Cuisine") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),
        legend.position = "bottom") +
  scale_fill_brewer(palette = "GnBu") +
  geom_text(aes(label = paste0(round(percentage, 1), "%")), position = position_stac
```

```r
category_counts <- train %>%
  count(center_type) %>%
  mutate(percentage = n / sum(n) * 100)

p2 <- ggplot(category_counts, aes(x = "", y = n, fill = center_type)) +
  geom_bar(stat = "identity", width = 1, color = "black") +  # Added border color
  coord_polar("y") +
  theme_void() +
  labs(title = "Pie Chart of Center Type") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),
        legend.position = "bottom") +
  scale_fill_brewer(palette = "BuGn") +
  geom_text(aes(label = paste0(round(percentage, 1), "%")), position = position_stac


grid.arrange(p1, p2, ncol = 2)
```



Pada pie chart dengan variabel cuisine, terlihat bahwa **masakan Italian memiliki proporsi terbesar**, yaitu 26.9%, diikuti oleh masakan Thai sebesar 25.9%. Sementara itu, masakan Indian mencakup 24.7% dari total, dan masakan Continental memiliki proporsi terkecil dengan 22.5%.

Pada pie chart dengan variabel center_type, distribusi pusat distribusi berdasarkan jenisnya menunjukkan bahwa TYPE_A mendominasi dengan 57.6% dari total pusat distribusi, sedangkan TYPE_B dan TYPE_C masing-masing memiliki proporsi 20.6% dan 21.8%. Hal ini menunjukkan bahwa **sebagian besar operasi distribusi dikelola oleh pusat dengan tipe A**, yang kemungkinan memiliki kapasitas lebih besar atau cakupan yang lebih luas dibandingkan tipe lainnya.

```r
category_orders <- train %>%
  group_by(category) %>%
  summarise(total_orders = mean(num_orders, na.rm = TRUE)) %>%
  arrange(desc(total_orders))

ggplot(category_orders, aes(x = reorder(category, -total_orders), y = total_orders,
  geom_bar(stat = "identity") +
  theme_minimal() +
  labs(title = "Mean Orders per Category",
       x = "Category",
       y = "Mean Num Orders") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        panel.grid = element_blank(),
        panel.background = element_blank(),
        plot.title = element_text(hjust = 0.5, face = "bold")) +
  scale_fill_gradient(low = "#88A9A1", high = "#48A6A7")
```

Pada bar plot dengan variabel `category`, kategori **Rice Bowl memiliki rata-rata pesanan tertinggi**, diikuti oleh Sandwich dan Salad, yang juga memiliki jumlah pesanan yang cukup tinggi dibandingkan kategori lainnya. Beverages dan Extras juga memiliki jumlah pesanan yang signifikan, meskipun lebih rendah dibandingkan tiga kategori teratas.

```r
p1 <- ggplot(train, aes(x = center_type, y = num_orders, fill = center_type)) +
  geom_boxplot(alpha = 0.7, outlier.color = "black", outlier.shape = 16) +
  theme_minimal() +
  labs(title = "Orders by Center Type",
```

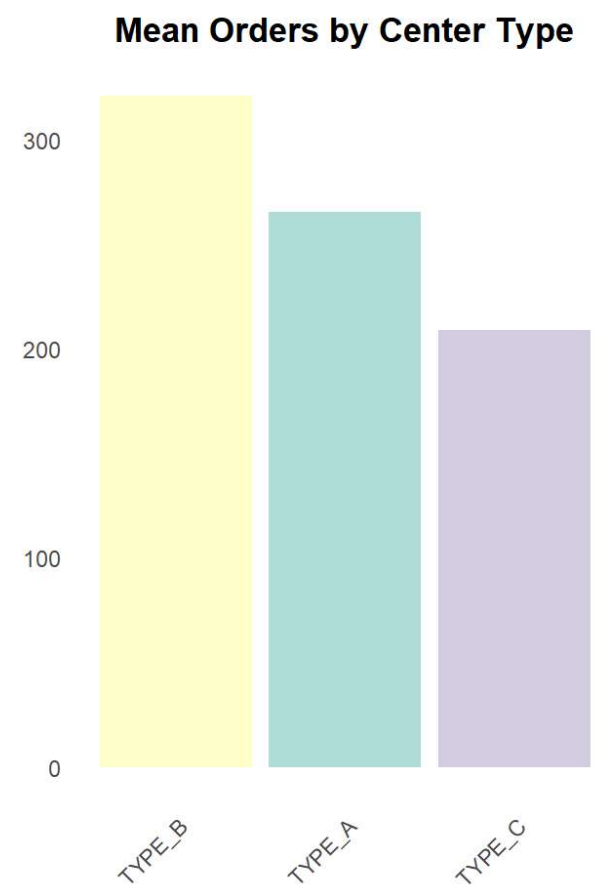
```
       x = "",
       y = "") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        panel.grid = element_blank(),
        panel.background = element_blank(),
        legend.position = "none",
        plot.title = element_text(hjust = 0.5, face = "bold")) +
  scale_fill_brewer(palette = "Set3")

p2 <- ggplot(train, aes(x = cuisine, y = num_orders, fill = cuisine)) +
  geom_boxplot(alpha = 0.7, outlier.color = "black", outlier.shape = 16) +
  theme_minimal() +
  labs(title = "Orders by Cuisine",
       x = "",
       y = "") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        panel.grid = element_blank(),
        panel.background = element_blank(),
        legend.position = "none",
        plot.title = element_text(hjust = 0.5, face = "bold")) +
  scale_fill_brewer(palette = "Set3")

grid.arrange(p1, p2, ncol = 2)
```

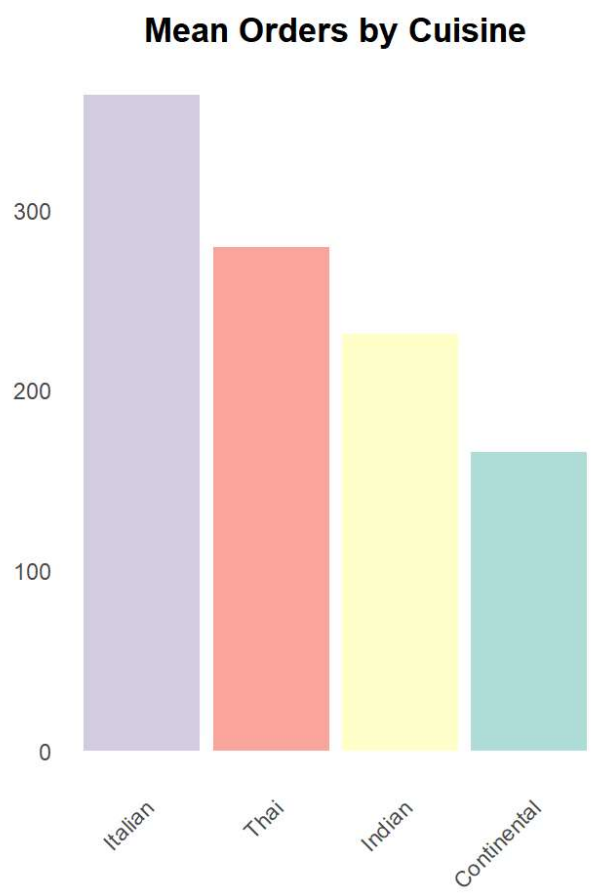
Pada Boxplot Orders by Center Type, terdapat tiga kategori pusat distribusi, yaitu TYPE_A, TYPE_B, dan TYPE_C. M**asing-masing kategori memiliki persebaran jumlah pesanan yang**

cukup mirip, dengan banyak outlier yang menunjukkan bahwa beberapa pusat distribusi memiliki pesanan yang jauh lebih tinggi dibandingkan mayoritas lainnya.

Pada Boxplot Orders by Cuisine, terdapat empat kategori masakan yaitu Continental, Indian, Italian, dan Thai. **Pola persebarannya juga menunjukkan bahwa sebagian besar pesanan berada di kisaran rendah**, namun terdapat beberapa outlier dengan jumlah pesanan yang sangat tinggi, terutama pada masakan Indian dan Italian.

```
center_orders <- train %>%
  group_by(center_type) %>%
  summarise(total_orders = mean(num_orders, na.rm = TRUE)) %>%
  arrange(desc(total_orders))

p1 <- ggplot(center_orders, aes(x = reorder(center_type, -total_orders), y = total_o
  geom_bar(stat = "identity", alpha = 0.7) +
  theme_minimal() +
  labs(title = "Mean Orders by Center Type",
       x = "",
       y = "") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        panel.grid = element_blank(),
        panel.background = element_blank(),
        legend.position = "none",
        plot.title = element_text(hjust = 0.5, face = "bold")) +
  scale_fill_brewer(palette = "Set3")

cuisine_data <- train %>%
  group_by(cuisine) %>%
  summarise(total_orders = mean(num_orders, na.rm = TRUE)) %>%
  arrange(desc(total_orders))

p2 <- ggplot(cuisine_data, aes(x = reorder(cuisine, -total_orders), y = total_orders
  geom_bar(stat = "identity", alpha = 0.7) +
  theme_minimal() +
  labs(title = "Mean Orders by Cuisine",
       x = "",
       y = "") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        panel.grid = element_blank(),
        panel.background = element_blank(),
        legend.position = "none",
        plot.title = element_text(hjust = 0.5, face = "bold")) +
  scale_fill_brewer(palette = "Set3")

grid.arrange(p1, p2, ncol = 2)
```





Pada barplot Mean Orders by Center Type, menunjukkan bahwa **pusat distribusi TYPE_B memiliki rata-rata jumlah pesanan tertinggi**, diikuti oleh TYPE_A, sementara TYPE_C memiliki rata-rata terendah. Hal ini dapat mengindikasikan bahwa pusat distribusi TYPE_B lebih dominan dalam pemenuhan pesanan dibandingkan dua tipe lainnya.

Pada barplot Mean Orders by Cuisine mengilustrasikan bahwa **masakan Italian memiliki rata-rata pesanan tertinggi**, diikuti oleh Thai dan Indian, sedangkan Continental memiliki rata-rata pesanan terendah. Ini menunjukkan bahwa masakan Italian lebih populer atau memiliki permintaan yang lebih tinggi dibandingkan jenis masakan lainnya.

Pre-Modelling

Split Train - Test Data

Dataset train menjadi **data latih (80%) dan data uji (20%)** secara stratifikasi berdasarkan `id`. Fungsi set.seed(50) memastikan pemilihan data tetap konsisten, sementara createDataPartition() menghasilkan indeks untuk data latih. Indeks ini kemudian digunakan untuk mengekstrak data latih dan data uji dari dataset asli.

