assignment_2_chan_jun_hao

2024-03-11

Report Structure

Report Summary

Project context This project aims to optimize a Tesco store's inventory by modeling the relationship between property type and buying behaviors of its residents. The project involves analyzing Tesco grocery data and London's property data to identify patterns and trends in the data. Machine learning techniques will be used to predict food categories purchased based on the property distribution. **Datasets used** 1. Tesco Grocery Data: A record of 420 M food items purchased by 1.6 M fidelity card owners who shopped at the 411 Tesco stores in Greater London over the course of the entire year of 2015. 2. London's Property Data: A breakdown of the dwelling stock down to a lower geographic level Lower layer Super Output Area or LSOA, categorized by the property build period and property type. **Research Question** Is it possible to optimize a Tesco Store's inventory by modeling the relationship between property type and buying behaviors of its residents? **Conclusion and Findings** 1. From the analysis, the feature that had the most influence on predicting food categories was month, and not so much about the property types. 2. Although property type features do have an impact on the food categories purchased, the month feature has a higher impact. With this overall insight in mind, perhaps for future analysis, we could explore the relationship between the month feature, in particular the weather statistics and the food categories purchased.

Import Packages

The code below imports the packages used.

```
# Import Packages
suppressMessages({
  library(tidyverse)    # Data manipulation
  library(lubridate)    # Date time objects
  library(ggplot2)      # Plotting
  library(readr)        # Read csv
  library(caret)        # Machine learning
  library(keras)        # Machine learning
  library(Metrics)      # Machine learning evluation
  library(ranger)       # Random forest
  library(dplyr)        # Data manipulation
  library(gridExtra)    # Plot
  library(stringr)      # String manipulation
  library(purrr)        # Data manipulation
  library(corrplot)     # Correlation plot
  library(forcats)      # Factor manipulation
  library(sf)           # Spatial data
  library(RColorBrewer) # Color palettes
  library(ggrepel)      # Plotting
})
```

```
## Warning: package 'readr' was built under R version 4.3.3

## Warning: package 'caret' was built under R version 4.3.3

## Warning: package 'keras' was built under R version 4.3.3

## Warning: package 'Metrics' was built under R version 4.3.3

## Warning: package 'ranger' was built under R version 4.3.3

## Warning: package 'gridExtra' was built under R version 4.3.3

## Warning: package 'sf' was built under R version 4.3.3
```

Import Data

The code below imports the downloaded data.

```
# Import Data
suppressMessages({
  property <- read_csv("data/assignment_2/dwelling-property-type-2015-lsoa-msoa.csv")
  property_metadata <- read_csv("data/assignment_2/voa-csv-metadata.csv")
  msoa_april <- read_csv("data/assignment_2/Apr_msoa_grocery.csv")
  msoa_august <- read_csv("data/assignment_2/Aug_msoa_grocery.csv")
  lsoa_january <- read_csv("data/assignment_2/Jan_lsoa_grocery.csv")
```

```r
  lsoa_february <- read_csv("data/assignment_2/Feb_lsoa_grocery.csv")
  lsoa_march <- read_csv("data/assignment_2/Mar_lsoa_grocery.csv")
  lsoa_april <- read_csv("data/assignment_2/Apr_lsoa_grocery.csv")
  lsoa_may <- read_csv("data/assignment_2/May_lsoa_grocery.csv")
  lsoa_june <- read_csv("data/assignment_2/Jun_lsoa_grocery.csv")
  lsoa_july <- read_csv("data/assignment_2/Jul_lsoa_grocery.csv")
  lsoa_august <- read_csv("data/assignment_2/Aug_lsoa_grocery.csv")
  lsoa_september <- read_csv("data/assignment_2/Sep_lsoa_grocery.csv")
  lsoa_october <- read_csv("data/assignment_2/Oct_lsoa_grocery.csv")
  lsoa_november <- read_csv("data/assignment_2/Nov_lsoa_grocery.csv")
  lsoa_december <- read_csv("data/assignment_2/Dec_lsoa_grocery.csv")
  london_map_path<- "data/assignment_2/LSOA_2011_London_gen_MHW.shp"
})
```

Desktop Research - London's Geographic Aggregations

Preliminary desktop research was done to **understand the different geographic aggregations in London.** The following information is taken from the Office for National Statistics, UK.

Boroughs London is divided into 32 boroughs and the City of London. The boroughs are the second level of local government below the Greater London Authority. They are responsible for local services such as schools, waste collection, and planning.

Wards Wards are the key building blocks of UK administrative geography. They are the areas used to elect local government councillors. Wards are made up of Output Areas (OAs).

Middle layer Super Output Areas Middle layer Super Output Areas (MSOAs) are made up of groups of Lower layer Super Output Areas (LSOAs), usually four or five. They comprise between 2,000 and 6,000 households and have a usually resident population between 5,000 and 15,000 persons.

Lower layer Super Output Areas Lower layer Super Output Areas (LSOAs) are made up of groups of Output Areas (OAs), usually four or five. They comprise between 400 and 1,200 households and have a usually resident population between 1,000 and 3,000 persons.

Preliminary EDA - Tesco Grocery Data

We preview the data here and understand the structure of the data. The data set is obtained from: https://figshare.com/collections/Tesco_Grocery_1_0/4769354/2.

Data set Background The dataset is a record of 420 M food items purchased by 1.6 M fidelity card owners who shopped at the 411 Tesco stores in Greater London over the course of the entire year of 2015, aggregated at the level of census areas to preserve anonymity.

Data set Categories The data set contains the following measurement categories: 1. Area ID 2. Nutritional Information 3. Food Categories 4. Population/Demographic Information 5. Geographical Information 6. Other Metrics

```r
# Display the first few rows of grocery dataset
head(lsoa_january)
```

```
## # A tibble: 6 x 202
##   area_id   weight weight_perc2.5 weight_perc25 weight_perc50 weight_perc75
##   <chr>      <dbl>          <dbl>         <dbl>         <dbl>         <dbl>
## 1 E01000001   324.             35           150           270           430
## 2 E01000002   312.             25           150           250           415
## 3 E01000003   334.             37           150           270           450
## 4 E01000005   362.             41           160           300           454
## 5 E01000006   451.             45           180           340           500
## 6 E01000007   455.             29           170           340           500
## # i 196 more variables: weight_perc97.5 <dbl>, weight_std <dbl>,
## #   weight_ci95 <dbl>, volume <dbl>, volume_perc2.5 <dbl>, volume_perc25 <dbl>,
## #   volume_perc50 <dbl>, volume_perc75 <dbl>, volume_perc97.5 <dbl>,
## #   volume_std <dbl>, volume_ci95 <dbl>, fat <dbl>, fat_perc2.5 <dbl>,
## #   fat_perc25 <dbl>, fat_perc50 <dbl>, fat_perc75 <dbl>, fat_perc97.5 <dbl>,
## #   fat_std <dbl>, fat_ci95 <dbl>, saturate <dbl>, saturate_perc2.5 <dbl>,
## #   saturate_perc25 <dbl>, saturate_perc50 <dbl>, saturate_perc75 <dbl>, ...
```

```
# Print feature names of the grocery data set
names(lsoa_january)
```

```
##   [1] "area_id"                  "weight"
##   [3] "weight_perc2.5"           "weight_perc25"
##   [5] "weight_perc50"            "weight_perc75"
##   [7] "weight_perc97.5"          "weight_std"
##   [9] "weight_ci95"              "volume"
##  [11] "volume_perc2.5"           "volume_perc25"
##  [13] "volume_perc50"            "volume_perc75"
##  [15] "volume_perc97.5"          "volume_std"
##  [17] "volume_ci95"              "fat"
##  [19] "fat_perc2.5"              "fat_perc25"
##  [21] "fat_perc50"               "fat_perc75"
##  [23] "fat_perc97.5"             "fat_std"
##  [25] "fat_ci95"                 "saturate"
##  [27] "saturate_perc2.5"         "saturate_perc25"
##  [29] "saturate_perc50"          "saturate_perc75"
##  [31] "saturate_perc97.5"        "saturate_std"
##  [33] "saturate_ci95"            "salt"
##  [35] "salt_perc2.5"             "salt_perc25"
##  [37] "salt_perc50"              "salt_perc75"
##  [39] "salt_perc97.5"            "salt_std"
##  [41] "salt_ci95"                "sugar"
##  [43] "sugar_perc2.5"            "sugar_perc25"
##  [45] "sugar_perc50"             "sugar_perc75"
##  [47] "sugar_perc97.5"           "sugar_std"
##  [49] "sugar_ci95"               "protein"
##  [51] "protein_perc2.5"          "protein_perc25"
##  [53] "protein_perc50"           "protein_perc75"
##  [55] "protein_perc97.5"         "protein_std"
##  [57] "protein_ci95"             "carb"
##  [59] "carb_perc2.5"             "carb_perc25"
##  [61] "carb_perc50"              "carb_perc75"
##  [63] "carb_perc97.5"            "carb_std"
##  [65] "carb_ci95"                "fibre"
##  [67] "fibre_perc2.5"            "fibre_perc25"
```

```
##  [69] "fibre_perc50"               "fibre_perc75"
##  [71] "fibre_perc97.5"             "fibre_std"
##  [73] "fibre_ci95"                 "alcohol"
##  [75] "alcohol_perc2.5"            "alcohol_perc25"
##  [77] "alcohol_perc50"             "alcohol_perc75"
##  [79] "alcohol_perc97.5"           "alcohol_std"
##  [81] "alcohol_ci95"               "energy_fat"
##  [83] "energy_fat_perc2.5"         "energy_fat_perc25"
##  [85] "energy_fat_perc50"          "energy_fat_perc75"
##  [87] "energy_fat_perc97.5"        "energy_fat_std"
##  [89] "energy_fat_ci95"            "energy_saturate"
##  [91] "energy_saturate_perc2.5"    "energy_saturate_perc25"
##  [93] "energy_saturate_perc50"     "energy_saturate_perc75"
##  [95] "energy_saturate_perc97.5"   "energy_saturate_std"
##  [97] "energy_saturate_ci95"       "energy_sugar"
##  [99] "energy_sugar_perc2.5"       "energy_sugar_perc25"
## [101] "energy_sugar_perc50"        "energy_sugar_perc75"
## [103] "energy_sugar_perc97.5"      "energy_sugar_std"
## [105] "energy_sugar_ci95"          "energy_protein"
## [107] "energy_protein_perc2.5"     "energy_protein_perc25"
## [109] "energy_protein_perc50"      "energy_protein_perc75"
## [111] "energy_protein_perc97.5"    "energy_protein_std"
## [113] "energy_protein_ci95"        "energy_carb"
## [115] "energy_carb_perc2.5"        "energy_carb_perc25"
## [117] "energy_carb_perc50"         "energy_carb_perc75"
## [119] "energy_carb_perc97.5"       "energy_carb_std"
## [121] "energy_carb_ci95"           "energy_fibre"
## [123] "energy_fibre_perc2.5"       "energy_fibre_perc25"
## [125] "energy_fibre_perc50"        "energy_fibre_perc75"
## [127] "energy_fibre_perc97.5"      "energy_fibre_std"
## [129] "energy_fibre_ci95"          "energy_alcohol"
## [131] "energy_alcohol_perc2.5"     "energy_alcohol_perc25"
## [133] "energy_alcohol_perc50"      "energy_alcohol_perc75"
## [135] "energy_alcohol_perc97.5"    "energy_alcohol_std"
## [137] "energy_alcohol_ci95"        "energy_tot"
## [139] "energy_tot_perc2.5"         "energy_tot_perc25"
## [141] "energy_tot_perc50"          "energy_tot_perc75"
## [143] "energy_tot_perc97.5"        "energy_tot_std"
## [145] "energy_tot_ci95"            "f_energy_fat"
## [147] "f_energy_saturate"          "f_energy_sugar"
## [149] "f_energy_protein"           "f_energy_carb"
## [151] "f_energy_fibre"             "f_energy_alcohol"
## [153] "energy_density"             "h_nutrients_weight"
## [155] "h_nutrients_weight_norm"    "h_nutrients_calories"
## [157] "h_nutrients_calories_norm" "f_beer"
## [159] "f_dairy"                    "f_eggs"
## [161] "f_fats_oils"                "f_fish"
## [163] "f_fruit_veg"                "f_grains"
## [165] "f_meat_red"                 "f_poultry"
## [167] "f_readymade"                "f_sauces"
## [169] "f_soft_drinks"              "f_spirits"
## [171] "f_sweets"                   "f_tea_coffee"
## [173] "f_water"                    "f_wine"
## [175] "f_dairy_weight"             "f_eggs_weight"
```

```
## [177] "f_fats_oils_weight"       "f_fish_weight"
## [179] "f_fruit_veg_weight"       "f_grains_weight"
## [181] "f_meat_red_weight"        "f_poultry_weight"
## [183] "f_readymade_weight"       "f_sauces_weight"
## [185] "f_sweets_weight"          "h_items"
## [187] "h_items_norm"             "h_items_weight"
## [189] "h_items_weight_norm"      "representativeness_norm"
## [191] "transaction_days"         "num_transactions"
## [193] "man_day"                  "population"
## [195] "male"                     "female"
## [197] "age_0_17"                 "age_18_64"
## [199] "age_65+"                  "avg_age"
## [201] "area_sq_km"               "people_per_sq_km"
```

```r
# Define a function to remove suffixes from column names
remove_suffix <- function(names_vector) {
  # Define a regular expression pattern to match the specified suffixes
  pattern <- "_std$|_ci95$|_perc\\d+\\.?\\d*$"

  # Remove the suffixes from the column names using gsub()
  cleaned_names <- gsub(pattern, "", names_vector)

  # Return the cleaned and unique column names
  return(unique(cleaned_names))
}

# Apply the function to the column names of your dataset
cleaned_names <- remove_suffix(names(lsoa_january))

# Print the cleaned and unique column names
print(cleaned_names)
```

```
##  [1] "area_id"                    "weight"
##  [3] "volume"                     "fat"
##  [5] "saturate"                   "salt"
##  [7] "sugar"                      "protein"
##  [9] "carb"                       "fibre"
## [11] "alcohol"                    "energy_fat"
## [13] "energy_saturate"            "energy_sugar"
## [15] "energy_protein"             "energy_carb"
## [17] "energy_fibre"               "energy_alcohol"
## [19] "energy_tot"                 "f_energy_fat"
## [21] "f_energy_saturate"          "f_energy_sugar"
## [23] "f_energy_protein"           "f_energy_carb"
## [25] "f_energy_fibre"             "f_energy_alcohol"
## [27] "energy_density"             "h_nutrients_weight"
## [29] "h_nutrients_weight_norm"    "h_nutrients_calories"
## [31] "h_nutrients_calories_norm" "f_beer"
## [33] "f_dairy"                    "f_eggs"
## [35] "f_fats_oils"                "f_fish"
## [37] "f_fruit_veg"                "f_grains"
## [39] "f_meat_red"                 "f_poultry"
## [41] "f_readymade"                "f_sauces"
## [43] "f_soft_drinks"              "f_spirits"
```

```
## [45] "f_sweets"              "f_tea_coffee"
## [47] "f_water"               "f_wine"
## [49] "f_dairy_weight"        "f_eggs_weight"
## [51] "f_fats_oils_weight"    "f_fish_weight"
## [53] "f_fruit_veg_weight"    "f_grains_weight"
## [55] "f_meat_red_weight"     "f_poultry_weight"
## [57] "f_readymade_weight"    "f_sauces_weight"
## [59] "f_sweets_weight"       "h_items"
## [61] "h_items_norm"          "h_items_weight"
## [63] "h_items_weight_norm"   "representativeness_norm"
## [65] "transaction_days"      "num_transactions"
## [67] "man_day"               "population"
## [69] "male"                  "female"
## [71] "age_0_17"              "age_18_64"
## [73] "age_65+"               "avg_age"
## [75] "area_sq_km"            "people_per_sq_km"
```

Observation

1. The grocery data set has 32 features.

2. The main categories of the data set are:

- Nutritional Information (19 features)
- Food Categories (17 features)
- Population/Demographic Information (7 features)
- Geographical Information (3 features)
- Other Metrics (12 features)

3. From the above, the categories are **heavily skewed towards Nutritional Information and Food Categories**.

4. The data set was given in both yearly and monthly formats, with each broken down into different geographical levels of granularity.