```
set.seed(50)
train_indices <- createDataPartition(train$id, p = 0.8, list = FALSE)
train_data <- train[train_indices, ]
test_data <- train[-train_indices, ]
```

Declare Target and Feature

Selanjutnya dilakukan pemisahan fitur dan target dalam dataset pelatihan dan pengujian. Setelah itu, data dipisahkan menjadi variabel yang berisi fitur dan variabel yang berisi target sehingga siap digunakan untuk proses pemodelan.

```
features <- setdiff(names(train), c("id", "num_orders"))
X_train <- train_data[, features, with = FALSE]
y_train <- train_data$num_orders
X_test <- test_data[, features, with = FALSE]
y_test <- test_data$num_orders
```

One-Hot Encoding

Data diubah menjadi format data.table, kemudian fitur kategorikal diidentifikasi dan dikonversi menjadi faktor. Selanjutnya, dilakukan **One-Hot Encoding untuk mengonversi kategori menjadi variabel biner.** Setelah itu, semua kolom dipastikan dalam bentuk numerik untuk melakukan pemodelan machine learning. Terakhir, dilakukan pengecekan untuk memastikan tidak ada nilai yang hilang (NA) dalam data yang telah diproses.

```
# Memastikan X_train dan X_test adalah data.table
setDT(X_train)
setDT(X_test)

# Identifikasi fitur kategorikal yang valid
categorical_features <- names(X_train)[sapply(X_train, is.character)]
categorical_features <- intersect(names(X_train), categorical_features)

# Konversi fitur kategorikal menjadi faktor
X_train[, (categorical_features) := lapply(.SD, as.factor), .SDcols = categorical_fe
X_test[, (categorical_features) := lapply(.SD, as.factor), .SDcols = categorical_fea

# Pastikan categorical_features hanya berisi fitur yang ada di X_train_final
setdiff(categorical_features, names(X_train))
```

```
character(0)
```

```
# One-Hot Encoding untuk fitur kategorikal pada X_train dan X_test
X_train_encoded <- dummy_cols(X_train, select_columns = categorical_features, remove
X_test_encoded <- dummy_cols(X_test, select_columns = categorical_features, remove_f

# Pisahkan kembali target (num_orders)
X_train_final <- X_train_encoded[, setdiff(names(X_train_encoded), "num_orders"), wi
X_test_final <- X_test_encoded[, setdiff(names(X_test_encoded), "num_orders"), with

# Pastikan semua kolom dalam data adalah numerik setelah encoding
X_train_final <- data.matrix(X_train_final)
X_test_final <- data.matrix(X_test_final)

# Konfirmasi bahwa tidak ada NA di data
sum(is.na(X_train_final))
```

```
[1] 0
```

```
sum(is.na(X_test_final))
```

```
[1] 0
```

Modelling

1. LightGBM

```
# Membuat objek dataset LightGBM
dtrain <- lgb.Dataset(
  data = as.matrix(X_train_final),
  label = y_train
)

dtest <- lgb.Dataset(
  data = as.matrix(X_test_final),
  label = y_test,
  reference = dtrain
)

# Parameter LightGBM
params <- list(
  objective = "regression",
  metric = "rmse",
  boosting = "gbdt",
  learning_rate = 0.05,
  num_leaves = 64,
  max_depth = -1,
  feature_fraction = 0.7,
  bagging_fraction = 0.7,
  bagging_freq = 5
)


# Cross Validation
cv_results <- lgb.cv(
  params = params,
  data = dtrain,
  nfold = 5,
  nrounds = 100,
  early_stopping_rounds = 10,
  verbose = 1
)
```

```
[LightGBM] [Warning] Found whitespace in feature_names, replace with underlines
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing
was 0.020833 seconds.
You can set `force_row_wise=true` to remove the overhead.
```

```
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 919
[LightGBM] [Info] Number of data points in the train set: 271187, number of used features: 31
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.020041 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 919
[LightGBM] [Info] Number of data points in the train set: 271187, number of used features: 31
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.019818 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 919
[LightGBM] [Info] Number of data points in the train set: 271186, number of used features: 31
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.019232 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 919
[LightGBM] [Info] Number of data points in the train set: 271186, number of used features: 31
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.042820 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 919
[LightGBM] [Info] Number of data points in the train set: 271186, number of used features: 31
[LightGBM] [Info] Start training from score 264.684705
[LightGBM] [Info] Start training from score 264.613676
[LightGBM] [Info] Start training from score 264.364215
[LightGBM] [Info] Start training from score 264.807733
[LightGBM] [Info] Start training from score 263.772481
[1]:  valid's rmse:390.846+6.23428
Will train until there is no improvement in 10 rounds.
[2]:  valid's rmse:379.851+6.22356
[3]:  valid's rmse:369.722+6.249
[4]:  valid's rmse:360.238+6.19362
[5]:  valid's rmse:351.496+6.2139
[6]:  valid's rmse:342.402+6.22568
[7]:  valid's rmse:334.046+6.33414
[8]:  valid's rmse:326.142+6.30951
[9]:  valid's rmse:318.698+6.39456
[10]:  valid's rmse:312.472+6.31648
[11]:  valid's rmse:306.478+6.37725
[12]:  valid's rmse:300.717+6.37678
[13]:  valid's rmse:295.093+6.44218
[14]:  valid's rmse:290.26+6.56096
[15]:  valid's rmse:284.78+6.6035
[16]:  valid's rmse:279.486+6.65879
```

```
[17]:  valid's rmse:274.832+6.49749
[18]:  valid's rmse:270.6+6.50914
[19]:  valid's rmse:266.342+6.55221
[20]:  valid's rmse:262.336+6.49489
[21]:  valid's rmse:259.113+6.50224
[22]:  valid's rmse:255.385+6.51497
[23]:  valid's rmse:251.893+6.52468
[24]:  valid's rmse:248.539+6.57214
[25]:  valid's rmse:245.664+6.56654
[26]:  valid's rmse:242.707+6.54367
[27]:  valid's rmse:240.077+6.5047
[28]:  valid's rmse:237.44+6.49781
[29]:  valid's rmse:234.784+6.39002
[30]:  valid's rmse:232.807+6.34433
[31]:  valid's rmse:230.937+6.36885
[32]:  valid's rmse:228.85+6.34512
[33]:  valid's rmse:226.984+6.33968
[34]:  valid's rmse:225.178+6.32908
[35]:  valid's rmse:223.573+6.27222
[36]:  valid's rmse:221.943+6.13831
[37]:  valid's rmse:220.304+6.04201
[38]:  valid's rmse:218.79+5.94865
[39]:  valid's rmse:217.23+5.92936
[40]:  valid's rmse:215.84+5.94019
[41]:  valid's rmse:214.507+5.8973
[42]:  valid's rmse:213.259+5.8471
[43]:  valid's rmse:211.997+5.76445
[44]:  valid's rmse:210.715+5.78671
[45]:  valid's rmse:209.592+5.72487
[46]:  valid's rmse:208.365+5.65499
[47]:  valid's rmse:207.354+5.65386
[48]:  valid's rmse:206.282+5.62264
[49]:  valid's rmse:205.064+5.56085
[50]:  valid's rmse:204.096+5.56672
[51]:  valid's rmse:203.135+5.49834
[52]:  valid's rmse:202.282+5.48091
[53]:  valid's rmse:201.489+5.4486
[54]:  valid's rmse:200.561+5.41533
[55]:  valid's rmse:199.732+5.34945
[56]:  valid's rmse:199.035+5.3323
[57]:  valid's rmse:198.292+5.29123
[58]:  valid's rmse:197.491+5.24862
[59]:  valid's rmse:196.663+5.23297
[60]:  valid's rmse:195.971+5.24093
[61]:  valid's rmse:195.212+5.17887
[62]:  valid's rmse:194.528+5.13589
[63]:  valid's rmse:193.88+5.05642
[64]:  valid's rmse:193.277+5.08905
[65]:  valid's rmse:192.602+5.08461
[66]:  valid's rmse:191.921+5.08642
[67]:  valid's rmse:191.319+5.04043
[68]:  valid's rmse:190.809+5.04189
[69]:  valid's rmse:190.199+5.02801
```

```
[70]:  valid's rmse:189.536+5.02103
[71]:  valid's rmse:189.025+5.01637
[72]:  valid's rmse:188.39+4.96903
[73]:  valid's rmse:187.693+4.94283
[74]:  valid's rmse:187.027+4.87848
[75]:  valid's rmse:186.401+4.78234
[76]:  valid's rmse:185.83+4.75189
[77]:  valid's rmse:185.354+4.76873
[78]:  valid's rmse:184.963+4.78882
[79]:  valid's rmse:184.45+4.73694
[80]:  valid's rmse:184.052+4.76835
[81]:  valid's rmse:183.583+4.77042
[82]:  valid's rmse:183.204+4.76529
[83]:  valid's rmse:182.737+4.74739
[84]:  valid's rmse:182.307+4.77158
[85]:  valid's rmse:181.788+4.78656
[86]:  valid's rmse:181.408+4.77088
[87]:  valid's rmse:181.07+4.71405
[88]:  valid's rmse:180.663+4.69683
[89]:  valid's rmse:180.309+4.71602
[90]:  valid's rmse:179.969+4.70809
[91]:  valid's rmse:179.486+4.66583
[92]:  valid's rmse:179.101+4.63248
[93]:  valid's rmse:178.741+4.63157
[94]:  valid's rmse:178.377+4.62489
[95]:  valid's rmse:178.011+4.62891
[96]:  valid's rmse:177.676+4.59772
[97]:  valid's rmse:177.287+4.59944
[98]:  valid's rmse:177.019+4.55841
[99]:  valid's rmse:176.67+4.53959
[100]:  valid's rmse:176.424+4.54757
Did not meet early stopping, best iteration is: [100]:  valid's rmse:176.424+4.54757
```

```
# Melatih model akhir dengan iterasi terbaik dari CV
model_lgb <- lgb.train(
  params = params,
  data = dtrain,
  nrounds = cv_results$best_iter
)
```

```
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing
was 0.013285 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 919
[LightGBM] [Info] Number of data points in the train set: 338983, number of used
features: 31
[LightGBM] [Info] Start training from score 264.448562
```

```
# Model telah dilatih
model_lgb
```

```
LightGBM Model (100 trees)
Objective: regression
Fitted to dataset with 31 columns
```

Model LightGBM telah dilatih dengan **100 pohon dan 31 fitur** tanpa memenuhi kondisi early stopping, yang berarti model berjalan hingga jumlah maksimum iterasi yang ditentukan. Model ini Tidak adanya early stopping menunjukkan bahwa jumlah pohon yang digunakan mungkin masih dapat dioptimalkan lebih lanjut.