Conclusion

From the initial data scoping, several categories was of interest. The main categories of interest were:

1. Food Categories
2. Population/Demographic Information
3. Geographical Information

In particular, I was interested to see if there was a **relationship between the urban environment and the food categories purchased.**

Preliminary EDA - London's Property Data

We preview the data here and understand the structure of the data. The data set is obtained from:https://data.london.gov.uk/dataset/property-build-period-lsoa.

Data set Background The data shows a breakdown of the dwelling stock down to a lower geographic level Lower layer Super Output Area or LSOA, categorized by the property build period and property type.

Data set Categories The data contains the following measurement categories: 1. Geography 2. Property Type 3. Tax Band 4. Number of Bedrooms 5. Number of Properties

```r
# Display head of property data
head(property)
```

```
## # A tibble: 6 x 38
##   GEOGRAPHY ECODE     AREA_NAME         BAND  TYPE_BUNGALOW_1 TYPE_BUNGALOW_2
##   <chr>     <chr>     <chr>             <chr> <chr>           <chr>
## 1 ENGWAL    K04000001 ENGLAND AND WALES All   278570          1201080
## 2 ENGWAL    K04000001 ENGLAND AND WALES A     178680          142960
## 3 ENGWAL    K04000001 ENGLAND AND WALES B     61140           265940
## 4 ENGWAL    K04000001 ENGLAND AND WALES C     26040           403180
## 5 ENGWAL    K04000001 ENGLAND AND WALES D     8170            248420
## 6 ENGWAL    K04000001 ENGLAND AND WALES E     3240            106430
## # i 32 more variables: TYPE_BUNGALOW_3 <chr>, TYPE_BUNGALOW_4 <chr>,
## #   TYPE_BUNGALOW_UNKW <chr>, BUNGALOW <chr>, TYPE_FLAT_MAIS_1 <chr>,
## #   TYPE_FLAT_MAIS_2 <chr>, TYPE_FLAT_MAIS_3 <chr>, TYPE_FLAT_MAIS_4 <chr>,
## #   TYPE_FLAT_MAIS_UNKW <chr>, FLAT_MAIS <chr>, TYPE_HOUSE_TERRACED_1 <chr>,
## #   TYPE_HOUSE_TERRACED_2 <chr>, TYPE_HOUSE_TERRACED_3 <chr>,
## #   TYPE_HOUSE_TERRACED_4 <chr>, TYPE_HOUSE_TERRACED_UNKW <chr>,
## #   HOUSE_TERRACED <chr>, TYPE_HOUSE_SEMI_1 <chr>, TYPE_HOUSE_SEMI_2 <chr>, ...
```

```r
# Display the property metadata
property_metadata
```

```
## # A tibble: 40 x 2
##    Variable   `Variable Description`
##    <chr>      <chr>
##  1 CODE       Unique identifier for administrative geographies as specified by ~
##  2 GEOG       Indicates the geographic level for which data are presented
##  3 AREA       Administrative area name
##  4 BAND       Council Tax Band
##  5 Bungalow1  Count of bungalows with one bedroom
##  6 Bungalow2  Count of bungalows with two bedrooms
##  7 Bungalow3  Count of bungalows with three or more bedrooms
##  8 BungalowZ  Count of bungalows where the number of bedrooms are unknown
##  9 Flat_Mais1 Count of Purpose built and converted flats/maisonettes with one b~
## 10 Flat_Mais2 Count of Purpose built and converted flats/maisonettes with two b~
## # i 30 more rows
```

Observation

1. There are 4 main categories of housing in the data set.
2. Each type is segregated by the tax band and the number of bedrooms.

Conclusion The property data set is useful for understanding the housing stock in London. This data set can be used to understand the **relationship between the housing stock and the food categories purchased**. Since **housing type is a good indicator for affluence**, it might not be a stretch to imagine it having a relationship with food categories purchased. For example, **a higher number of detached houses might correlate with higher spending on luxury food items.**

Research Question

Is it possible to **optimize a Tesco Store's inventory** by modeling the relationship between **property type and buying behaviors** of its residents?

Hypothesis

The hypothesis is that **there is a relationship between the property type and the food categories purchased.** For example, residents living in detached houses might purchase more luxury food items compared to residents living in flats.

Methodology

1. Clean and preprocess the data.
2. Merge the Tesco grocery data with the property data.
3. Analyze the relationship between the property type and the food categories purchased.
4. Identify patterns and trends in the data.
5. Develop a model to predict food categories purchased based on the property distribution.
6. Evaluate the model's performance and interpret the results.
7. Provide recommendations for Tesco based on the findings.

Assumptions

1. Property type is an indicator of the residents' socio-economic status.
2. Consumer spending habits of the residents (in this case the card members) are consistent over the years.
3. The Tesco grocery data is representative of the general population's buying behavior.

Data Cleaning - Property

The code below cleans the property data set and checks what percentage of the grocery data is a subset of the property data.

```
# Remove columns with the prefix 'TYPE'
property_cleaned <- property %>%
  select(-starts_with("TYPE"))
```

```
# Rename column
property_cleaned_lsoa <- property %>%
  rename(area_id = ECODE)
```

```r
# Extract lsoa rows from column 'geography'
property_cleaned_lsoa <- property_cleaned %>%
  filter(str_detect(GEOGRAPHY, "LSOA"))

# If 'ALL_PROPERTIES' is null or 0, remove it
property_cleaned_lsoa <- property_cleaned_lsoa %>%
  filter(!is.na(ALL_PROPERTIES) & ALL_PROPERTIES != 0)

# Remove duplicates
property_cleaned_lsoa <- distinct(property_cleaned_lsoa)

# Count of number of rows
nrow(property_cleaned_lsoa)
```

```
## [1] 36430
```

```r
# Extract msoa rows from column 'geography'
property_cleaned_msoa <- property_cleaned %>%
  filter(str_detect(GEOGRAPHY, "MSOA"))

# If 'ALL_PROPERTIES' is null or 0, remove it
property_cleaned_msoa <- property_cleaned_msoa %>%
  filter(!is.na(ALL_PROPERTIES) & ALL_PROPERTIES != 0)

# Remove duplicates
property_cleaned_msoa <- distinct(property_cleaned_msoa)

# Count of number of rows
nrow(property_cleaned_msoa)
```

```
## [1] 8431
```

```r
# Check the dimensions of lsoa and msoa April data frames
cat("Dimensions of LSOA April:", dim(lsoa_april), "\n")
```

```
## Dimensions of LSOA April: 4272 202
```

```r
cat("Dimensions of MSOA April:", dim(msoa_april), "\n")
```

```
## Dimensions of MSOA April: 981 202
```

```r
# Calculate and display the percentages to 2 decimal places
lsoa_percentage <- round(dim(lsoa_april)[1] / nrow(property_cleaned_lsoa) * 100, 2)
msoa_percentage <- round(dim(msoa_april)[1] / nrow(property_cleaned_msoa) * 100, 2)

cat("LSOA April as a percentage:", lsoa_percentage, "%\n")
```

```
## LSOA April as a percentage: 11.73 %
```

```r
cat("MSOA April as a percentage:", msoa_percentage, "%\n")
```

```
## MSOA April as a percentage: 11.64 %
```

Observation

1. The property data set has been cleaned and the relevant columns have been selected.
2. The grocery data set based on area code, **about 11%** (for both), of the property dataset. **Meaning there are more property data than grocery data, and not every area code has a tesco grocery store.**

Conclusion

Thus we **select the lsoa grocery data sets for its smaller granularity**, and continue doing data cleaning for that series.

```r
# Rename ECODE to area_id
property_cleaned_lsoa <- property_cleaned_lsoa %>%
  rename(area_id = ECODE)
```

```r
# Drop rows where BAND is not equal to "All"
property_cleaned_lsoa <- property_cleaned_lsoa %>%
  filter(BAND == "All")
```

```r
# Drop Columns
property_cleaned_lsoa <- property_cleaned_lsoa %>%
  select(-c(GEOGRAPHY, ALL_PROPERTIES, BAND))
```

```r
# Print data type of columns
sapply(property_cleaned_lsoa, class)
```

```
##        area_id      AREA_NAME       BUNGALOW      FLAT_MAIS HOUSE_TERRACED
##    "character"    "character"    "character"    "character"    "character"
##     HOUSE_SEMI HOUSE_DETACHED         ANNEXE          OTHER        UNKNOWN
##    "character"    "character"    "character"    "character"    "character"
```

```r
suppressWarnings({
  # Convert specific columns to numeric
  property_cleaned_lsoa[, c("BUNGALOW", "FLAT_MAIS", "HOUSE_TERRACED", "HOUSE_SEMI", "HOUSE_DETACHED",

  # Fill NA in specific columns with 0
  property_cleaned_lsoa[, c("BUNGALOW", "FLAT_MAIS", "HOUSE_TERRACED", "HOUSE_SEMI", "HOUSE_DETACHED",
})
```

Feature Engineering - Property Distribution

```r
# Create new column 'Total', which is the sum of property columns
property_cleaned_lsoa$Total <- rowSums(property_cleaned_lsoa[, c("BUNGALOW", "FLAT_MAIS", "HOUSE_TERRAC

# Calculate the percentage of each column and save as new columns
property_cleaned_lsoa <- property_cleaned_lsoa %>%
  mutate(
    BUNGALOW_perc = BUNGALOW / Total * 100,
    FLAT_MAIS_perc = FLAT_MAIS / Total * 100,
    HOUSE_TERRACED_perc = HOUSE_TERRACED / Total * 100,
    HOUSE_SEMI_perc = HOUSE_SEMI / Total * 100,
    HOUSE_DETACHED_perc = HOUSE_DETACHED / Total * 100,
    ANNEXE_perc = ANNEXE / Total * 100,
    OTHER_perc = OTHER / Total * 100,
    UNKNOWN_perc = UNKNOWN / Total * 100
  )
```

```r
# Identify numeric columns
numeric_cols <- sapply(property_cleaned_lsoa, is.numeric)

# Replace NaN values with 0 in numeric columns
property_cleaned_lsoa[, numeric_cols] <- lapply(property_cleaned_lsoa[, numeric_cols], function(x) repla
```

Display Cleaned Dataset

```r
head(property_cleaned_lsoa)
```

```
## # A tibble: 6 x 19
##   area_id  AREA_NAME BUNGALOW FLAT_MAIS HOUSE_TERRACED HOUSE_SEMI HOUSE_DETACHED
##   <chr>    <chr>        <dbl>     <dbl>          <dbl>      <dbl>          <dbl>
## 1 E010000~ City of ~        0      1090             10          0              0
## 2 E010000~ City of ~        0      1140             50          0              0
## 3 E010000~ City of ~        0       910              0          0              0
## 4 E010000~ City of ~        0       680              0          0              0
## 5 E010000~ Barking ~        0       150            380          0              0
## 6 E010000~ Barking ~        0       790            150          0              0
## # i 12 more variables: ANNEXE <dbl>, OTHER <dbl>, UNKNOWN <dbl>, Total <dbl>,
## #   BUNGALOW_perc <dbl>, FLAT_MAIS_perc <dbl>, HOUSE_TERRACED_perc <dbl>,
## #   HOUSE_SEMI_perc <dbl>, HOUSE_DETACHED_perc <dbl>, ANNEXE_perc <dbl>,
## #   OTHER_perc <dbl>, UNKNOWN_perc <dbl>
```

```r
dim(property_cleaned_lsoa)
```

```
## [1] 4835   19
```

Conclusion

As part of feature engineering, the percentage of housing type for each area was calculated. The property data set has been cleaned and the relevant columns have been selected. The property distribution has been calculated and the data set is ready for analysis. **The next step would be to clean the Tesco grocery data set and join it with the property data set.**