```r
# Melakukan prediksi pada data uji
preds_lgb <- predict(model_lgb, as.matrix(X_test_final))

# Menghitung RMSE
rmse <- sqrt(mean((preds_lgb - y_test)^2))
cat("Root Mean Squared Error (RMSE):", rmse, "\n")
```

```
Root Mean Squared Error (RMSE): 176.3715
```

```r
# Menghitung R-squared (R²)
sst <- sum((y_test - mean(y_test))^2)
sse <- sum((y_test - preds_lgb)^2)
r2_lgb <- 1 - (sse / sst)
cat("R-squared (R²):", r2_lgb, "\n")
```

```
R-squared (R²): 0.811893
```

Nilai R-squared (R^2) sebesar 0.8119 menunjukkan bahwa model LightGBM mampu menjelaskan sekitar **81.19% variasi dalam data target**, sementara sisanya, sekitar 18.81%, tidak terjelaskan oleh model. Hal ini mengindikasikan bahwa model memiliki performa yang cukup baik, meskipun masih dapat dilakukan perbaikan.

2. XGBoost

```r
# Membuat objek DMatrix untuk XGBoost
dtrain <- xgb.DMatrix(
  data = as.matrix(X_train_final),  # Gunakan data yang sudah di-encode
  label = y_train
)

dtest <- xgb.DMatrix(
  data = as.matrix(X_test_final),  # Gunakan data yang sudah di-encode
  label = y_test
)

# Parameter XGBoost
params <- list(
  objective = "reg:squarederror",
  eval_metric = "rmse",
  booster = "gbtree",
  eta = 0.05,
  max_depth = 8,
```

```
  min_child_weight = 10,
  gamma = 0.1,
  lambda = 1,
  alpha = 0.5,
  subsample = 0.7,
  colsample_bytree = 0.7,
  nthread = 4,
  verbosity = 1
)

# Cross Validation
cv_results <- xgb.cv(
  params = params,
  data = dtrain,
  nfold = 5,
  nrounds = 100,
  early_stopping_rounds = 10,
  verbose = 1
)
```

```
[1] train-rmse:464.513296+1.849680  test-rmse:464.616822+4.956429
Multiple eval metrics are present. Will use test_rmse for early stopping.
Will train until test_rmse hasn't improved in 10 rounds.

[2] train-rmse:448.365790+1.913375  test-rmse:448.644986+4.772332
[3] train-rmse:433.212782+1.803278  test-rmse:433.646620+4.860895
[4] train-rmse:418.878171+2.220904  test-rmse:419.555634+4.303489
[5] train-rmse:405.026606+2.278084  test-rmse:405.885394+4.149921
[6] train-rmse:392.446416+2.513721  test-rmse:393.402832+3.887075
[7] train-rmse:379.947880+2.682017  test-rmse:381.128752+3.694712
[8] train-rmse:368.668824+3.085693  test-rmse:369.961463+3.527113
[9] train-rmse:357.963252+2.785216  test-rmse:359.441510+3.909315
[10]    train-rmse:347.919071+2.437392  test-rmse:349.562103+4.279738
[11]    train-rmse:338.323825+2.084211  test-rmse:340.122200+4.373432
[12]    train-rmse:328.962338+1.995807  test-rmse:330.920740+4.358936
[13]    train-rmse:320.494503+1.693035  test-rmse:322.730217+4.516905
[14]    train-rmse:312.177857+1.491912  test-rmse:314.614301+4.731979
[15]    train-rmse:304.679289+1.496704  test-rmse:307.332805+4.701510
[16]    train-rmse:297.356221+1.286847  test-rmse:300.363615+4.877911
[17]    train-rmse:290.662739+1.178105  test-rmse:293.832031+4.927845
[18]    train-rmse:284.205809+0.951451  test-rmse:287.602455+5.123078
[19]    train-rmse:278.090557+0.998564  test-rmse:281.704190+5.053966
[20]    train-rmse:272.279534+1.135456  test-rmse:276.106705+4.993313
[21]    train-rmse:267.177031+1.321770  test-rmse:271.157273+5.127918
[22]    train-rmse:262.287611+1.033163  test-rmse:266.425382+5.151945
[23]    train-rmse:257.397663+1.186429  test-rmse:261.726752+5.009993
[24]    train-rmse:252.630792+1.021912  test-rmse:257.157306+5.168086
[25]    train-rmse:248.339958+0.985618  test-rmse:253.151340+5.282543
[26]    train-rmse:244.254818+0.896690  test-rmse:249.245613+5.241142
[27]    train-rmse:240.311814+1.084367  test-rmse:245.531130+5.238039
[28]    train-rmse:236.785306+1.173059  test-rmse:242.190415+5.013784
[29]    train-rmse:233.485370+1.210233  test-rmse:239.135642+5.059750
[30]    train-rmse:230.320352+1.199424  test-rmse:236.184857+4.961613
```

```
[31]    train-rmse:227.232227+1.078329  test-rmse:233.232894+5.065517
[32]    train-rmse:224.165311+1.053338  test-rmse:230.457014+5.080380
[33]    train-rmse:221.278479+1.315476  test-rmse:227.769594+5.093058
[34]    train-rmse:218.795281+1.291260  test-rmse:225.403664+5.003222
[35]    train-rmse:216.199553+1.379623  test-rmse:223.037834+4.954419
[36]    train-rmse:213.851868+1.317055  test-rmse:220.877256+5.013762
[37]    train-rmse:211.554855+1.316034  test-rmse:218.721887+4.978302
[38]    train-rmse:209.493718+1.407923  test-rmse:216.816753+4.921657
[39]    train-rmse:207.507823+1.558801  test-rmse:215.000907+4.842650
[40]    train-rmse:205.280147+1.537344  test-rmse:212.920779+4.947250
[41]    train-rmse:203.455795+1.514970  test-rmse:211.263746+4.822219
[42]    train-rmse:201.462594+1.550080  test-rmse:209.456490+4.648538
[43]    train-rmse:199.668767+1.392436  test-rmse:207.832453+4.613104
[44]    train-rmse:197.995209+1.331615  test-rmse:206.360890+4.462095
[45]    train-rmse:196.432960+1.262649  test-rmse:204.984530+4.378380
[46]    train-rmse:194.819615+1.197164  test-rmse:203.561753+4.281108
[47]    train-rmse:193.391398+1.208788  test-rmse:202.269402+4.271390
[48]    train-rmse:191.840518+1.195538  test-rmse:200.909355+4.353289
[49]    train-rmse:190.313565+1.186384  test-rmse:199.506042+4.270251
[50]    train-rmse:188.823426+1.039798  test-rmse:198.181318+4.242385
[51]    train-rmse:187.369717+0.961009  test-rmse:196.919557+4.339931
[52]    train-rmse:186.210558+0.908712  test-rmse:195.871205+4.497250
[53]    train-rmse:184.978868+0.960962  test-rmse:194.760725+4.362202
[54]    train-rmse:183.795463+0.988969  test-rmse:193.649523+4.297970
[55]    train-rmse:182.623460+1.099221  test-rmse:192.647492+4.196651
[56]    train-rmse:181.514337+1.073502  test-rmse:191.707736+4.067365
[57]    train-rmse:180.447226+1.054960  test-rmse:190.790084+4.055163
[58]    train-rmse:179.463677+1.043560  test-rmse:189.974532+4.027744
[59]    train-rmse:178.498084+0.978515  test-rmse:189.141486+3.958348
[60]    train-rmse:177.590610+0.982207  test-rmse:188.340411+3.875676
[61]    train-rmse:176.727985+0.856006  test-rmse:187.582946+3.908867
[62]    train-rmse:175.816784+0.808538  test-rmse:186.797385+4.007336
[63]    train-rmse:175.006758+0.866014  test-rmse:186.088926+3.949612
[64]    train-rmse:174.129088+0.924652  test-rmse:185.302528+3.950246
[65]    train-rmse:173.318417+0.942568  test-rmse:184.566978+3.990659
[66]    train-rmse:172.521668+0.799820  test-rmse:183.835695+3.954610
[67]    train-rmse:171.824710+0.855066  test-rmse:183.237996+3.932258
[68]    train-rmse:171.139679+0.810282  test-rmse:182.683083+3.909787
[69]    train-rmse:170.360842+0.685795  test-rmse:182.005246+3.938008
[70]    train-rmse:169.726660+0.712393  test-rmse:181.434413+3.893829
[71]    train-rmse:168.968988+0.593071  test-rmse:180.821204+3.866955
[72]    train-rmse:168.368444+0.561655  test-rmse:180.336941+3.886846
[73]    train-rmse:167.689517+0.582788  test-rmse:179.804293+3.924481
[74]    train-rmse:167.138972+0.627579  test-rmse:179.324723+3.857740
[75]    train-rmse:166.419934+0.629068  test-rmse:178.690049+3.784853
[76]    train-rmse:165.838111+0.698128  test-rmse:178.189007+3.737814
[77]    train-rmse:165.223918+0.719624  test-rmse:177.691807+3.746566
[78]    train-rmse:164.641849+0.741722  test-rmse:177.184425+3.685816
[79]    train-rmse:164.020905+0.744081  test-rmse:176.652494+3.654828
[80]    train-rmse:163.496841+0.660682  test-rmse:176.212728+3.638099
[81]    train-rmse:162.975541+0.657509  test-rmse:175.771142+3.576163
[82]    train-rmse:162.406152+0.637312  test-rmse:175.323341+3.629867
[83]    train-rmse:161.916592+0.597995  test-rmse:174.935604+3.611458
```