Data Cleaning - Tesco

Approach

1. Filter for the relevant columns.
2. Check for null values, and remove or impute them if necessary.
3. Check for duplicates and remove them if necessary.
4. Standardize the column names.

```
# Define the columns to select
columns_to_select <- c(
  "area_id",
  "f_beer",
  "f_dairy",
  "f_eggs",
  "f_fats_oils",
  "f_fish",
  "f_fruit_veg",
  "f_grains",
  "f_meat_red",
  "f_poultry",
  "f_readymade",
  "f_sauces",
  "f_soft_drinks",
  "f_spirits",
  "f_sweets",
  "f_tea_coffee",
  "f_water",
  "f_wine",
  "population",
  "male",
  "female",
  "age_0_17",
  "age_18_64",
  "age_65+",
  "avg_age",
  "area_sq_km",
  "people_per_sq_km"
)
```

```
# Named list of all data frames by month
monthly_data_frames <- list(
  lsoa_january = lsoa_january,
  lsoa_february = lsoa_february,
  lsoa_march = lsoa_march,
  lsoa_april = lsoa_april,
  lsoa_may = lsoa_may,
  lsoa_june = lsoa_june,
  lsoa_july = lsoa_july,
  lsoa_august = lsoa_august,
  lsoa_september = lsoa_september,
  lsoa_october = lsoa_october,
  lsoa_november = lsoa_november,
```

```r
  lsoa_december = lsoa_december
)
```

```r
# Write a function for selecting columns
select_columns <- function(data, columns) {
  selected_data <- data %>%
    select(all_of(columns))

  return(selected_data)
}
```

```r
# Use map to iterate over the list, apply the function, and add month numbers
cleaned_monthly_data_with_month <- purrr::imap(monthly_data_frames, function(data, month_name) {
  # Apply column selection
  cleaned_data <- select_columns(data, columns_to_select)

  # Assign month number based on position in list
  month_number <- match(month_name, names(monthly_data_frames))
  cleaned_data$month <- month_number

  return(cleaned_data)
})
```

```r
# Combine all cleaned monthly data with month numbers into a single dataframe
lsoa_grocery_data <- bind_rows(cleaned_monthly_data_with_month)
```

```r
# Display the first few rows and the dimensions of the combined data set
head(lsoa_grocery_data)
```

```
## # A tibble: 6 x 28
##   area_id    f_beer f_dairy  f_eggs f_fats_oils f_fish f_fruit_veg f_grains
##   <chr>       <dbl>   <dbl>   <dbl>       <dbl>  <dbl>       <dbl>    <dbl>
## 1 E01000001 0.0123   0.154  0.00924      0.0220 0.0236       0.333    0.109
## 2 E01000002 0.00806  0.114  0.00967      0.0182 0.0262       0.366    0.115
## 3 E01000003 0.0220   0.139  0.00917      0.0242 0.0147       0.280    0.121
## 4 E01000005 0.0124   0.126  0.0121       0.0248 0.0203       0.288    0.134
## 5 E01000006 0.0133   0.110  0.0139       0.0316 0.0213       0.264    0.160
## 6 E01000007 0.00431  0.0982 0.0134       0.0307 0.0282       0.231    0.165
## # i 20 more variables: f_meat_red <dbl>, f_poultry <dbl>, f_readymade <dbl>,
## #   f_sauces <dbl>, f_soft_drinks <dbl>, f_spirits <dbl>, f_sweets <dbl>,
## #   f_tea_coffee <dbl>, f_water <dbl>, f_wine <dbl>, population <dbl>,
## #   male <dbl>, female <dbl>, age_0_17 <dbl>, age_18_64 <dbl>, ‘age_65+‘ <dbl>,
## #   avg_age <dbl>, area_sq_km <dbl>, people_per_sq_km <dbl>, month <int>
```

Plotting the distribution of Tesco Stores in London

```r
# Import the shapefile
suppressMessages({
  london_map <- st_read(london_map_path)
})
```

```
## Reading layer `LSOA_2011_London_gen_MHW' from data source
##   `C:\Users\colin\OneDrive\Desktop\data science mod\working\data\assignment_2\LSOA_2011_London_gen_MI
##   using driver `ESRI Shapefile'
## Simple feature collection with 4835 features and 14 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 503574.2 ymin: 155850.8 xmax: 561956.7 ymax: 200933.6
## Projected CRS: OSGB36 / British National Grid
```

```r
# Display the first few rows of the imported shapefile
head(london_map)
```

```
## Simple feature collection with 6 features and 14 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 531948.3 ymin: 180733.9 xmax: 545296.3 ymax: 184700.6
## Projected CRS: OSGB36 / British National Grid
##    LSOA11CD                  LSOA11NM  MSOA11CD                 MSOA11NM
## 1 E01000001       City of London 001A E02000001       City of London 001
## 2 E01000002       City of London 001B E02000001       City of London 001
## 3 E01000003       City of London 001C E02000001       City of London 001
## 4 E01000005       City of London 001E E02000001       City of London 001
## 5 E01000006 Barking and Dagenham 016A E02000017 Barking and Dagenham 016
## 6 E01000007 Barking and Dagenham 015A E02000016 Barking and Dagenham 015
##     LAD11CD              LAD11NM   RGN11CD RGN11NM USUALRES HHOLDRES COMESTRES
## 1 E09000001       City of London E12000007  London     1465     1465         0
## 2 E09000001       City of London E12000007  London     1436     1436         0
## 3 E09000001       City of London E12000007  London     1346     1250        96
## 4 E09000001       City of London E12000007  London      985      985         0
## 5 E09000002 Barking and Dagenham E12000007  London     1703     1699         4
## 6 E09000002 Barking and Dagenham E12000007  London     1391     1391         0
##   POPDEN HHOLDS AVHHOLDSZ                       geometry
## 1  112.9    876       1.7 MULTIPOLYGON (((532105.1 18...
## 2   62.9    830       1.7 MULTIPOLYGON (((532746.8 18...
## 3  227.7    817       1.5 MULTIPOLYGON (((532135.1 18...
## 4   52.0    467       2.1 MULTIPOLYGON (((533807.9 18...
## 5  116.2    543       3.1 MULTIPOLYGON (((545122 1843...
## 6   69.6    612       2.3 MULTIPOLYGON (((544180.3 18...
```

Observation

From the london map, it looks like GSS_CODE is the column containing the area codes that we can join the datasets on.

```r
# Rename GSS_CODE to area_id

london_map <- london_map %>%
  rename(area_id = LSOA11CD)
```

```r
# Count number of row
nrow(london_map)
```

```
## [1] 4835
```

```r
# Check for similar rows in both data sets london_map and lsoa_grocery_data based on the row area_id
common_area_ids <- intersect(london_map$area_id, lsoa_grocery_data$area_id)

# Count the number of common area ids
length(common_area_ids)
```

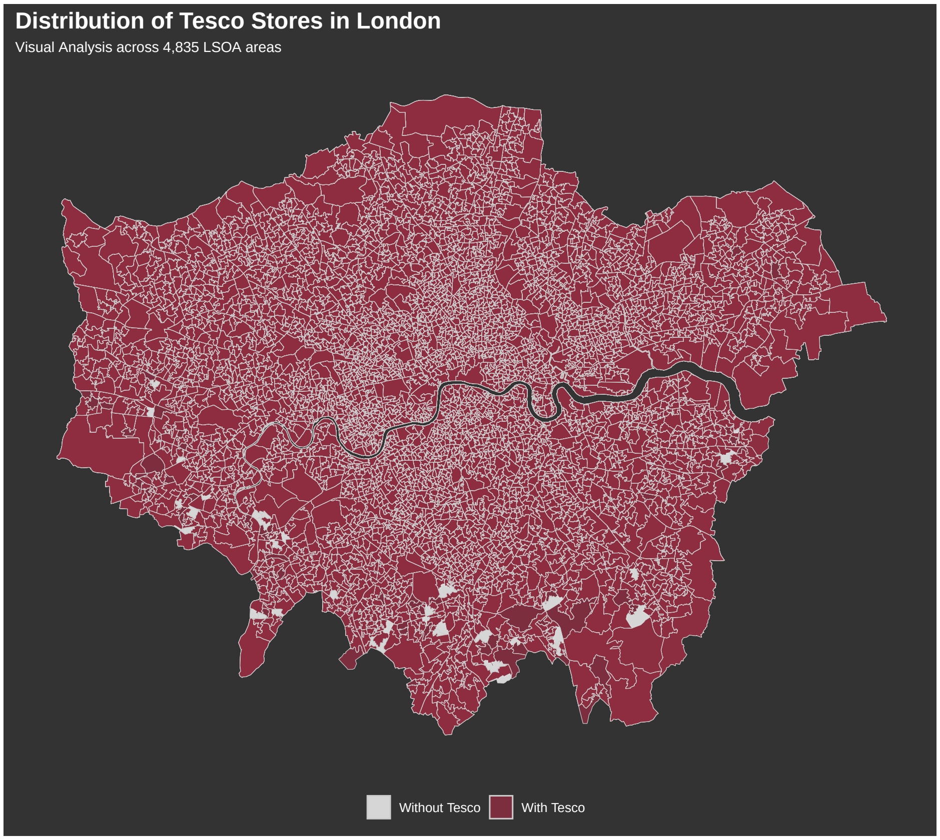
```
## [1] 4799
```

```r
# Number of rows not overlapping
nrow(london_map) - length(common_area_ids)
```

```
## [1] 36
```

```r
# Merge the datasets, marking areas with Tesco stores
merged_df <- london_map %>%
  left_join(lsoa_grocery_data %>% mutate(has_tesco = TRUE), by = "area_id") %>%
  replace_na(list(has_tesco = FALSE))  # Areas not in lsoa_grocery_data are marked as FALSE
```

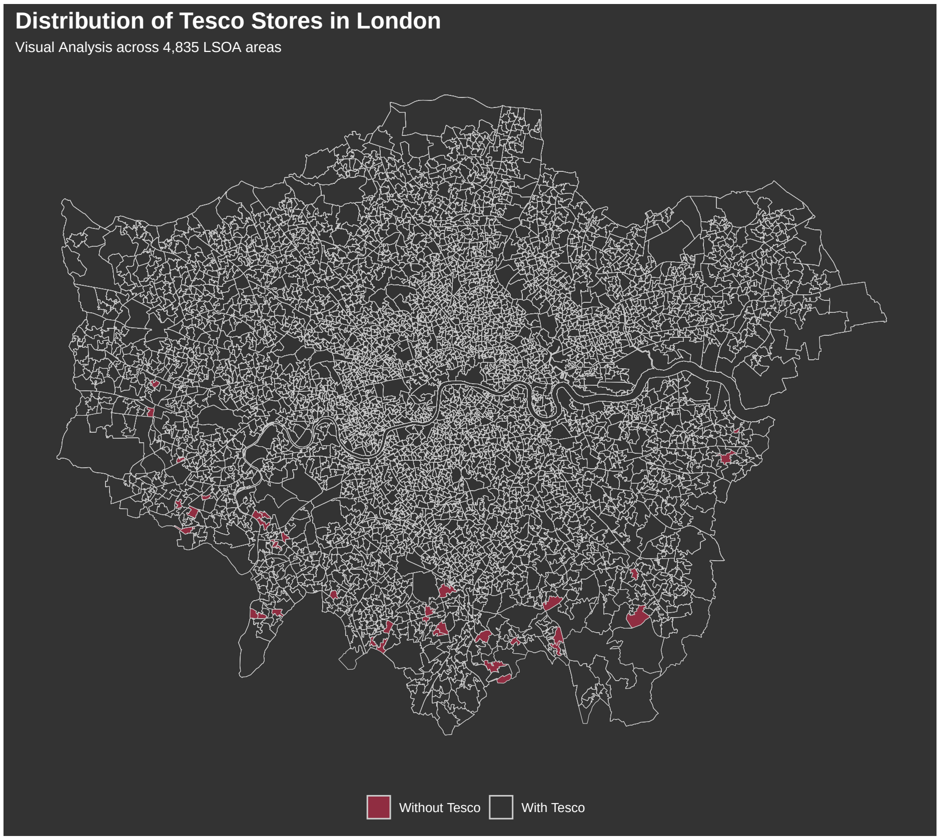
```r
ggplot(data = merged_df) +
  geom_sf(aes(fill = has_tesco), color = "#CCCCCC", size = 0.01, alpha = 0.8) +
  scale_fill_manual(values = c("TRUE" = "#902D41", "FALSE" = "#FFFFFF"),
                    name = "",
                    labels = c("TRUE" = "With Tesco", "FALSE" = "Without Tesco")) +
  labs(title = "Distribution of Tesco Stores in London",
       subtitle = "Visual Analysis across 4,835 LSOA areas") +
  theme_minimal() +
  theme(legend.position = "bottom",
        legend.background = element_blank(),  # Remove legend border
        plot.title = element_text(size = 16, face = "bold", color = "white"),
        plot.subtitle = element_text(size = 10, color = "white"),
        plot.background = element_rect(fill = "#333333", color = NA),
        panel.background = element_rect(fill = "#333333", color = NA),
        axis.title = element_blank(),  # Remove axis titles
        axis.text = element_blank(),   # Remove axis text
        legend.text = element_text(color = "white"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank()
      )
```



Observation

The majority of London areas has a Tesco store. Thus **plotting the inverse might make the map more readable.**

```
ggplot(data = merged_df) +
  geom_sf(aes(fill = has_tesco), color = "#CCCCCC", size = 0.001, alpha = 1) +
  scale_fill_manual(values = c("TRUE" = "#333333", "FALSE" = "#902D41"),
                    name = "",
                    labels = c("TRUE" = "With Tesco", "FALSE" = "Without Tesco")) +
  labs(title = "Distribution of Tesco Stores in London",
       subtitle = "Visual Analysis across 4,835 LSOA areas") +
  theme_minimal() +
  theme(legend.position = "bottom",
        legend.background = element_blank(),  # Remove legend border
        plot.title = element_text(size = 16, face = "bold", color = "white"),
        plot.subtitle = element_text(size = 10, color = "white"),
        plot.background = element_rect(fill = "#333333", color = NA),
        panel.background = element_rect(fill = "#333333", color = NA),
        axis.title = element_blank(),  # Remove axis titles
        axis.text = element_blank(),   # Remove axis text
        legend.text = element_text(color = "white"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank()
      )
```



Observation

The map shows that **most areas in London have a Tesco store**, and it is the **southern parts of London that do not have Tesco stores**, perhaps dominated by a different supermarket chain.