```
[84]    train-rmse:161.347874+0.580857  test-rmse:174.498878+3.599330
[85]    train-rmse:160.862396+0.529242  test-rmse:174.135591+3.713987
[86]    train-rmse:160.371550+0.518545  test-rmse:173.757264+3.663779
[87]    train-rmse:159.909224+0.549395  test-rmse:173.368333+3.666589
[88]    train-rmse:159.449213+0.537031  test-rmse:172.976397+3.679832
[89]    train-rmse:159.048907+0.568203  test-rmse:172.623434+3.658107
[90]    train-rmse:158.636583+0.521294  test-rmse:172.294486+3.714323
[91]    train-rmse:158.235448+0.487942  test-rmse:171.929491+3.673596
[92]    train-rmse:157.815210+0.479873  test-rmse:171.600221+3.709017
[93]    train-rmse:157.340470+0.470235  test-rmse:171.223268+3.709595
[94]    train-rmse:156.905559+0.456237  test-rmse:170.875527+3.658445
[95]    train-rmse:156.459275+0.458994  test-rmse:170.498649+3.620796
[96]    train-rmse:156.061109+0.430097  test-rmse:170.187068+3.666730
[97]    train-rmse:155.759440+0.364852  test-rmse:169.940598+3.632335
[98]    train-rmse:155.398975+0.429991  test-rmse:169.638319+3.634854
[99]    train-rmse:154.986496+0.423236  test-rmse:169.296394+3.635458
[100]   train-rmse:154.661982+0.462658  test-rmse:169.020780+3.613838
```

```r
# Melatih model akhir dengan iterasi terbaik dari CV
model_xgb <- xgb.train(
  params = params,
  data = dtrain,
  nrounds = cv_results$best_iteration
)

# Model telah dilatih
model_xgb
```

```
##### xgb.Booster
raw: 1.4 Mb
call:
  xgb.train(params = params, data = dtrain, nrounds = cv_results$best_iteration)
params (as set within xgb.train):
  objective = "reg:squarederror", eval_metric = "rmse", booster = "gbtree", eta =
"0.05", max_depth = "8", min_child_weight = "10", gamma = "0.1", lambda = "1", alpha
= "0.5", subsample = "0.7", colsample_bytree = "0.7", nthread = "4", verbosity =
"1", validate_parameters = "TRUE"
xgb.attributes:
  niter
callbacks:
  cb.print.evaluation(period = print_every_n)
# of features: 31
niter: 100
nfeatures : 31
```

Model XGBoost telah dilatih menggunakan **100 iterasi dengan 31 fitur** menggunakan parameter yang telah ditentukan. Model ini menggunakan metrik evaluasi rmse untuk regresi.

```r
# Melakukan prediksi pada data uji
preds_xgb <- predict(model_xgb, as.matrix(X_test_final))

# Menghitung RMSE
```

```r
rmse <- sqrt(mean((preds_xgb - y_test)^2))
cat("Root Mean Squared Error (RMSE):", rmse, "\n")
```

```
Root Mean Squared Error (RMSE): 165.4861
```

```r
# Menghitung R-squared (R²)
sst <- sum((y_test - mean(y_test))^2)
sse <- sum((y_test - preds_xgb)^2)
r2_xgb <- 1 - (sse / sst)
cat("R-squared (R²):", r2_xgb, "\n")
```

```
R-squared (R²): 0.8343959
```

Model XGBoost menghasilkan nilai R-squared (R²) sebesar 0.8344 yang menunjukkan bahwa model dapat menjelaskan sekitar **83.44% variasi dalam data target**, sementara sisanya sekitar 16.56% tidak dapat dijelaskan oleh model.

3. Random Forest

```r
# Membuat objek data frame untuk Random Forest
dtrain <- data.frame(X_train_final, y_train)
dtest <- data.frame(X_test_final, y_test)

# Melatih model Random Forest
model_rf <- randomForest(
  x = X_train_final,
  y = y_train,
  ntree = 400,
  mtry = sqrt(ncol(X_train_final)),
  importance = TRUE
)

# Model telah dilatih
model_rf
```

```
Call:
 randomForest(x = X_train_final, y = y_train, ntree = 400, mtry =
sqrt(ncol(X_train_final)),      importance = TRUE)
               Type of random forest: regression
                     Number of trees: 400
No. of variables tried at each split: 6

          Mean of squared residuals: 28608.46
                    % Var explained: 82.37
```

```r
# Melakukan prediksi pada data uji
preds_rf <- predict(model_rf, X_test_final)

# Menghitung RMSE
```

```
rmse <- sqrt(mean((preds_rf - y_test)^2))
cat("Root Mean Squared Error (RMSE):", rmse, "\n")
```

```
Root Mean Squared Error (RMSE): 169.5341
```

```
# Menghitung R-squared (R²)
sst <- sum((y_test - mean(y_test))^2)
sse <- sum((y_test - preds_rf)^2)
r2_rf <- 1 - (sse / sst)
cat("R-squared (R²):", r2_rf, "\n")
```

```
R-squared (R²): 0.826195
```

Model Random Forest menunjukkan kinerja yang cukup baik dengan R^2 sebesar 0.8262, yang berarti model dapat menjelaskan **82.62% variasi dalam data target.** Namun, RMSE sebesar 169.53 menunjukkan bahwa rata-rata kesalahan prediksi masih cukup besar, yang mengindikasikan peluang untuk peningkatan model.

4. Ensemble - Averaging

```
# Menormalisasi bobot agar totalnya menjadi 1
total_r2 <- r2_lgb + r2_xgb + r2_rf
w_lgb <- r2_lgb / total_r2
w_xgb <- r2_xgb / total_r2
w_rf <- r2_rf / total_r2

# Weighted Ensemble Prediction
pred_ensemble <- (w_lgb * preds_lgb) + (w_xgb * preds_xgb) + (w_rf * preds_rf)

# Menghitung RMSE
rmse_ens <- sqrt(mean((pred_ensemble - y_test)^2))
cat("Root Mean Squared Error (RMSE):", rmse_ens, "\n")
```

```
Root Mean Squared Error (RMSE): 167.4428
```

```
# Menghitung R-squared (R²)
sst <- sum((y_test - mean(y_test))^2)
sse <- sum((y_test - pred_ensemble)^2)
r2_ens <- 1 - (sse / sst)
cat("R-squared (R²):", r2_ens, "\n")
```

```
R-squared (R²): 0.8304566
```

Hasil ensemble averaging menunjukkan peningkatan performa dibandingkan model individual. Dengan RMSE sebesar 169.53, model ensemble memiliki kesalahan rata-rata yang lebih kecil dibandingkan model sebelumya, yang menunjukkan peningkatan akurasi prediksi. Selain itu, R^2 sebesar 0.8305 menunjukkan bahwa model ini mampu menjelaskan **83.44% variasi dalam data target. Peningkatan ini membuktikan bahwa menggabungkan LightGBM, XGBoost, dan Random Forest** dengan evaluasi berbasis R^2 dapat menghasilkan prediksi yang lebih stabil dan akurat.

5. Stacking

```r
# Gabungkan prediksi sebagai fitur baru
stack_train <- data.frame(
  y_train,
  preds_lgb = predict(model_lgb, as.matrix(X_train_final)),
  preds_xgb = predict(model_xgb, as.matrix(X_train_final)),
  preds_rf  = predict(model_rf, X_train_final)
)

stack_test <- data.frame(
  y_test,
  preds_lgb = predict(model_lgb, as.matrix(X_test_final)),
  preds_xgb = predict(model_xgb, as.matrix(X_test_final)),
  preds_rf  = predict(model_rf, X_test_final)
)

# Meta-model (Linear Regression atau Ridge)
meta_model <- train(y_train ~ ., data = stack_train,
                    method = "gbm",
                    trControl = trainControl(method = "cv", number = 5))
```

```
Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1   148741.3689             nan     0.1000 14505.3721
     2   136548.2117             nan     0.1000 12151.2139
     3   125698.8311             nan     0.1000 10903.0938
     4   116357.7788             nan     0.1000 9271.6204
     5   107975.5577             nan     0.1000 8386.3445
     6   100517.9403             nan     0.1000 7514.8266
     7    93919.7109             nan     0.1000 6604.2057
     8    88154.8213             nan     0.1000 5832.8977
     9    82242.0270             nan     0.1000 5864.4042
    10    77586.0462             nan     0.1000 4690.1611
    20    46132.2034             nan     0.1000 1902.8617
    40    24620.9731             nan     0.1000  501.2343
    60    18774.9283             nan     0.1000  134.2240
    80    17094.9072             nan     0.1000   33.2973
   100    16595.4951             nan     0.1000   -9.9691
   120    16429.4871             nan     0.1000  -40.7600
   140    16322.2846             nan     0.1000   -0.2862
   150    16290.6605             nan     0.1000  -35.1968

Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1   143972.4061             nan     0.1000 18649.2757
     2   127123.3031             nan     0.1000 16507.2058
     3   112277.8476             nan     0.1000 14899.5952
     4    99503.5831             nan     0.1000 12819.3102
     5    89408.9603             nan     0.1000 10100.6422
     6    80096.2664             nan     0.1000 9265.3749
     7    72225.8232             nan     0.1000 7746.8600
     8    65180.8595             nan     0.1000 7284.5512
     9    59559.9665             nan     0.1000 5600.2604
```

```
     10    54547.3754             nan     0.1000 5117.2776
     20    27807.3853             nan     0.1000 1048.2589
     40    17475.8271             nan     0.1000  131.6968
     60    16430.8752             nan     0.1000  -27.6551
     80    16125.2962             nan     0.1000  -25.0594
    100    15928.7979             nan     0.1000  -29.0435
    120    15747.0761             nan     0.1000   11.8454
    140    15624.2643             nan     0.1000  -24.7120
    150    15576.6663             nan     0.1000   -2.7454

Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1   140386.0770             nan     0.1000 23031.3640
     2   121768.5851             nan     0.1000 18907.8258
     3   104955.7413             nan     0.1000 16357.4235
     4    91697.3427             nan     0.1000 13363.1655
     5    80129.3490             nan     0.1000 11087.9504
     6    70795.9444             nan     0.1000 9516.6277
     7    62636.2288             nan     0.1000 8173.9257
     8    55990.2995             nan     0.1000 6721.5405
     9    50055.5424             nan     0.1000 6187.1921
    10    44979.8955             nan     0.1000 4988.4393
    20    22483.5978             nan     0.1000  957.6074
    40    16646.6449             nan     0.1000   26.3235
    60    15971.9673             nan     0.1000   -1.0045
    80    15609.5711             nan     0.1000   11.3103
   100    15396.9491             nan     0.1000  -10.5677
   120    15163.5913             nan     0.1000   -7.3532
   140    14988.2259             nan     0.1000  -26.8412
   150    14903.8471             nan     0.1000  -19.8626

Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1   145203.7494             nan     0.1000 14080.5493
     2   133014.2072             nan     0.1000 12148.6321
     3   122663.8017             nan     0.1000 10266.0526
     4   113128.7498             nan     0.1000 9269.1967
     5   105156.3289             nan     0.1000 7910.4143
     6    97655.1912             nan     0.1000 7511.5265
     7    91412.1090             nan     0.1000 6312.5619
     8    85454.9625             nan     0.1000 5927.0438
     9    79949.8091             nan     0.1000 5492.4817
    10    75115.2002             nan     0.1000 4660.4652
    20    44352.3222             nan     0.1000 2228.2883
    40    23369.6006             nan     0.1000  441.0335
    60    18015.5357             nan     0.1000  121.5151
    80    16452.4548             nan     0.1000   44.1906
   100    15969.7844             nan     0.1000   -3.0497
   120    15821.6775             nan     0.1000    2.7550
   140    15717.7708             nan     0.1000    0.9573
   150    15677.7350             nan     0.1000    2.3663

Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1   139908.0033             nan     0.1000 19684.5991
     2   123692.5669             nan     0.1000 16349.9920
```

```
      3   109675.0277             nan     0.1000 13824.0161
      4    97531.7905             nan     0.1000 12137.2454
      5    87217.8304             nan     0.1000 10212.0447
      6    77994.7861             nan     0.1000 9399.3484
      7    70023.5210             nan     0.1000 7816.0490
      8    63349.0526             nan     0.1000 6664.8330
      9    57501.5199             nan     0.1000 5855.7063
     10    52624.8695             nan     0.1000 4706.8617
     20    26521.4205             nan     0.1000 1327.6421
     40    16790.2482             nan     0.1000  123.5104
     60    15788.7972             nan     0.1000  -15.6398
     80    15508.1540             nan     0.1000    2.5246
    100    15224.7126             nan     0.1000   -9.3465
    120    15061.9041             nan     0.1000  -12.8816
    140    14853.1121             nan     0.1000   15.9811
    150    14766.0311             nan     0.1000  -15.3872

Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1   136877.5343             nan     0.1000 22733.3143
     2   118655.6111             nan     0.1000 18694.9545
     3   102447.6127             nan     0.1000 16265.4089
     4    89234.9460             nan     0.1000 13431.6135
     5    78351.1755             nan     0.1000 11019.3031
     6    68625.1439             nan     0.1000 9614.3297
     7    60759.3096             nan     0.1000 7991.1134
     8    53903.4588             nan     0.1000 6819.8774
     9    48163.3846             nan     0.1000 5833.3637
    10    43255.1211             nan     0.1000 4916.7647
    20    21625.5716             nan     0.1000  968.0691
    40    15999.8683             nan     0.1000   36.4788
    60    15316.0733             nan     0.1000   -1.1037
    80    14987.9627             nan     0.1000   17.7534
   100    14679.9270             nan     0.1000   29.1147
   120    14496.0497             nan     0.1000   -7.5966
   140    14367.1691             nan     0.1000  -14.1231
   150    14269.5182             nan     0.1000   24.8656

Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1   149153.7619             nan     0.1000 14642.3754
     2   136833.6886             nan     0.1000 12172.9889
     3   126053.6132             nan     0.1000 10746.0794
     4   116709.8662             nan     0.1000 9535.2347
     5   108384.8230             nan     0.1000 8290.3783
     6   101046.4212             nan     0.1000 7444.8419
     7    94409.3537             nan     0.1000 6559.6679
     8    88353.2879             nan     0.1000 6065.0835
     9    82812.3615             nan     0.1000 5408.0407
    10    77995.6975             nan     0.1000 4836.0161
    20    46484.8743             nan     0.1000 2089.3348
    40    24735.1873             nan     0.1000  166.2764
    60    18809.6366             nan     0.1000  171.6236
    80    17146.9637             nan     0.1000   46.8264
   100    16653.3236             nan     0.1000   14.5644
```

```
   120    16506.3416             nan     0.1000  -53.5809
   140    16429.8030             nan     0.1000  -47.3512
   150    16401.6634             nan     0.1000    0.5115

Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1   144323.3902             nan     0.1000 19178.7894
     2   127270.1014             nan     0.1000 17215.3541
     3   112386.4717             nan     0.1000 15062.1912
     4    99848.5252             nan     0.1000 12503.6938
     5    89848.3550             nan     0.1000 9909.5684
     6    80712.2458             nan     0.1000 9144.5741
     7    72876.7447             nan     0.1000 7954.5550
     8    66160.2423             nan     0.1000 6689.7437
     9    59723.0024             nan     0.1000 6311.8549
    10    54564.7975             nan     0.1000 5152.7494
    20    28018.4138             nan     0.1000 1359.7755
    40    17707.2495             nan     0.1000  137.7274
    60    16569.1823             nan     0.1000   -2.8618
    80    16210.2565             nan     0.1000    6.6548
   100    16014.9252             nan     0.1000  -30.4298
   120    15860.2573             nan     0.1000    2.9691
   140    15734.7168             nan     0.1000  -10.8348
   150    15716.0572             nan     0.1000   -7.5975

Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1   140109.8502             nan     0.1000 23252.5741
     2   120683.0271             nan     0.1000 19378.7753
     3   104710.0021             nan     0.1000 16157.7864
     4    91556.0296             nan     0.1000 13260.3197
     5    80310.3532             nan     0.1000 11507.1193
     6    70534.6779             nan     0.1000 9502.6980
     7    62698.4473             nan     0.1000 7718.2494
     8    55659.8297             nan     0.1000 6918.6639
     9    49829.3930             nan     0.1000 5764.7913
    10    44758.2755             nan     0.1000 4861.6603
    20    22621.0838             nan     0.1000  967.6176
    40    16703.0058             nan     0.1000  -35.5716
    60    16045.5725             nan     0.1000   22.4398
    80    15691.2307             nan     0.1000  -33.2691
   100    15422.8629             nan     0.1000  -26.2392
   120    15246.2596             nan     0.1000    3.9160
   140    15115.9438             nan     0.1000  -19.0056
   150    15036.1794             nan     0.1000   10.6739

Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1   148214.9896             nan     0.1000 14239.4653
     2   136170.4513             nan     0.1000 12044.3381
     3   125398.7052             nan     0.1000 10663.6563
     4   116210.6732             nan     0.1000 9125.1996
     5   107534.2834             nan     0.1000 8545.3280
     6   100411.8860             nan     0.1000 6950.6708
     7    93867.8861             nan     0.1000 6586.9108
     8    87721.7263             nan     0.1000 6065.6147
```

```
     9    82501.8813             nan     0.1000 5222.1612
    10    77588.8468             nan     0.1000 4893.5972
    20    46576.5278             nan     0.1000 2116.0583
    40    24837.8184             nan     0.1000  474.4211
    60    18950.3412             nan     0.1000  172.6816
    80    17132.6688             nan     0.1000   52.7658
   100    16578.3207             nan     0.1000   17.0132
   120    16385.3113             nan     0.1000   -8.1555
   140    16293.3169             nan     0.1000    3.1286
   150    16253.9476             nan     0.1000  -19.3252

Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1   143677.1275             nan     0.1000 18650.2740
     2   126905.6394             nan     0.1000 16547.8459
     3   112116.6149             nan     0.1000 14792.5103
     4   100060.9178             nan     0.1000 12112.3083
     5    89671.5620             nan     0.1000 10484.5959
     6    80814.6785             nan     0.1000 8831.3919
     7    72886.5792             nan     0.1000 8191.2640
     8    66115.3923             nan     0.1000 6761.6431
     9    60160.9967             nan     0.1000 6117.0275
    10    55118.6867             nan     0.1000 4978.6560
    20    28150.4529             nan     0.1000 1266.8424
    40    17525.6606             nan     0.1000  130.1492
    60    16438.1603             nan     0.1000  -14.1202
    80    16085.1175             nan     0.1000   -8.3071
   100    15900.9974             nan     0.1000  -21.2062
   120    15737.3533             nan     0.1000  -12.7264
   140    15605.5727             nan     0.1000   -5.1816
   150    15519.6700             nan     0.1000  -19.6602

Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1   139733.6801             nan     0.1000 23039.5750
     2   120932.0965             nan     0.1000 18699.7648
     3   104399.1192             nan     0.1000 16440.4035
     4    91356.5201             nan     0.1000 13216.1904
     5    79953.8379             nan     0.1000 11499.9061
     6    70231.8604             nan     0.1000 9691.4739
     7    62242.5827             nan     0.1000 7832.2693
     8    55747.2874             nan     0.1000 6033.6880
     9    49873.5470             nan     0.1000 5927.1504
    10    45001.3701             nan     0.1000 4794.7250
    20    22470.4748             nan     0.1000  994.5733
    40    16634.3255             nan     0.1000   54.6228
    60    15920.5687             nan     0.1000   38.5409
    80    15565.5818             nan     0.1000   -1.9057
   100    15317.2496             nan     0.1000   10.3205
   120    15046.6902             nan     0.1000  -19.0233
   140    14851.2134             nan     0.1000   -6.7077
   150    14764.0472             nan     0.1000    4.0206

Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1   147842.3733             nan     0.1000 14446.5427
```

```
      2   135655.1271             nan     0.1000 12207.9907
      3   124784.5059             nan     0.1000 11000.6530
      4   115370.6488             nan     0.1000 9254.7563
      5   106974.2029             nan     0.1000 8549.7005
      6    99692.0085             nan     0.1000 7307.9604
      7    93092.8653             nan     0.1000 6369.6995
      8    87005.6287             nan     0.1000 6011.3387
      9    81656.9844             nan     0.1000 5349.5093
     10    76641.6224             nan     0.1000 4948.9240
     20    45582.3857             nan     0.1000 2067.1050
     40    24250.7267             nan     0.1000  524.0396
     60    18553.6846             nan     0.1000  143.7512
     80    16925.7157             nan     0.1000   16.5473
    100    16523.9972             nan     0.1000   -2.9128
    120    16326.6998             nan     0.1000   -7.2972
    140    16248.9317             nan     0.1000  -42.3818
    150    16212.1181             nan     0.1000   -9.2136