Conclusion

The Tesco grocery data set has been cleaned and the relevant columns have been selected. The data set is ready for analysis. **The next step would be to join the grocery data with the property data.**

Data Cleaning - Join Data sets

The code below joins the grocery data with the property data.

```
# Join the grocery data with the property data
lsoa_grocery_property <- left_join(lsoa_grocery_data, property_cleaned_lsoa, by = "area_id")
```

```
# Area name Column
lsoa_grocery_property <- lsoa_grocery_property %>%
  select(area_id, AREA_NAME, everything()) %>%
  mutate(AREA_NAME = substr(AREA_NAME, 1, nchar(AREA_NAME) - 5))
```

```r
# Print all unique values of column 'AREA_NAME'
unique(lsoa_grocery_property$AREA_NAME)
```

```
##  [1] "City of London"         "Barking and Dagenham"   "Barnet"
##  [4] "Bexley"                 "Brent"                  "Bromley"
##  [7] "Camden"                 "Croydon"                "Ealing"
## [10] "Enfield"                "Greenwich"              "Hackney"
## [13] "Hammersmith and Fulham" "Haringey"               "Harrow"
## [16] "Havering"               "Hillingdon"             "Hounslow"
## [19] "Islington"              "Kensington and Chelsea" "Kingston upon Thames"
## [22] "Lambeth"                "Lewisham"               "Merton"
## [25] "Newham"                 "Redbridge"              "Richmond upon Thames"
## [28] "Southwark"              "Sutton"                 "Tower Hamlets"
## [31] "Waltham Forest"         "Wandsworth"             "Westminster"
```

```r
# Define a named vector to rename columns
new_column_names <- c(
  "area_id" = "area_id",
  "AREA_NAME" = "area_name",
  "f_beer" = "beer",
  "f_dairy" = "dairy",
  "f_eggs" = "eggs",
  "f_fats_oils" = "fatty_oils",
  "f_fish" = "fish",
  "f_fruit_veg" = "fruit_veg",
  "f_grains" = "grains",
  "f_meat_red" = "red_meat",
  "f_poultry" = "poultry",
  "f_readymade" = "readymade",
  "f_sauces" = "sauces",
  "f_soft_drinks" = "soft_drinks",
  "f_spirits" = "spirits",
  "f_sweets" = "sweets",
  "f_tea_coffee" = "tea_coffee",
  "f_water" = "water",
  "f_wine" = "wine",
  "population" = "population",
  "male" = "male",
  "female" = "female",
  "age_0_17" = "children",
  "age_18_64" = "adult",
  "age_65+" = "senior",
  "avg_age" = "average_age",
  "area_sq_km" = "area_sq_km",
  "people_per_sq_km" = "people_per_sq_km",
  "month" = "month",
  "BUNGALOW" = "bungalow",
  "FLAT_MAIS" = "masionette",
  "HOUSE_TERRACED" = "terrace",
  "HOUSE_SEMI" = "semi_detached",
  "HOUSE_DETACHED" = "detached",
  "ANNEXE" = "annexe",
  "OTHER" = "other",
```

```r
  "UNKNOWN" = "unknown",
  "Total" = "total",
  "BUNGALOW_perc" = "bungalow_perc",
  "FLAT_MAIS_perc" = "masionette_perc",
  "HOUSE_TERRACED_perc" = "terrace_perc",
  "HOUSE_SEMI_perc" = "semi_detached_perc",
  "HOUSE_DETACHED_perc" = "detached_perc",
  "ANNEXE_perc" = "annexe_perc",
  "OTHER_perc" = "other_perc",
  "UNKNOWN_perc" = "unknown_perc"
)
```

```r
# Rename the columns
names(lsoa_grocery_property) <- new_column_names
```

```r
# If data type of column is numeric, round to 3 decimal places
lsoa_grocery_property <- lsoa_grocery_property %>%
  mutate_if(is.numeric, ~round(., 3))
```

```r
# Drop Columns
lsoa_grocery_property <- lsoa_grocery_property %>%
  select(-area_id)
```

```r
# Convert area_name to factor
lsoa_grocery_property$area_name <- as.factor(lsoa_grocery_property$area_name)
```

```r
#head
head(lsoa_grocery_property)
```

```
## # A tibble: 6 x 45
##   area_name  beer dairy  eggs fatty_oils  fish fruit_veg grains red_meat poultry
##   <fct>     <dbl> <dbl> <dbl>      <dbl> <dbl>     <dbl>  <dbl>    <dbl>   <dbl>
## 1 City of ~ 0.012 0.154 0.009      0.022 0.024     0.333  0.109    0.048   0.018
## 2 City of ~ 0.008 0.114 0.01       0.018 0.026     0.366  0.115    0.051   0.019
## 3 City of ~ 0.022 0.139 0.009      0.024 0.015     0.28   0.121    0.051   0.015
## 4 City of ~ 0.012 0.126 0.012      0.025 0.02      0.288  0.134    0.05    0.021
## 5 Barking ~ 0.013 0.11  0.014      0.032 0.021     0.264  0.16     0.048   0.026
## 6 Barking ~ 0.004 0.098 0.013      0.031 0.028     0.231  0.165    0.037   0.021
## # i 35 more variables: readymade <dbl>, sauces <dbl>, soft_drinks <dbl>,
## #   spirits <dbl>, sweets <dbl>, tea_coffee <dbl>, water <dbl>, wine <dbl>,
## #   population <dbl>, male <dbl>, female <dbl>, children <dbl>, adult <dbl>,
## #   senior <dbl>, average_age <dbl>, area_sq_km <dbl>, people_per_sq_km <dbl>,
## #   month <dbl>, bungalow <dbl>, masionette <dbl>, terrace <dbl>,
## #   semi_detached <dbl>, detached <dbl>, annexe <dbl>, other <dbl>,
## #   unknown <dbl>, total <dbl>, bungalow_perc <dbl>, masionette_perc <dbl>, ...
```

```r
# Reorder columns
lsoa_grocery_property <- lsoa_grocery_property %>%
  select(area_name, month, population:unknown_perc, everything())
```

```r
# Final check for null values
lsoa_grocery_property %>%
  summarise_all(~sum(is.na(.)))
```

```
## # A tibble: 1 x 45
##   area_name month population  male female children adult senior average_age
##       <int> <int>      <int> <int>  <int>    <int> <int>  <int>       <int>
## 1         0     0          0     0      0        0     0      0           0
## # i 36 more variables: area_sq_km <int>, people_per_sq_km <int>,
## #   bungalow <int>, masionette <int>, terrace <int>, semi_detached <int>,
## #   detached <int>, annexe <int>, other <int>, unknown <int>, total <int>,
## #   bungalow_perc <int>, masionette_perc <int>, terrace_perc <int>,
## #   semi_detached_perc <int>, detached_perc <int>, annexe_perc <int>,
## #   other_perc <int>, unknown_perc <int>, beer <int>, dairy <int>, eggs <int>,
## #   fatty_oils <int>, fish <int>, fruit_veg <int>, grains <int>, ...
```

```r
# Final Data frame 1 for machine learning
lsoa_grocery_property
```

```
## # A tibble: 51,295 x 45
##    area_name     month population  male female children adult senior average_age
##    <fct>         <dbl>      <dbl> <dbl>  <dbl>    <dbl> <dbl>  <dbl>       <dbl>
##  1 City of Lond~     1       1296   685    611      179   766    351        48.3
##  2 City of Lond~     1       1156   616    540      197   656    303        47.4
##  3 City of Lond~     1       1350   713    637      152   850    348        48.4
##  4 City of Lond~     1       1121   604    517      294   675    152        35.6
##  5 Barking and ~     1       2040  1040   1000      563  1317    160        32.1
##  6 Barking and ~     1       2101   999   1102      653  1380     68        27.4
##  7 Barking and ~     1       1566   818    748      582   938     46        27.3
##  8 Barking and ~     1       1775   957    818      387  1229    159        34.3
##  9 Barking and ~     1       3195  1732   1463      878  2225     92        28.0
## 10 Barking and ~     1       1670   888    782      443  1097    130        32.0
## # i 51,285 more rows
## # i 36 more variables: area_sq_km <dbl>, people_per_sq_km <dbl>,
## #   bungalow <dbl>, masionette <dbl>, terrace <dbl>, semi_detached <dbl>,
## #   detached <dbl>, annexe <dbl>, other <dbl>, unknown <dbl>, total <dbl>,
## #   bungalow_perc <dbl>, masionette_perc <dbl>, terrace_perc <dbl>,
## #   semi_detached_perc <dbl>, detached_perc <dbl>, annexe_perc <dbl>,
## #   other_perc <dbl>, unknown_perc <dbl>, beer <dbl>, dairy <dbl>, ...
```

Conclusion

The data cleaning process has been completed and both dataset has been joined and is ready for analysis. **The next step would be to analyze the relationship between the property type and the food categories purchased. This will be done using machine learning techniques involving multi-label regression prediction.** The model will predict the food categories purchased based on the property distribution. The model's performance will be evaluated and the results will be interpreted. Recommendations will be provided for Tesco based on the findings.

ML 1 - Data Preparation

This section involves splitting the data into training and testing sets, encoding the categorical variables, and preparing the data for model training.

```
# Split the data into training and testing sets
set.seed(123)
```

```
# Function to prepare training and testing sets

prepare_training_testing_sets <- function(data, num_targets, train_prop = 0.8) {
  # Split data into input features and targets based on num_targets
  input_features <- data[, 1:(ncol(data) - num_targets)]
  targets <- data[, (ncol(data) - num_targets + 1):ncol(data)]

  # Convert factor variables in input features to dummy variables
  input_features_encoded <- model.matrix(~ . - 1, data = input_features)

  # Print column names of the encoded input features
  cat("Column names of encoded input features:\n")
  print(colnames(input_features_encoded))

  # Calculate the number of rows to sample for training data
  train_size <- floor(train_prop * nrow(input_features_encoded))

  # Randomly sample row indices for the training data
  train_indices <- sample(seq_len(nrow(input_features_encoded)), size = train_size)

  # Create training and testing sets
  train_data <- input_features_encoded[train_indices, ]
  train_targets <- targets[train_indices, ]
  test_data <- input_features_encoded[-train_indices, ]
  test_targets <- targets[-train_indices, ]

  # Print the dimensions of the training and testing sets
  cat("\nDimensions of the training set:\n")
  print(dim(train_data))

  cat("\nDimensions of the testing set:\n")
  print(dim(test_data))

  # Return a list containing the training and testing sets
  return(list(train_data = train_data, train_targets = train_targets,
              test_data = test_data, test_targets = test_targets))
}
```

```
# Call the function to prepare the training and testing sets
results <- prepare_training_testing_sets(lsoa_grocery_property, num_targets = 17, train_prop = 0.8)
```

```
## Column names of encoded input features:
##  [1] "area_nameBarking and Dagenham"   "area_nameBarnet"
##  [3] "area_nameBexley"                 "area_nameBrent"
##  [5] "area_nameBromley"                "area_nameCamden"
```

```
##  [7] "area_nameCity of London"         "area_nameCroydon"
##  [9] "area_nameEaling"                 "area_nameEnfield"
## [11] "area_nameGreenwich"              "area_nameHackney"
## [13] "area_nameHammersmith and Fulham" "area_nameHaringey"
## [15] "area_nameHarrow"                 "area_nameHavering"
## [17] "area_nameHillingdon"             "area_nameHounslow"
## [19] "area_nameIslington"              "area_nameKensington and Chelsea"
## [21] "area_nameKingston upon Thames"   "area_nameLambeth"
## [23] "area_nameLewisham"               "area_nameMerton"
## [25] "area_nameNewham"                 "area_nameRedbridge"
## [27] "area_nameRichmond upon Thames"   "area_nameSouthwark"
## [29] "area_nameSutton"                 "area_nameTower Hamlets"
## [31] "area_nameWaltham Forest"         "area_nameWandsworth"
## [33] "area_nameWestminster"            "month"
## [35] "population"                      "male"
## [37] "female"                          "children"
## [39] "adult"                           "senior"
## [41] "average_age"                     "area_sq_km"
## [43] "people_per_sq_km"                "bungalow"
## [45] "masionette"                      "terrace"
## [47] "semi_detached"                   "detached"
## [49] "annexe"                          "other"
## [51] "unknown"                         "total"
## [53] "bungalow_perc"                   "masionette_perc"
## [55] "terrace_perc"                    "semi_detached_perc"
## [57] "detached_perc"                   "annexe_perc"
## [59] "other_perc"                      "unknown_perc"
##
## Dimensions of the training set:
## [1] 41036    60
##
## Dimensions of the testing set:
## [1] 10259    60
```

```r
# Accessing the training and testing sets
train_data <- results$train_data
train_targets <- results$train_targets
test_data <- results$test_data
test_targets <- results$test_target
```

Conclusion

The data has been split into training and testing sets, and the categorical variables have been encoded. The data is now ready for model training. **The next step would be to train the multi-label regression model.**

ML 1 - Model Training

This section involves training a **multi-label regression model using multiple random forest models and aggregating them** to predict the food categories purchased based on the property distribution.

This code chunk is commented out as it takes a long time to run. The trained models were saved locally on the author's pc. The models are too large to be cached.

```
# # Initialize a list to store models
# models <- list()
#
# # Initialize a vector to store training times
# training_times <- numeric(length = length(colnames(train_targets)))
#
# # Capture start time for total training time
# total_start_time <- Sys.time()
#
# # Loop through each target column
# for (i in seq_along(colnames(train_targets))) {
#   target_name <- colnames(train_targets)[i]
#
#   # Extract the current target column
#   current_target <- train_targets[[target_name]]
#
#   # Combine the current target with the train data
#   data_for_model <- cbind(current_target = current_target, train_data) # Ensure this column is correc
#
#   # Capture start time for individual model training
#   start_time <- Sys.time()
#
#   # Train a model for the current target
#   models[[target_name]] <- ranger(
#     dependent.variable.name = "current_target",
#     data = data_for_model,
#     num.trees = 500,
#     importance = 'impurity',
#     min.node.size = 5,
#     write.forest = TRUE
#   )
#
#   # Save the model
#   saveRDS(models[[target_name]], paste0("model_", target_name, ".rds"))
#
#   # Capture end time for individual model training
#   end_time <- Sys.time()
#   training_time <- end_time - start_time
#
#   # Store the training time
#   training_times[i] <- training_time
#
#   # Print the training time for the current model
#   cat("Time taken to train model for", target_name, ":", training_time, "seconds\n")
# }
#
# # Directly sum the individual training times for total
# summed_total_training_time <- sum(training_times)
#
# # Print the corrected total training time
# cat("Total training time:", summed_total_training_time, "seconds\n")
```

ML 1 - Model Evaluation

This section involves predicting the target variables on the test data and evaluating the model's performance using metrics such as RMSE. **RMSE** is selected as the metric because it allows for direct comparison of the model's performance across different target variables. Since the result is aggregarted, rmse allows for a single metric to evaluate the model's performance.

```r
# Initialize lists to store predictions and evaluation metrics
predictions <- list()
evaluation_metrics <- data.frame(Target=character(),
                                 RMSE=numeric(),
                                 stringsAsFactors=FALSE)
```

```r
# Loop through each target column to predict and evaluate
for (target_name in colnames(train_targets)) {
  # Load the trained model
  model <- readRDS(paste0("model_", target_name, ".rds"))

  # Predict on the test data
  prediction <- predict(model, data = test_data)$predictions
  predictions[[target_name]] <- prediction

  # Actual values
  actual <- test_targets[[target_name]]

  # Calculate RMSE
  rmse_val <- rmse(actual, prediction)

  # Store evaluation metrics
  evaluation_metrics <- rbind(evaluation_metrics,
                              data.frame(Target=target_name,
                                         RMSE=rmse_val))
}

# Print evaluation metrics summary
print(evaluation_metrics)
```

```
##         Target        RMSE
## 1         beer 0.006843340
## 2        dairy 0.011233429
## 3         eggs 0.002401449
## 4   fatty_oils 0.005923448
## 5         fish 0.005033080
## 6    fruit_veg 0.021369612
## 7       grains 0.014033841
## 8     red_meat 0.006925979
## 9      poultry 0.004005329
## 10   readymade 0.010089397
## 11      sauces 0.003891467
## 12 soft_drinks 0.011175970
## 13     spirits 0.003004328
## 14      sweets 0.019736465
## 15  tea_coffee 0.004493196
```

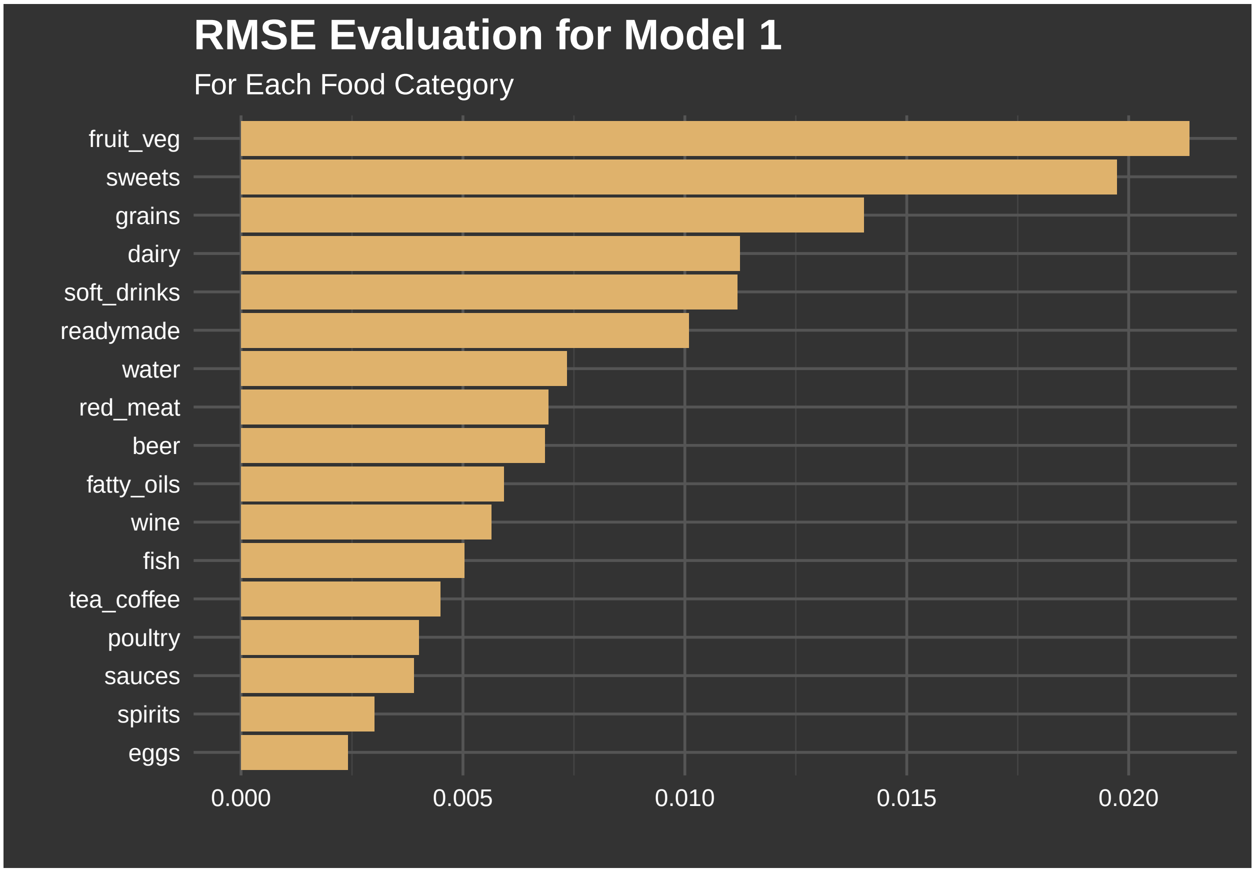
```
## 16       water 0.007342333
## 17        wine 0.005634684
```

```r
# Calculate the total RMSE
total_rmse <- sqrt(mean(evaluation_metrics$RMSE^2))

# Print the total RMSE
cat("Total RMSE:", total_rmse, "\n")
```

```
## Total RMSE: 0.01000903
```

```r
# Set the theme for a dark background
theme_dark_background <- theme_minimal(base_family = "sans") +
  theme(
    text = element_text(color = "white"),
    plot.background = element_rect(fill = "#333333", color = NA), # Dark grey background
    panel.background = element_rect(fill = "#333333", color = NA),
    axis.title = element_text(size = 14, color = "white"),
    axis.text = element_text(color = "white"),
    legend.background = element_rect(fill = "#333333"),
    legend.text = element_text(color = "white"),
    panel.grid.major = element_line(color = "#555555"),
    panel.grid.minor = element_line(color = "#444444"),
    plot.title = element_text(size = 16, face = "bold", color = "white")
  )
```

```r
# RMSE Plot
ggplot(evaluation_metrics, aes(x = reorder(Target, RMSE), y = RMSE)) +
  geom_bar(stat = "identity", fill = "#DFB26C") +
  coord_flip() +
  theme_minimal() +
  labs(title = "RMSE Evaluation for Model 1",
       subtitle = "For Each Food Category",
       x = "",
       y = "") +
  theme_dark_background
```



Observation

1. The RMSE values vary from a range of 0.002 to 0.02.
2. The **RMSE values are low, indicating that the model is performing well** in predicting the food categories purchased based on the property distribution.

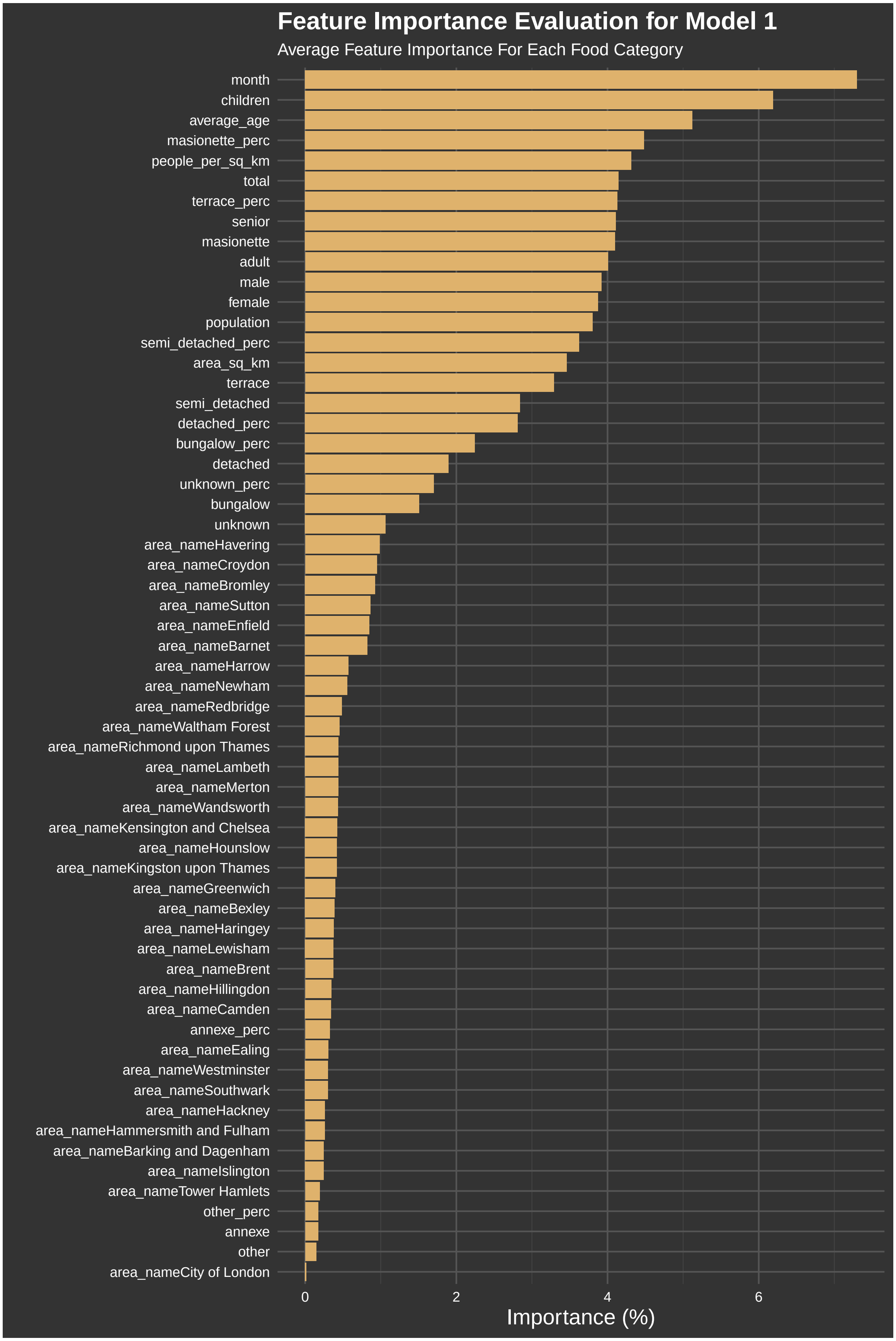
```r
# Initialize a list to store feature importance from each model
feature_importances <- list()

# Loop through each target column to load the model and calculate feature importance
for (target_name in colnames(train_targets)) {
  # Load the trained model
  model <- readRDS(paste0("model_", target_name, ".rds"))

  # Extract feature importance
  # NOTE: Adjust the method to extract importance according to your model type
  importance <- model$variable.importance

  # Normalize the feature importance to sum up to 1 (or 100%)
  importance_normalized <- importance / sum(importance)

  # Store the normalized importance in the list
  feature_importances[[target_name]] <- importance_normalized
}
```

```r
# Calculate the average feature importance across all models
# This step assumes that all models share the same features in the same order
# Convert the list to a matrix for easier column-wise operations
importance_matrix <- do.call("cbind", feature_importances)

# Calculate the mean importance for each feature across all models
average_importance <- rowMeans(importance_matrix)

# Optionally, convert to percentage
feature_importance_data <- average_importance * 100

# View the aggregated feature importance
print(feature_importance_data)
```

```
##   area_nameBarking and Dagenham                 area_nameBarnet
##                      0.24708808                      0.82417395
##                 area_nameBexley                  area_nameBrent
##                      0.38962707                      0.37105505
##                area_nameBromley                 area_nameCamden
##                      0.92666829                      0.34350065
##          area_nameCity of London                area_nameCroydon
##                      0.01790436                      0.94914827
##                 area_nameEaling                area_nameEnfield
##                      0.30472284                      0.85079641
##              area_nameGreenwich                area_nameHackney
##                      0.40030222                      0.26257383
## area_nameHammersmith and Fulham               area_nameHaringey
##                      0.25915984                      0.37739265
##                 area_nameHarrow               area_nameHavering
##                      0.57498881                      0.98529592
##             area_nameHillingdon               area_nameHounslow
##                      0.34611117                      0.42027036
##              area_nameIslington area_nameKensington and Chelsea
##                      0.24683864                      0.42617951
##   area_nameKingston upon Thames                area_nameLambeth
##                      0.41971813                      0.43959622
##               area_nameLewisham                 area_nameMerton
##                      0.37461029                      0.43836449
##                 area_nameNewham              area_nameRedbridge
##                      0.55798429                      0.48678108
##   area_nameRichmond upon Thames              area_nameSouthwark
##                      0.44062631                      0.30267385
##                 area_nameSutton          area_nameTower Hamlets
##                      0.86517576                      0.19271665
##         area_nameWaltham Forest             area_nameWandsworth
##                      0.45734979                      0.43320775
##            area_nameWestminster                           month
##                      0.30340701                      7.29816087
##                      population                            male
##                      3.80193919                      3.91881100
##                          female                        children
##                      3.87364309                      6.18972218
##                           adult                          senior
```

```
##                  4.00659121                  4.11067897
##                 average_age                 area_sq_km
##                  5.12377110                  3.46240971
##             people_per_sq_km                   bungalow
##                  4.31407776                  1.50946726
##                  masionette                     terrace
##                  4.10105107                  3.28905171
##                semi_detached                   detached
##                  2.84349164                  1.89675092
##                      annexe                      other
##                  0.17311983                  0.14672538
##                     unknown                      total
##                  1.06300581                  4.14485165
##                bungalow_perc             masionette_perc
##                  2.24467526                  4.48417681
##                 terrace_perc          semi_detached_perc
##                  4.12992946                  3.62615407
##                detached_perc                 annexe_perc
##                  2.81306292                  0.32558026
##                   other_perc                unknown_perc
##                  0.17333284                  1.69975843
```

```r
# Plot the feature importance
ggplot(data = data.frame(Feature = names(feature_importance_data), Importance = feature_importance_data
  geom_bar(stat = "identity", fill = "#DFB26C") +
  coord_flip() +
  theme_minimal() +
  labs(title = "Feature Importance Evaluation for Model 1",
       subtitle = "Average Feature Importance For Each Food Category",
       x = "",
       y = "Importance (%)") +
  theme_dark_background
```



Feature Importance Evaluation for Model 1
Average Feature Importance For Each Food Category
month
children
average_age
masionette_perc
people_per_sq_km
total
terrace_perc
senior
masionette
adult
male
female
population
semi_detached_perc
area_sq_km
terrace
semi_detached
detached_perc
bungalow_perc
detached
unknown_perc
bungalow
unknown
area_nameHavering
area_nameCroydon
area_nameBromley
area_nameSutton
area_nameEnfield
area_nameBarnet
area_nameHarrow
area_nameNewham
area_nameRedbridge
area_nameWaltham Forest
area_nameRichmond upon Thames
area_nameLambeth
area_nameMerton
area_nameWandsworth
area_nameKensington and Chelsea
area_nameHounslow
area_nameKingston upon Thames
area_nameGreenwich
area_nameBexley
area_nameHaringey
area_nameLewisham
area_nameBrent
area_nameHillingdon
area_nameCamden
annexe_perc
area_nameEaling
area_nameWestminster
area_nameSouthwark
area_nameHackney
area_nameHammersmith and Fulham
area_nameBarking and Dagenham
area_nameIslington
area_nameTower Hamlets
other_perc
annexe
other
area_nameCity of London
0
2
4
6
Importance (%)