Iter   TrainDeviance   ValidDeviance   StepSize   Improve
      1   143040.6597             nan     0.1000 19168.4749
      2   126077.7615             nan     0.1000 16940.4444
      3   111752.8615             nan     0.1000 14303.3254
      4    99163.3260             nan     0.1000 12637.8574
      5    88637.9863             nan     0.1000 10514.1835
      6    79606.1651             nan     0.1000 9123.0602
      7    71825.1183             nan     0.1000 7583.1707
      8    64936.3416             nan     0.1000 6752.1392
      9    59044.7888             nan     0.1000 5951.1920
     10    53860.9294             nan     0.1000 5149.2425
     20    27485.9521             nan     0.1000 1394.0245
     40    17353.0458             nan     0.1000   81.7095
     60    16305.2956             nan     0.1000  -13.5327
     80    16028.1447             nan     0.1000  -28.5111
    100    15812.9273             nan     0.1000  -11.5289
    120    15677.3911             nan     0.1000  -26.6797
    140    15585.5864             nan     0.1000  -26.5244
    150    15512.4320             nan     0.1000    2.0698

Iter   TrainDeviance   ValidDeviance   StepSize   Improve
      1   139261.1443             nan     0.1000 22988.3402
      2   120170.9998             nan     0.1000 18938.8281
      3   103802.4914             nan     0.1000 16298.0908
      4    90528.1711             nan     0.1000 13024.0544
      5    79008.4567             nan     0.1000 11576.9838
      6    69680.3694             nan     0.1000 9109.0849
      7    61748.6943             nan     0.1000 7750.9891
      8    54906.6798             nan     0.1000 6898.8726
      9    49182.6611             nan     0.1000 5779.3146
     10    44257.3345             nan     0.1000 5083.5497
     20    22142.3004             nan     0.1000  959.0931
     40    16485.5900             nan     0.1000   25.1957
     60    15860.4306             nan     0.1000   21.7471
     80    15547.7253             nan     0.1000  -10.0488
```

```
   100    15233.7293             nan     0.1000  -22.8122
   120    15010.7107             nan     0.1000  -15.2368
   140    14870.9043             nan     0.1000  -47.2367
   150    14802.2184             nan     0.1000  -52.0849

Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1   139415.1048             nan     0.1000 23084.2465
     2   120553.8124             nan     0.1000 18913.2735
     3   104166.1511             nan     0.1000 16314.6601
     4    90691.5153             nan     0.1000 13666.9634
     5    79525.6212             nan     0.1000 11329.8061
     6    69901.9108             nan     0.1000 9885.6209
     7    61822.6553             nan     0.1000 7945.3351
     8    54951.9533             nan     0.1000 6790.1159
     9    49265.0919             nan     0.1000 5463.7023
    10    44355.4032             nan     0.1000 4842.6012
    20    22327.0843             nan     0.1000  904.3325
    40    16479.8600             nan     0.1000   59.4266
    60    15867.5611             nan     0.1000  -19.2958
    80    15494.2997             nan     0.1000   35.1478
   100    15248.1918             nan     0.1000  -11.8119
   120    15032.7148             nan     0.1000   -9.8130
   140    14823.4082             nan     0.1000    2.4229
   150    14742.7727             nan     0.1000    6.7481
```

```r
# Prediksi menggunakan meta-model
preds_stack <- predict(meta_model, stack_test)

# Menghitung RMSE
rmse <- sqrt(mean((preds_stack - y_test)^2))
cat("Root Mean Squared Error (RMSE):", rmse, "\n")
```

```
Root Mean Squared Error (RMSE): 163.5543
```

```r
# Menghitung R-squared (R²)
sst <- sum((y_test - mean(y_test))^2)
sse <- sum((y_test - preds_stack)^2)
r2_stack <- 1 - (sse / sst)
cat("R-squared (R²):", r2_stack, "\n")
```

```
R-squared (R²): 0.8382397
```

Metode stacking dengan Gradient Boosting Machine (GBM) sebagai meta-model menunjukkan peningkatan performa yang signifikan dibandingkan pendekatan individual maupun ensemble averaging. Dengan RMSE sebesar 163.55, model ini memiliki kesalahan rata-rata yang lebih kecil dibandingkan ensemble averaging, menunjukkan peningkatan akurasi prediksi. Selain itu, R^2 sebesar 0.8382 menunjukkan bahwa model mampu menjelaskan **83.82% variasi dalam data target. Peningkatan ini mengonfirmasi bahwa stacking dengan meta-model yang kuat dapat lebih efektif dalam menggabungkan kekuatan model dasar, menghasilkan prediksi yang lebih akurat dan stabil.**

Comparing R^2

```
# Membuat Data Frame untuk R-squared
r2_results <- data.frame(
  Model = c("LightGBM", "XGBoost", "Random Forest", "Averanging", "Stacking"),
  R_squared = c(r2_lgb, r2_xgb, r2_rf,r2_ens, r2_stack)
)

# Menampilkan tabel
print(r2_results)
```

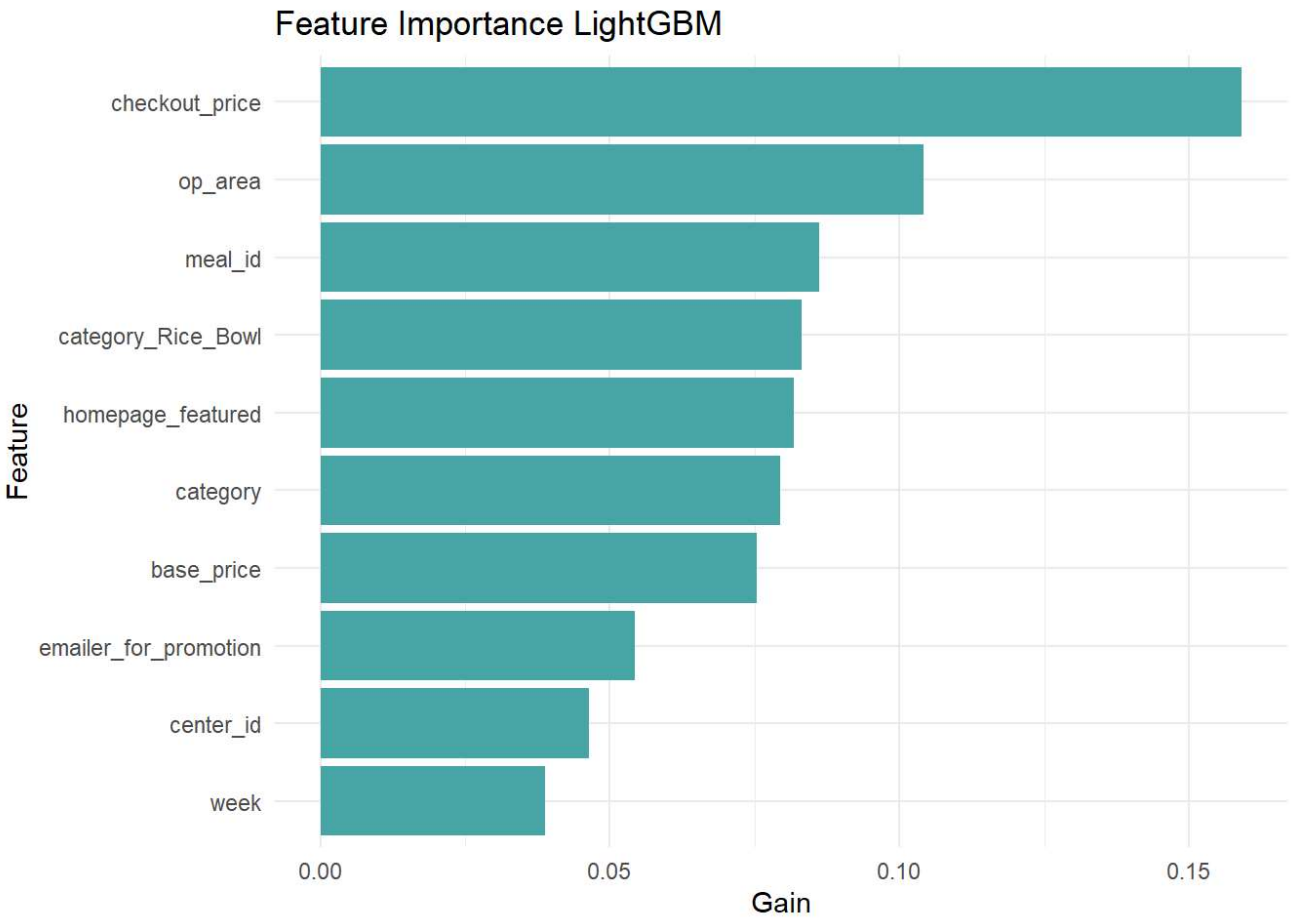
```
          Model R_squared
1      LightGBM 0.8118930
2       XGBoost 0.8343959
3 Random Forest 0.8261950
4    Averanging 0.8304566
5      Stacking 0.8382397
```

Berdasarkan hasil pemodelan, metode stacking menunjukkan performa terbaik dengan **R^2 sebesar 83.82%**, lebih tinggi dibandingkan model individu maupun ensemble- averaging. Maka, **model stacking akan digunakan untuk melakukan prediksi num_orders pada week 135-145** karena mampu memberikan hasil yang lebih akurat dan stabil dalam menjelaskan variabilitas data.

Feature Importance

###1. LightGBM

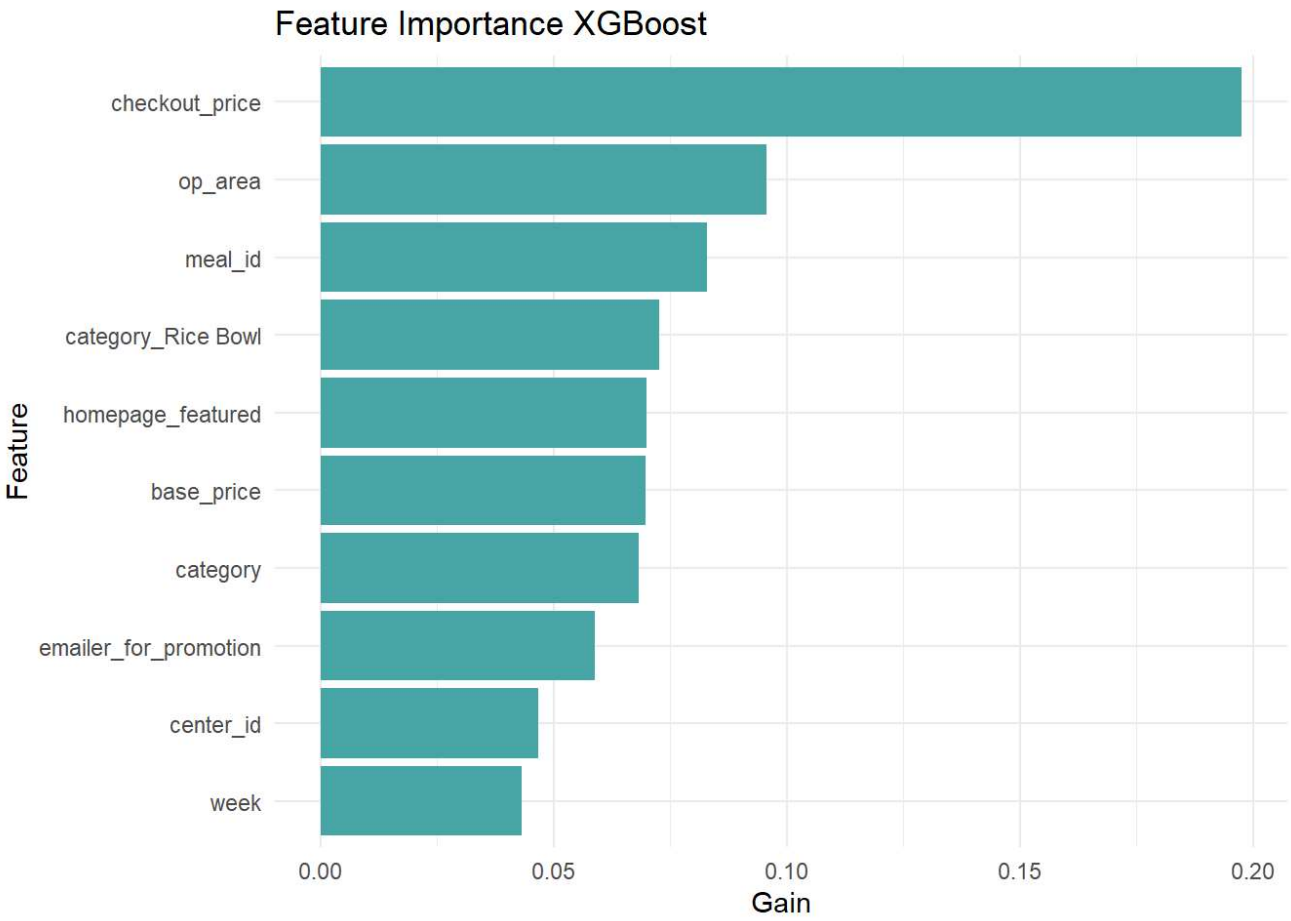
```
# Import feature importance
importance_lgb <- lgb.importance(model_lgb, percentage = TRUE)