Observation

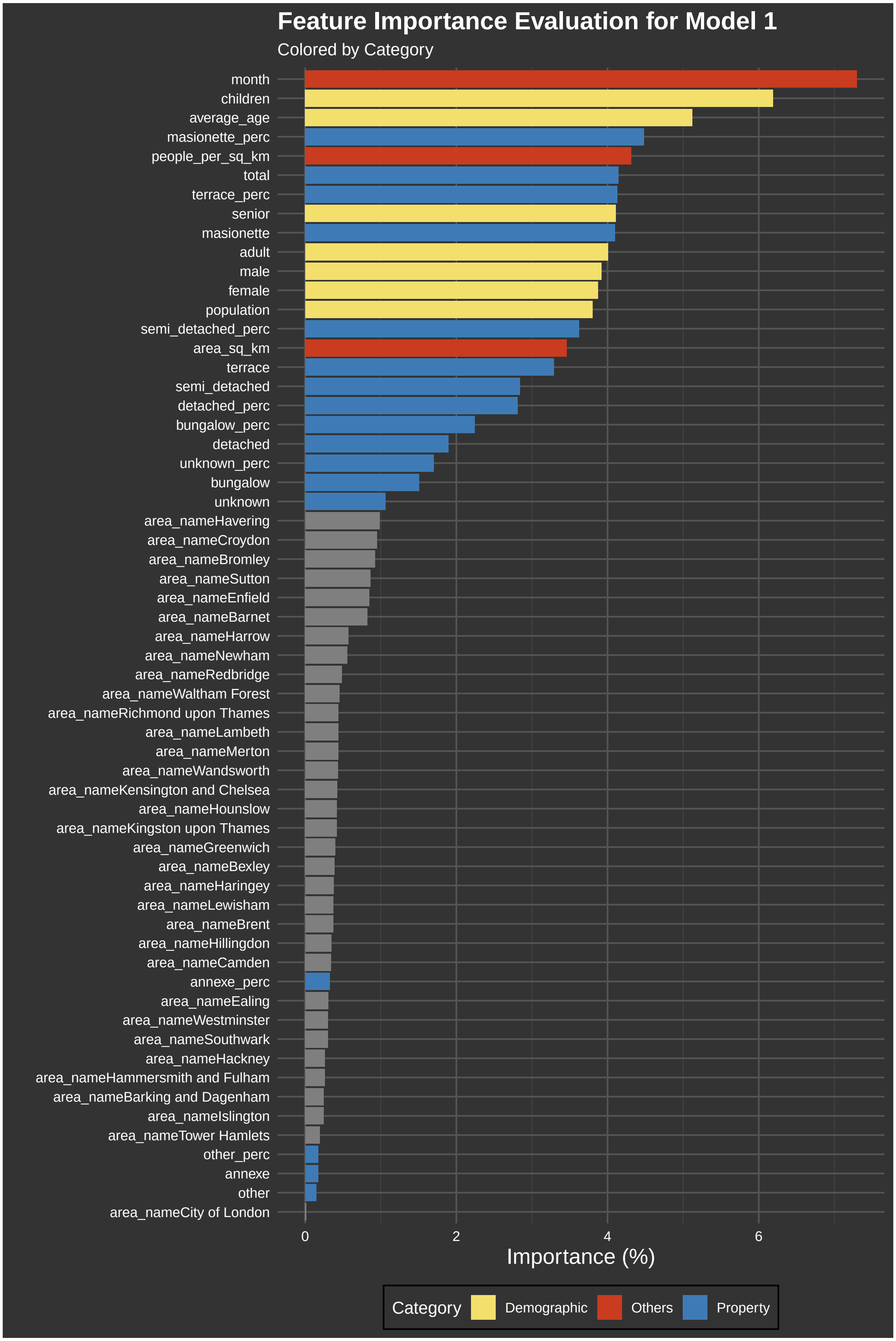
1. The **area name has almost no impact** on the food categories purchased. With all of them scoring less than 1% feature importance.
2. Surprisingly, the **month and children feature** have the highest feature importance. This is unexpected as the month and children feature are not directly related to the food categories purchased.

For a clearer understanding, we will re plot the same graph without the area name feature, and split the remaining features 3 categorical colors for better readability.

```
# Define the lists of features
demographic <- c("children", "average_age", "population", "male", "female", "adult", "senior")
others <- c("month", "area_sq_km", "people_per_sq_km")
property <- c("bungalow", "masionette", "terrace", "semi_detached", "detached", "annexe", "other", "unkn
              "total", "bungalow_perc", "masionette_perc", "terrace_perc", "semi_detached_perc",
              "detached_perc", "annexe_perc", "other_perc", "unknown_perc")

# Filter out columns with the prefix 'area_name'
demographic_features <- setdiff(demographic, grep("^area_name", demographic, value = TRUE))
others_features <- setdiff(others, grep("^area_name", others, value = TRUE))
property_features <- setdiff(property, grep("^area_name", property, value = TRUE))
```

```
#Convert the feature importance data to a data frame
feature_importance_data <- data.frame(Feature = names(feature_importance_data), Importance = feature_imp
```

```
# Define colors for each category
demographic_color <- "#F3DF6C"  # Mustard Yellow
others_color <- "#C93C20"      # Vermilion
property_color <- "#3E7BB6"

# Add a new column indicating the category of each feature
feature_importance_data <- feature_importance_data %>%
  mutate(Category = case_when(
    Feature %in% demographic_features ~ "Demographic",
    Feature %in% others_features ~ "Others",
    Feature %in% property_features ~ "Property",
    TRUE ~ "Other"
  ))

# Plot the feature importance
ggplot(data = feature_importance_data, aes(x = reorder(Feature, Importance), y = Importance, fill = Cate
  geom_bar(stat = "identity") +
  coord_flip() +
  theme_minimal() +
  labs(title = "Feature Importance Evaluation for Model 1",
       subtitle = "Colored by Category",
       x = "",
       y = "Importance (%)") +
  scale_fill_manual(values = c(Demographic = demographic_color, Others = others_color, Property = proper
  theme_dark_background +
  theme(legend.position = "bottom")
```


Feature Importance Evaluation for Model 1
Colored by Category
month
children
average_age
masionette_perc
people_per_sq_km
total
terrace_perc
senior
masionette
adult
male
female
population
semi_detached_perc
area_sq_km
terrace
semi_detached
detached_perc
bungalow_perc
detached
unknown_perc
bungalow
unknown
area_nameHavering
area_nameCroydon
area_nameBromley
area_nameSutton
area_nameEnfield
area_nameBarnet
area_nameHarrow
area_nameNewham
area_nameRedbridge
area_nameWaltham Forest
area_nameRichmond upon Thames
area_nameLambeth
area_nameMerton
area_nameWandsworth
area_nameKensington and Chelsea
area_nameHounslow
area_nameKingston upon Thames
area_nameGreenwich
area_nameBexley
area_nameHaringey
area_nameLewisham
area_nameBrent
area_nameHillingdon
area_nameCamden
annexe_perc
area_nameEaling
area_nameWestminster
area_nameSouthwark
area_nameHackney
area_nameHammersmith and Fulham
area_nameBarking and Dagenham
area_nameIslington
area_nameTower Hamlets
other_perc
annexe
other
area_nameCity of London
0
2
4
6
Importance (%)
Category
Demographic
Others
Property

Observation

1. For 'others' category, their feature importance seems to be scattered around, with no clear pattern.
2. For 'demographic' category, their feature importance as a category is higher than the 'property' category. **This is unexpected as the property category was expected to have a higher feature importance.**
3. However it is noteworthy to mention that the **difference between these categories is not much**, about 1-3% difference.

Conclusion

1. The **RMSE values are low, indicating that the model is performing well** in predicting the food categories purchased based on the property distribution.
2. The **average feature importance** shows that the **month and children feature** have the highest feature importance. This is unexpected as the month and children feature are not directly related to the food categories purchased.
3. The **area name has almost no impact** on the food categories purchased. With all of them scoring less than 1% feature importance.

Followup Action

1. We will modify the training dataset to exclude the area name feature and retrain the model to see if the feature importance changes.

ML 2 - Revisit Data Preparation

```
# Call the function to prepare the training and testing sets
results_no_area_name <- prepare_training_testing_sets(lsoa_grocery_property, num_targets = 17, train_pr
```

```
## Column names of encoded input features:
##  [1] "area_nameBarking and Dagenham"   "area_nameBarnet"
##  [3] "area_nameBexley"                 "area_nameBrent"
##  [5] "area_nameBromley"                "area_nameCamden"
##  [7] "area_nameCity of London"         "area_nameCroydon"
##  [9] "area_nameEaling"                 "area_nameEnfield"
## [11] "area_nameGreenwich"              "area_nameHackney"
## [13] "area_nameHammersmith and Fulham" "area_nameHaringey"
## [15] "area_nameHarrow"                 "area_nameHavering"
## [17] "area_nameHillingdon"             "area_nameHounslow"
## [19] "area_nameIslington"              "area_nameKensington and Chelsea"
## [21] "area_nameKingston upon Thames"   "area_nameLambeth"
## [23] "area_nameLewisham"               "area_nameMerton"
## [25] "area_nameNewham"                 "area_nameRedbridge"
## [27] "area_nameRichmond upon Thames"   "area_nameSouthwark"
## [29] "area_nameSutton"                 "area_nameTower Hamlets"
## [31] "area_nameWaltham Forest"         "area_nameWandsworth"
## [33] "area_nameWestminster"            "month"
## [35] "population"                      "male"
```

```
## [37] "female"                          "children"
## [39] "adult"                           "senior"
## [41] "average_age"                     "area_sq_km"
## [43] "people_per_sq_km"                "bungalow"
## [45] "masionette"                      "terrace"
## [47] "semi_detached"                   "detached"
## [49] "annexe"                          "other"
## [51] "unknown"                         "total"
## [53] "bungalow_perc"                   "masionette_perc"
## [55] "terrace_perc"                    "semi_detached_perc"
## [57] "detached_perc"                   "annexe_perc"
## [59] "other_perc"                      "unknown_perc"
##
## Dimensions of the training set:
## [1] 41036    60
##
## Dimensions of the testing set:
## [1] 10259    60
```

```
# Accessing the training and testing sets
train_data_no_area_name <- results_no_area_name$train_data
train_targets_no_area_name <- results_no_area_name$train_targets
test_data_no_area_name <- results_no_area_name$test_data
test_targets_no_area_name <- results_no_area_name$test_targets
```

```
# Drop the 'area_name' columns from the training and testing sets
train_data_no_area_name <- train_data_no_area_name[, -grep("^area_name", colnames(train_data_no_area_nar
test_data_no_area_name <- test_data_no_area_name[, -grep("^area_name", colnames(test_data_no_area_name)
```

ML 2 - Model Training

This section involves training a **multi-label regression model using multiple random forest models and aggregating them** to predict the food categories purchased based on the property distribution without the 'area_name' feature.

This code chunk is commented out as it takes a long time to run. The trained models were saved locally on the author's pc. The models are too large to be cached.

```
# # Initialize a list to store models for datasets without 'area_name'
# models_no_area_name <- list()
#
# # Initialize a vector to store training times for datasets without 'area_name'
# training_times_no_area_name <- numeric(length = length(colnames(train_targets_no_area_name)))
#
# # Capture start time for total training time for datasets without 'area_name'
# total_start_time_no_area_name <- Sys.time()
#
# # Loop through each target column in datasets without 'area_name'
# for (i in seq_along(colnames(train_targets_no_area_name))) {
#   target_name <- colnames(train_targets_no_area_name)[i]
#
#   # Extract the current target column
```

```
#   current_target <- train_targets_no_area_name[[target_name]]
#
#   # Combine the current target with the train data without 'area_name'
#   data_for_model <- cbind(current_target = current_target, train_data_no_area_name)
#
#   # Capture start time for individual model training
#   start_time <- Sys.time()
#
#   # Train a model for the current target
#   models_no_area_name[[target_name]] <- ranger(
#     dependent.variable.name = "current_target",
#     data = data_for_model,
#     num.trees = 500,
#     importance = 'impurity',
#     min.node.size = 5,
#     write.forest = TRUE
#   )
#
#   # Save the model
#   saveRDS(models_no_area_name[[target_name]], paste0("model_no_area_name_", target_name, ".rds"))
#
#   # Capture end time for individual model training
#   end_time <- Sys.time()
#   training_time <- end_time - start_time
#
#   # Store the training time in the vector
#   training_times_no_area_name[i] <- training_time
#
#   # Print the training time for the current model
#   cat("Time taken to train model for", target_name, ":", training_time, "seconds\n")
# }
#
# # Directly sum the individual training times for total for datasets without 'area_name'
# summed_total_training_time_no_area_name <- sum(training_times_no_area_name)
#
# # Print the corrected total training time for datasets without 'area_name'
# cat("Total training time for datasets without 'area_name':", summed_total_training_time_no_area_name,
```

ML 2 - Model Evaluation

This section involves predicting the target variables on the test data and evaluating the model's performance using metrics such as RMSE for datasets without 'area_name'.

```
# Initialize lists to store predictions and evaluation metrics for datasets without 'area_name'
predictions_no_area_name <- list()
evaluation_metrics_no_area_name <- data.frame(Target=character(),
                                              RMSE=numeric(),
                                              stringsAsFactors=FALSE)
```

```
# Loop through each target column to predict and evaluate for datasets without 'area_name'
for (target_name in colnames(train_targets_no_area_name)) {
```

```r
  # Load the trained model for datasets without 'area_name'
  model_no_area_name <- readRDS(paste0("model_no_area_name_", target_name, ".rds"))

  # Predict on the test data for datasets without 'area_name'
  prediction_no_area_name <- predict(model_no_area_name, data = test_data_no_area_name)$predictions
  predictions_no_area_name[[target_name]] <- prediction_no_area_name

  # Actual values for datasets without 'area_name'
  actual_no_area_name <- test_targets_no_area_name[[target_name]]

  # Calculate RMSE for datasets without 'area_name'
  rmse_val_no_area_name <- rmse(actual_no_area_name, prediction_no_area_name)

  # Store evaluation metrics for datasets without 'area_name'
  evaluation_metrics_no_area_name <- rbind(evaluation_metrics_no_area_name,
                                           data.frame(Target=target_name,
                                                      RMSE=rmse_val_no_area_name))
}

# Print evaluation metrics summary for datasets without 'area_name'
print(evaluation_metrics_no_area_name)
```

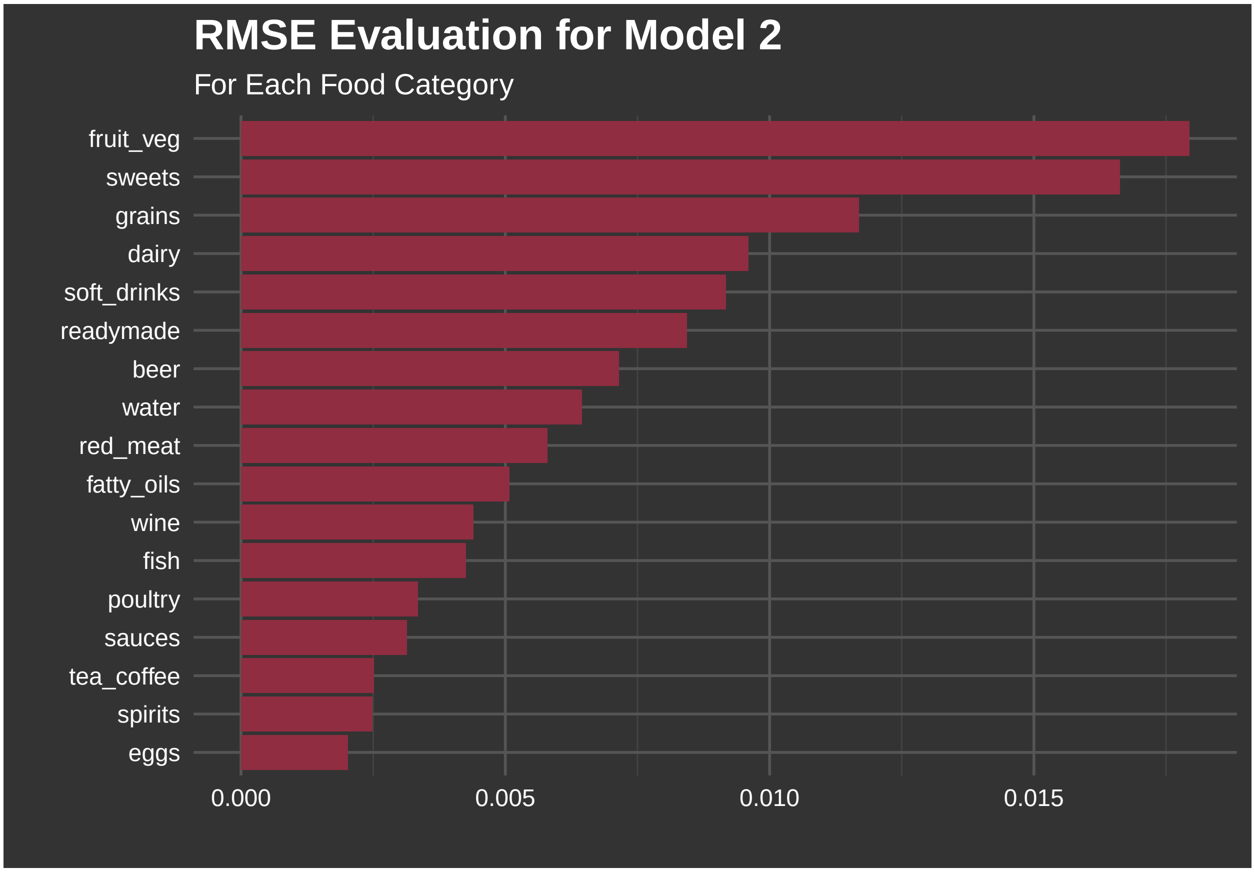
```
##         Target        RMSE
## 1         beer 0.007148098
## 2        dairy 0.009596135
## 3         eggs 0.002019935
## 4   fatty_oils 0.005079519
## 5         fish 0.004256015
## 6    fruit_veg 0.017943483
## 7       grains 0.011686767
## 8     red_meat 0.005798609
## 9      poultry 0.003348154
## 10   readymade 0.008433137
## 11      sauces 0.003135908
## 12 soft_drinks 0.009174215
## 13     spirits 0.002481543
## 14      sweets 0.016625335
## 15  tea_coffee 0.002513191
## 16       water 0.006451212
## 17        wine 0.004389627
```

```r
# Calculate the total RMSE for datasets without 'area_name'
total_rmse_no_area_name <- sqrt(mean(evaluation_metrics_no_area_name$RMSE^2))

# Print the total RMSE for datasets without 'area_name'
cat("Total RMSE for datasets without 'area_name':", total_rmse_no_area_name, "\n")
```

```
## Total RMSE for datasets without 'area_name': 0.008434664
```

```r
# RMSE Plot for datasets without 'area_name'
ggplot(evaluation_metrics_no_area_name, aes(x = reorder(Target, RMSE), y = RMSE)) +
  geom_bar(stat = "identity", fill = "#902D41") +
```

```r
  coord_flip() +
  theme_minimal() +
  labs(title = "RMSE Evaluation for Model 2",
       subtitle = "For Each Food Category",
       x = "",
       y = "") +
  theme_dark_background
```



Observation

1. The RMSE values for datasets without 'area_name' are similar to the RMSE values for the complete dataset.
2. The **RMSE values are low, indicating that the model is performing well** in predicting the food categories purchased based on the property distribution for datasets without 'area_name'.

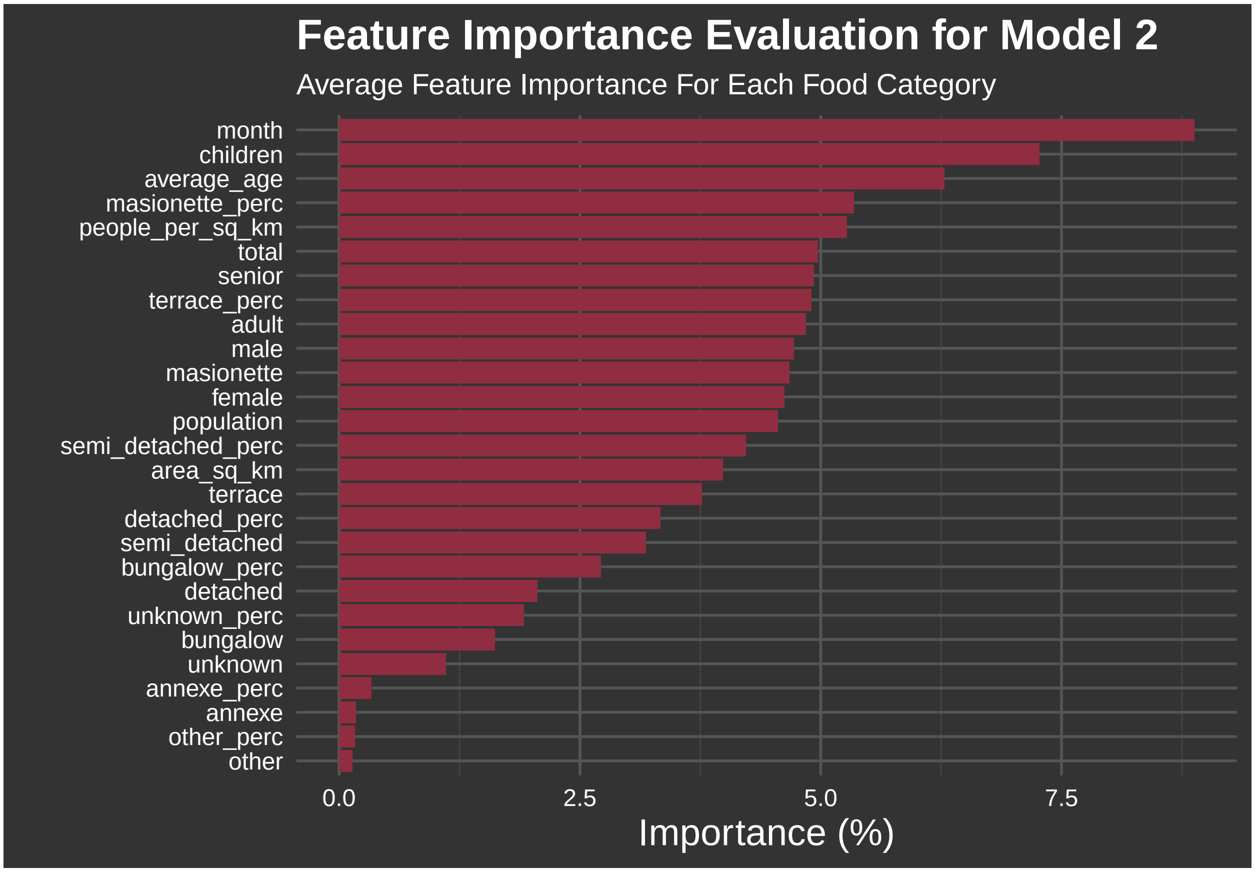
```r
# Calculate the feature importance for datasets without 'area_name'
feature_importance_no_area_name <- list()

# Loop through each target column to load the model and calculate feature importance for datasets witho
for (target_name in colnames(train_targets_no_area_name)) {
  # Load the trained model for datasets without 'area_name'
  model_no_area_name <- readRDS(paste0("model_no_area_name_", target_name, ".rds"))
```

```r
  # Extract feature importance for datasets without 'area_name'
  importance_no_area_name <- model_no_area_name$variable.importance

  # Normalize the feature importance to sum up to 1 (or 100%) for datasets without 'area_name'
  importance_normalized_no_area_name <- importance_no_area_name / sum(importance_no_area_name)

  # Store the normalized importance in the list for datasets without 'area_name'
  feature_importance_no_area_name[[target_name]] <- importance_normalized_no_area_name
}

# Calculate the average feature importance across all models for datasets without 'area_name'
# This step assumes that all models share the same features in the same order for datasets without 'are
# Convert the list to a matrix for easier column-wise operations for datasets without 'area_name'
importance_matrix_no_area_name <- do.call("cbind", feature_importance_no_area_name)

# Calculate the mean importance for each feature across all models for datasets without 'area_name'
average_importance_no_area_name <- rowMeans(importance_matrix_no_area_name)

# Optionally, convert to percentage for datasets without 'area_name'
feature_importance_data_no_area_name <- average_importance_no_area_name * 100
```

```r
# Convert the feature importance data to a data frame for datasets without 'area_name'
feature_importance_data_no_area_name <- data.frame(Feature = names(feature_importance_data_no_area_name)
```

```r
# Order by importance for datasets without 'area_name'
feature_importance_data_no_area_name <- feature_importance_data_no_area_name %>%
  arrange(desc(Importance))
```

```r
# Plot the feature importance for datasets without 'area_name'
ggplot(feature_importance_data_no_area_name, aes(x = reorder(Feature, Importance), y = Importance)) +
  geom_bar(stat = "identity", fill = "#902D41" ) +
  coord_flip() +
  theme_minimal() +
  labs(title = "Feature Importance Evaluation for Model 2",
       subtitle = "Average Feature Importance For Each Food Category",
       x = "",
       y = "Importance (%)") +
  theme_dark_background
```



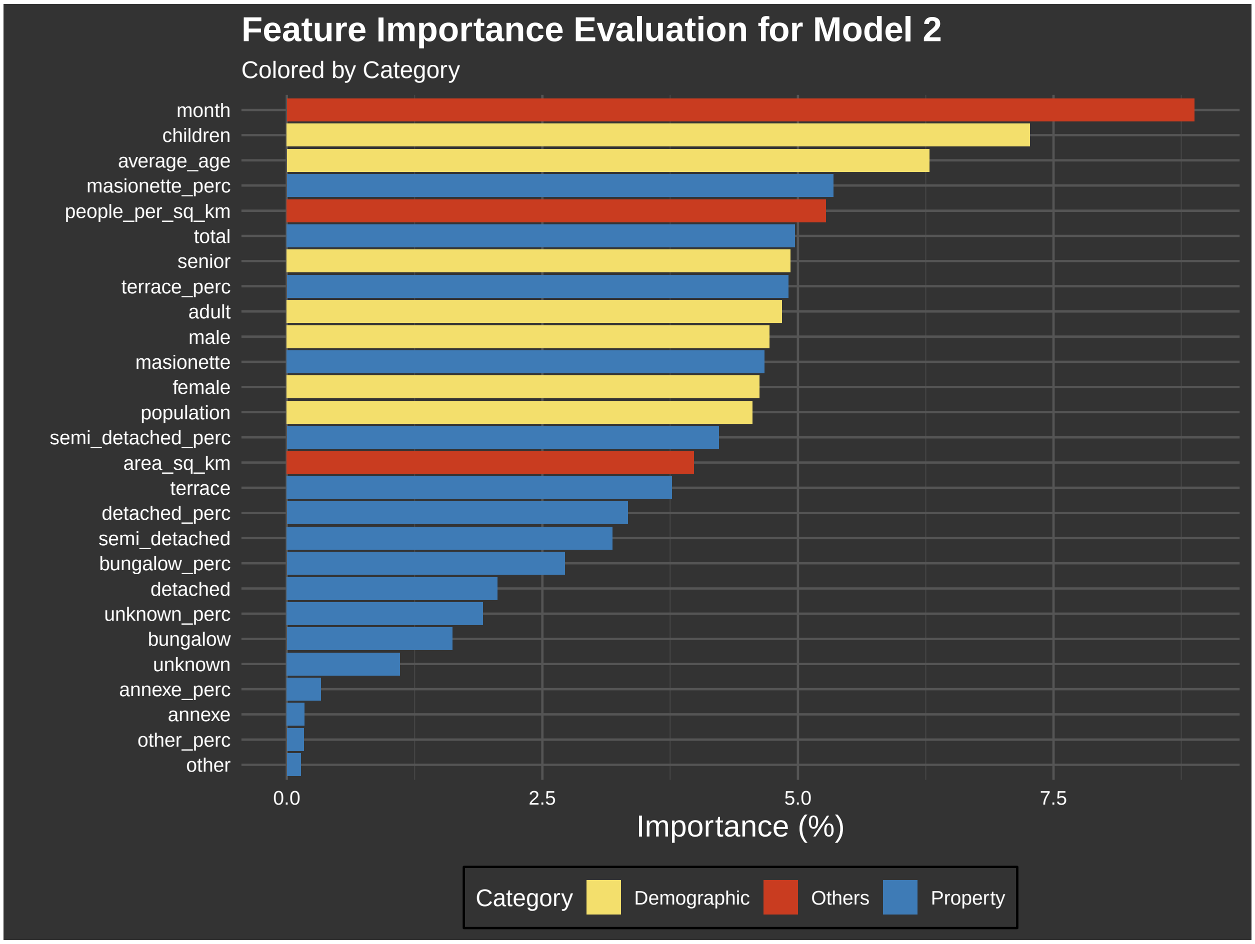
Observation

1. The **average feature importance for datasets without 'area_name'** shows that the **month and children feature** have the highest feature importance. This is again unexpected as the month and children feature are not directly related to the food categories purchased.

```
#Category colors
demographic_color <- "#F3DF6C"  # Mustard Yellow
others_color <- "#C93C20"       # Vermilion
property_color <- "#3E7BB6"     # Deep Sky Blue
```

```
# add new column indicating the category of each feature
feature_importance_data_no_area_name <- feature_importance_data_no_area_name %>%
  mutate(Category = case_when(
    Feature %in% demographic_features ~ "Demographic",
    Feature %in% others_features ~ "Others",
    Feature %in% property_features ~ "Property",
    TRUE ~ "Other"
  ))

# Plot the feature importance for datasets without 'area_name'
ggplot(data = feature_importance_data_no_area_name, aes(x = reorder(Feature, Importance), y = Importanc
  geom_bar(stat = "identity") +
  coord_flip() +
```

```r
  theme_minimal() +
  labs(title = "Feature Importance Evaluation for Model 2",
       subtitle = "Colored by Category",
       x = "",
       y = "Importance (%)") +
  scale_fill_manual(values = c(Demographic = demographic_color, Others = others_color, Property = prope
  theme_dark_background +
  theme(legend.position = "bottom")
```