# Konversi ke dataframe untuk plotting manual
importance_lgb$Feature <- factor(importance_lgb$Feature, levels = rev(importance_lgb

# Plot dengan ggplot2 dan warna kustom
ggplot(importance_lgb[1:10, ], aes(x = Gain, y = Feature)) +
  geom_col(fill = "#48A6A7") +
  labs(title = "Feature Importance LightGBM", x = "Gain", y = "Feature") +
  theme_minimal()
```



Berdasarkan grafik feature importance model LightGBM, fitur yang paling berpengaruh dalam memprediksi jumlah pesanan (`num_orders`) adalah `checkout_price`, diikuti oleh `op_area` dan kategori makanan seperti `category_Rice_Bowl`. Hal ini menunjukkan bahwa **harga akhir dari suatu produk memiliki pengaruh terbesar terhadap jumlah pesanan** karena harga sering kali menjadi faktor utama dalam keputusan pembelian. Selain itu, `op_area` berada di urutan kedua yang mengindikasikan bahwa **lokasi atau cakupan layanan memengaruhi jumlah pesanan.** Fitur seperti `homepage_featured` dan `emailer_for_promotion` juga memiliki peran penting, yang mencerminkan pengaruh promosi dan visibilitas di halaman utama terhadap permintaan. Fitur lainnya seperti `base_price`, `meal_id`, dan `center_id` berkontribusi lebih rendah, namun tetap relevan dalam membangun model. Hal ini dapat menunjukkan pentingnya mengoptimalkan harga, strategi promosi, dan distribusi geografis untuk meningkatkan jumlah pesanan.

2. XGBoost

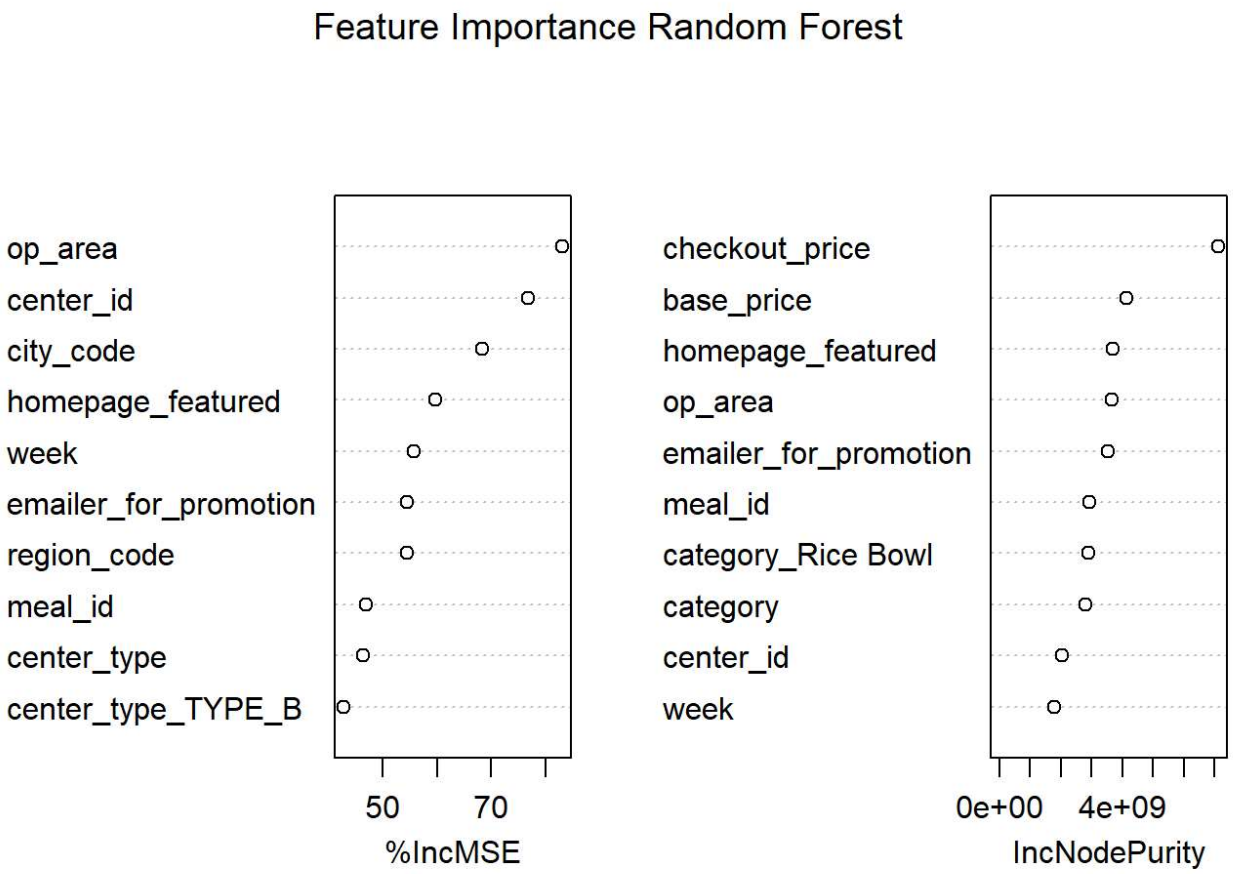
```
# Mendapatkan feature importance
importance_xgb <- xgb.importance(feature_names = colnames(X_train_final), model = mo

# Konversi ke dataframe untuk plotting manual
importance_xgb$Feature <- factor(importance_xgb$Feature, levels = rev(importance_xgb

# Plot dengan ggplot2 dan warna kustom
ggplot(importance_xgb[1:10, ], aes(x = Gain, y = Feature)) +
  geom_col(fill = "#48A6A7") +
  labs(title = "Feature Importance XGBoost", x = "Gain", y = "Feature") +
  theme_minimal()
```



Berdasarkan grafik feature importance dari model XGBoost, fitur yang memiliki pengaruh paling besar dalam memprediksi jumlah pesanan (`num_orders`) adalah `checkout_price`, diikuti oleh `op_area` dan kategori makanan seperti `category_Rice_Bowl`. **Kondisi ini mirip dengan model LightGBM sebelumnya.** Fitur `op_area` juga berada di urutan kedua, yang mengindikasikan bahwa lokasi operasional atau cakupan layanan memiliki pengaruh signifikan terhadap permintaan. Fitur lainnya seperti `homepage_featured` dan `emailer_for_promotion` juga memiliki kontribusi yang penting, mencerminkan bahwa promosi dan visibilitas di halaman utama berdampak pada peningkatan jumlah pesanan. Selain itu, fitur seperti `base_price`, `category_Sandwich`, dan `cuisine_Italian` relevan untuk memahami pola permintaan. Kondisi ini dapat menunjukkan mengoptimalkan harga, lokasi layanan, dan strategi promosi untuk meningkatkan jumlah pesanan.

3. Random Forest

```
### Feature Importance untuk Random Forest
importance_rf <- importance(model_rf)
varImpPlot(model_rf, n.var = 10, main = "Feature Importance Random Forest")
```



Berdasarkan grafik feature importance dari model Random Forest, ada dua metrik utama yang digunakan, yaitu %IncMSE (**peningkatan Mean Squared Error jika fitur diacak**) dan IncNodePurity (**peningkatan purity node dalam pohon keputusan**).

Berdasarkan %IncMSE, Fitur `op_area` memiliki pengaruh terbesar terhadap akurasi model, yang menunjukkan bahwa **area operasi sangat penting dalam memprediksi jumlah pesanan** (`num_orders`). Fitur lain yang signifikan adalah `center_id`, `city_code`, dan `homepage_featured`, yang mengindikasikan **bahwa faktor lokasi, pusat distribusi, dan visibilitas di halaman utama memengaruhi hasil prediksi.** `checkout_price` juga memiliki pengaruh besar, yang sesuai dengan pentingnya harga dalam keputusan pembelian.

Dari segi node purity, fitur `checkout_price` menunjukkan kontribusi terbesar, diikuti oleh `homepage_featured`, `base_price`, dan `op_area`. Hal ini menunjukkan bahwa **fitur-fitur tersebut secara signifikan membantu membagi data menjadi kelompok-kelompok yang lebih homogen** di sepanjang pohon keputusan.

Secara keseluruhan, `checkout_price` dan `op_area` konsisten sebagai fitur yang sangat penting dalam kedua metrik ini. Hal ini memberikan petunjuk bahwa **strategi harga, cakupan operasional, serta promosi memiliki dampak besar terhadap jumlah pesanan.**

Prediction

Pre-Processing Data

Data Testing dilakukan pre-processing data seperti yang sebelumnya sudah dilakukan pada data training untuk membangkit model. Hal ini dilakukan untuk **memastikan konsistensi antara data training dan testing** sehingga model yang telah dilatih pada data training dapat memproses data testing dengan karakteristik yang serupa.

```
test <- test %>%
  left_join(meal_info, by = "meal_id")

head(test)
```

```
# A tibble: 6 × 10
      id  week center_id meal_id checkout_price base_price emailer_for_promotion
   <dbl> <dbl>     <dbl>   <dbl>          <dbl>      <dbl>                  <dbl>
1 1.02e6   136        55    1885          148.        148.                      0
2 1.40e6   136        55    1993          151.        151.                      0
3 1.01e6   136        55    2539          152.        151.                      0
4 1.04e6   136        55    2631           96.0       166.                      0
5 1.02e6   136        55    1248           97         166.                      0
6 1.19e6   136        55    1778          165.        165.                      0
# i 3 more variables: homepage_featured <dbl>, category <chr>, cuisine <chr>
```

```
test <- test %>%
  left_join(fulfilment_center_info, by = "center_id")

head(test)
```

```
# A tibble: 6 × 14
      id  week center_id meal_id checkout_price base_price emailer_for_promotion
   <dbl> <dbl>     <dbl>   <dbl>          <dbl>      <dbl>                  <dbl>
1 1.02e6   136        55    1885          148.        148.                      0
2 1.40e6   136        55    1993          151.        151.                      0
3 1.01e6   136        55    2539          152.        151.                      0
4 1.04e6   136        55    2631           96.0       166.                      0
5 1.02e6   136        55    1248           97         166.                      0
6 1.19e6   136        55    1778          165.        165.                      0
# i 7 more variables: homepage_featured <dbl>, category <chr>, cuisine <chr>,
#   city_code <int>, region_code <int>, center_type <chr>, op_area <dbl>
```

```
str(test)
```

```
tibble [32,821 × 14] (S3: tbl_df/tbl/data.frame)
 $ id                   : num [1:32821] 1017495 1395634 1007493 1042952 1022147 ...
 $ week                 : num [1:32821] 136 136 136 136 136 136 136 136 136 136 ...
 $ center_id            : num [1:32821] 55 55 55 55 55 55 55 55 55 55 ...
 $ meal_id              : num [1:32821] 1885 1993 2539 2631 1248 ...
 $ checkout_price       : num [1:32821] 148 151 152 96 97 ...
 $ base_price           : num [1:32821] 148 151 151 166 166 ...
 $ emailer_for_promotion: num [1:32821] 0 0 0 0 0 0 0 0 0 0 ...
 $ homepage_featured    : num [1:32821] 0 0 0 0 0 0 0 0 1 0 ...
 $ category             : chr [1:32821] "Beverages" "Beverages" "Beverages"
"Beverages" ...
```

```
 $ cuisine              : chr [1:32821] "Thai" "Thai" "Thai" "Indian" ...
 $ city_code            : int [1:32821] 647 647 647 647 647 647 647 647 647 647 ...
 $ region_code          : int [1:32821] 56 56 56 56 56 56 56 56 56 56 ...
 $ center_type          : chr [1:32821] "TYPE_C" "TYPE_C" "TYPE_C" "TYPE_C" ...
 $ op_area              : num [1:32821] 2 2 2 2 2 2 2 2 2 2 ...
```

```r
# Memastikan X_train dan X_test adalah data.table
setDT(test)

# Identifikasi fitur kategorikal yang valid
categorical_features <- names(test)[sapply(test, is.character)]
categorical_features <- intersect(names(test), categorical_features)

# Konversi fitur kategorikal menjadi faktor
test[, (categorical_features) := lapply(.SD, as.factor), .SDcols = categorical_featu

# Pastikan categorical_features hanya berisi fitur yang ada di X_train_final
setdiff(categorical_features, names(test))  # Harusnya output kosong
```

```
character(0)
```

```r
# One-Hot Encoding untuk fitur kategorikal pada X_train dan X_test
test_encoded <- dummy_cols(test, select_columns = categorical_features, remove_first

# Pastikan semua kolom dalam data adalah numerik setelah encoding
test_final <-test_encoded %>%
  select(-id)

test_final <- data.matrix(test_final)
head(test_final)
```

```
     week center_id meal_id checkout_price base_price emailer_for_promotion
[1,]  136        55    1885         148.44     148.44                     0
[2,]  136        55    1993         151.38     151.38                     0
[3,]  136        55    2539         152.35     151.35                     0
[4,]  136        55    2631          96.03     165.93                     0
[5,]  136        55    1248          97.00     165.93                     0
[6,]  136        55    1778         164.90     164.90                     0
     homepage_featured category cuisine city_code region_code center_type
[1,]                 0        1       4       647          56           3
[2,]                 0        1       4       647          56           3
[3,]                 0        1       4       647          56           3
[4,]                 0        1       2       647          56           3
[5,]                 0        1       2       647          56           3
[6,]                 0        1       3       647          56           3
     op_area category_Biryani category_Desert category_Extras category_Fish
[1,]       2                0               0               0             0
[2,]       2                0               0               0             0
[3,]       2                0               0               0             0
[4,]       2                0               0               0             0
[5,]       2                0               0               0             0
[6,]       2                0               0               0             0
```

```
     category_Other Snacks category_Pasta category_Pizza category_Rice Bowl
[1,]                     0              0              0                  0
[2,]                     0              0              0                  0
[3,]                     0              0              0                  0
[4,]                     0              0              0                  0
[5,]                     0              0              0                  0
[6,]                     0              0              0                  0
     category_Salad category_Sandwich category_Seafood category_Soup
[1,]              0                 0                0             0
[2,]              0                 0                0             0
[3,]              0                 0                0             0
[4,]              0                 0                0             0
[5,]              0                 0                0             0
[6,]              0                 0                0             0
     category_Starters cuisine_Indian cuisine_Italian cuisine_Thai
[1,]                 0              0               0            1
[2,]                 0              0               0            1
[3,]                 0              0               0            1
[4,]                 0              1               0            0
[5,]                 0              1               0            0
[6,]                 0              0               1            0
     center_type_TYPE_B center_type_TYPE_C
[1,]                  0                 1
[2,]                  0                 1
[3,]                  0                 1
[4,]                  0                 1
[5,]                  0                 1
[6,]                  0                 1
```

```r
# Konfirmasi bahwa tidak ada NA di data
sum(is.na(test_final))  # Harusnya 0
```

```
[1] 0
```

Data pada minggu ke-136 sampai minggu ke-145 yang sudah ada dalam data training sebelumnya akan digunakan sebagai validasi dari hasil prediksi `num_orders`.

```r
y_test <- test_validation$num_orders
```

Stacking

Metode stacking dengan **Gradient Boosting Machine (GBM) sebagai meta-model** akan dilakukan untuk prediksi karena memiliki performa terbaik dibandingkan dengan model-model lainnya.

```r
stack_test <- data.frame(
  y_test,
  preds_lgb = predict(model_lgb, as.matrix(test_final)),
  preds_xgb = predict(model_xgb, as.matrix(test_final)),
  preds_rf  = predict(model_rf, test_final)
)
```

```r
# Prediksi menggunakan meta-model
preds_stack <- predict(meta_model, stack_test)

# Menghitung RMSE
rmse <- sqrt(mean((preds_stack - y_test)^2))
cat("Root Mean Squared Error (RMSE):", rmse, "\n")
```

```
Root Mean Squared Error (RMSE): 123.5809
```

```r
# Menghitung R-squared (R²)
sst <- sum((y_test - mean(y_test))^2)
sse <- sum((y_test - preds_stack)^2)
r2_stack <- 1 - (sse / sst)
cat("R-squared (R²):", r2_stack, "\n")
```

```
R-squared (R²): 0.8003736
```

Hasil prediksi menggunakan metode stacking dengan Gradient Boosting Machine (GBM) sebagai meta-model menunjukkan performa yang cukup baik. Nilai Root Mean Squared Error (RMSE) sebesar 123.58 menunjukkan bahwa rata-rata kesalahan prediksi model terhadap nilai aktual berada dalam kisaran tersebut. Sementara itu, nilai **R-squared (R^2) sebesar 0.8004 mengindikasikan bahwa model dapat menjelaskan sekitar 80.04% variasi dalam validasi hasil prediksi.** Ini menunjukkan bahwa pendekatan stacking berhasil menggabungkan keunggulan dari model LightGBM, XGBoost, dan Random Forest untuk meningkatkan akurasi prediksi dibandingkan dengan model individu. Namun, masih terdapat sekitar 19.96% variasi yang tidak dapat dijelaskan oleh model, yang bisa menjadi **peluang untuk perbaikan lebih lanjut, misalnya dengan melakukan tuning hyperparameter atau menambahkan fitur tambahan.**