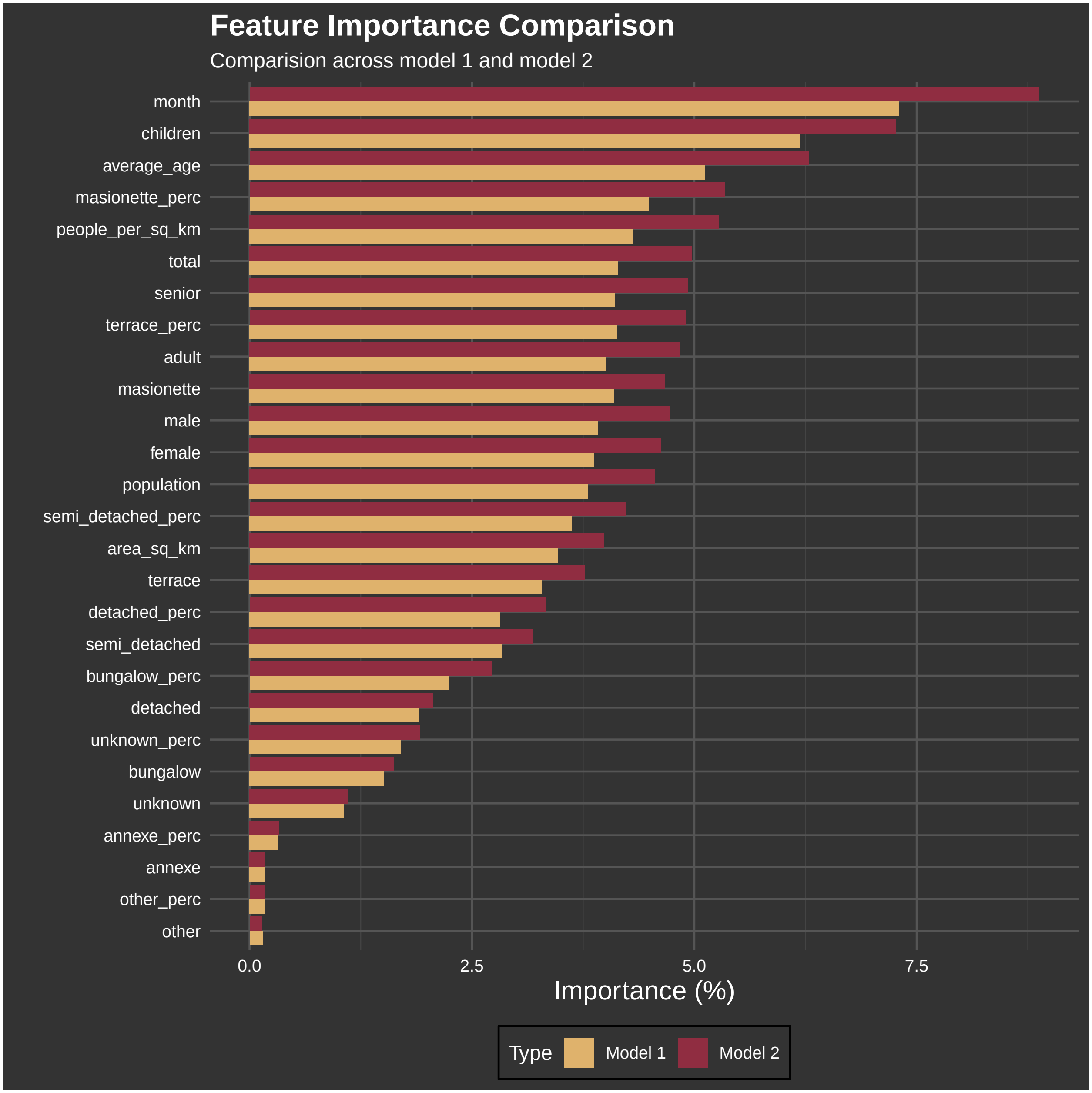


Observation

In terms of feature importance of categories, as a while, demographics performs the best, followed by others than property. **This disputes the origional hypothesis, and suggests that there are other categories that play a larger influence** on the food categories purchased.

```r
# Drop the features with area_name prefix
feature_importance_data_dropped <- feature_importance_data %>%
  filter(!grepl("^area_name", Feature))
```

```r
 # plot feature_importance_data_dropped and feature_importance_data_no_area_name horizontally
# Combine the two data frames into one for plotting
combined_data <- bind_rows(
  mutate(feature_importance_data_dropped, Type = "With 'area_name'"),
  mutate(feature_importance_data_no_area_name, Type = "Without 'area_name'")
)

# Plot comparision
ggplot(combined_data, aes(x = reorder(Feature, Importance), y = Importance, fill = Type)) +
  geom_bar(stat = "identity", position = "dodge") +
  coord_flip() +
  theme_minimal() +
  labs(title = "Feature Importance Comparison",
       subtitle = "Comparision across model 1 and model 2",
       x = "",
       y = "Importance (%)") +
  scale_fill_manual(values = c("With 'area_name'" = "#DFB26C", "Without 'area_name'" = "#902D41"),
                    labels = c("With 'area_name'" = "Model 1", "Without 'area_name'" = "Model 2")) +
  theme_dark_background +
  theme(legend.position = "bottom")
```

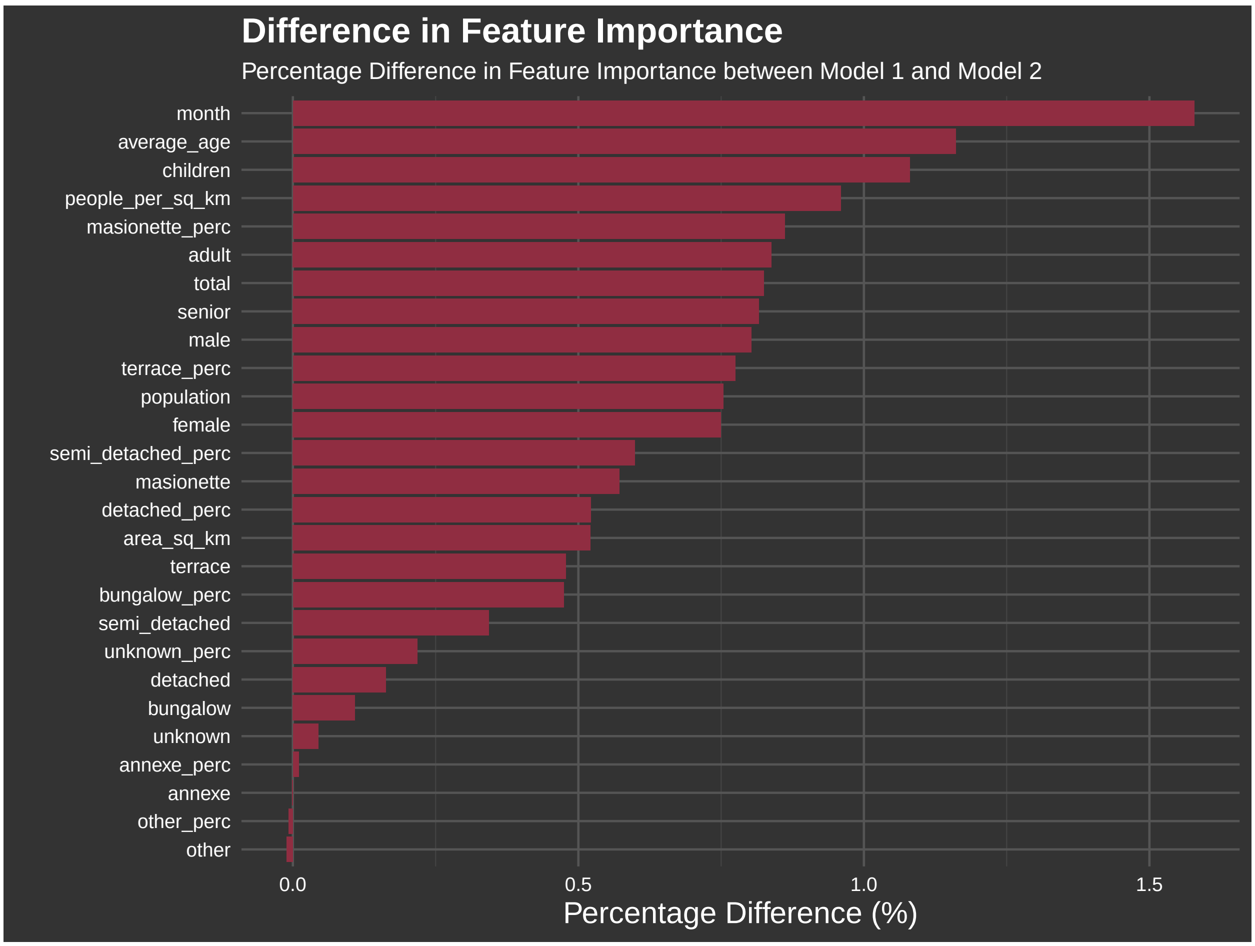


Observation

With the reduction of the 'area_name' feature, the feature importance of the remaining features has increased. **This suggests that the 'area_name' feature was diluting the feature importance of the other features.**

```
# Plot the difference in feature importance between feature_importance_data_dropped and feature_importa
feature_importance_difference <- feature_importance_data_no_area_name %>%
  left_join(feature_importance_data_dropped, by = "Feature") %>%
  mutate(Difference = Importance.x - Importance.y) %>%
  arrange(desc(Difference))
```

```
# Plot the difference in feature importance
ggplot(feature_importance_difference, aes(x = reorder(Feature, Difference), y = Difference)) +
  geom_bar(stat = "identity", fill = "#902D41" ) +
  coord_flip() +
  theme_minimal() +
  labs(title = "Difference in Feature Importance",
       subtitle = "Percentage Difference in Feature Importance between Model 1 and Model 2",
       x = "",
       y = "Percentage Difference (%)") +
  theme_dark_background
```



Observation

1. Without the inclusion of the area name features, we would **assume that the difference in feature importance would increase proportionally**. However, the difference in feature importance not proportional, with **month increasing almost 1.5 times compared to other features.**
2. This would suggest that **month has a stronger impact on the food categories purchased compared to other features.** This is unexpected as the month feature is not directly related to the food categories purchased.

Implementation - New Tesco Store Scenario

This section involves predicting the food categories purchased for a new Tesco store in an area that does not have a Tesco store. The area selected is **Croydon**.

```r
# Extract a row where Areas not in lsoa_grocery_data are marked as FALSE
no_tesco_area <- merged_df[!merged_df$has_tesco, ]

nrow(no_tesco_area)
```

```
## [1] 36
```

```r
# Filter out random row from no_tesco_area
no_tesco_area_single <- no_tesco_area[sample(nrow(no_tesco_area), 1), ]


no_tesco_area_single
```

```
## Simple feature collection with 1 feature and 42 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 538116.4 ymin: 162156.8 xmax: 539227.7 ymax: 163501
## Projected CRS: OSGB36 / British National Grid
##         area_id     LSOA11NM  MSOA11CD    MSOA11NM   LAD11CD LAD11NM   RGN11CD
## 10682 E01001082 Croydon 032E E02000225 Croydon 032 E09000008 Croydon E12000007
##       RGN11NM USUALRES HHOLDRES COMESTRES POPDEN HHOLDS AVHHOLDSZ f_beer
## 10682  London     1627     1627         0   29.4    560       2.9     NA
##       f_dairy f_eggs f_fats_oils f_fish f_fruit_veg f_grains f_meat_red
## 10682      NA     NA          NA     NA          NA       NA         NA
##       f_poultry f_readymade f_sauces f_soft_drinks f_spirits f_sweets
## 10682        NA          NA       NA            NA        NA       NA
##       f_tea_coffee f_water f_wine population male female age_0_17 age_18_64
## 10682           NA      NA     NA         NA   NA     NA       NA        NA
##       age_65+ avg_age area_sq_km people_per_sq_km month has_tesco
## 10682      NA      NA         NA               NA    NA     FALSE
##                             geometry
## 10682 MULTIPOLYGON (((538990.3 16...
```

Observation

We select a random area which does not have a tesco store to predict on to demonstrate the model usage. The area selected is **'E01001082'**, which is **Croydon**.

```r
# Print all feature names
colnames(train_data_no_area_name)
```

```
##  [1] "month"            "population"       "male"
##  [4] "female"           "children"         "adult"
##  [7] "senior"           "average_age"      "area_sq_km"
## [10] "people_per_sq_km" "bungalow"         "masionette"
```

```
## [13] "terrace"           "semi_detached"      "detached"
## [16] "annexe"            "other"              "unknown"
## [19] "total"             "bungalow_perc"      "masionette_perc"
## [22] "terrace_perc"      "semi_detached_perc" "detached_perc"
## [25] "annexe_perc"       "other_perc"         "unknown_perc"
```

```
# Generate dataframe for new tesco store

# Column names
column_names <- c("month", "population", "male", "female", "children", "adult",
                  "senior", "average_age", "area_sq_km", "people_per_sq_km",
                  "bungalow", "masionette", "terrace", "semi_detached",
                  "detached", "annexe", "other", "unknown", "total",
                  "bungalow_perc", "masionette_perc", "terrace_perc",
                  "semi_detached_perc", "detached_perc", "annexe_perc",
                  "other_perc", "unknown_perc")

# Specify values for each column
values <- list(
  month = 1,
  population = 10000,
  male = 5000,
  female = 5000,
  children = 2000,
  adult = 6000,
  senior = 2000,
  average_age = 40,
  area_sq_km = 10,
  people_per_sq_km = 1000,
  bungalow = 100,
  masionette = 200,
  terrace = 300,
  semi_detached = 150,
  detached = 250,
  annexe = 50,
  other = 50,
  unknown = 100,
  total = 1000,
  bungalow_perc = 10,
  masionette_perc = 20,
  terrace_perc = 30,
  semi_detached_perc = 15,
  detached_perc = 25,
  annexe_perc = 5,
  other_perc = 5,
  unknown_perc = 10
)

# Convert values to a one-row dataframe
one_row_df <- as.data.frame(matrix(unlist(values), nrow = 1, byrow = TRUE))

# Set column names
colnames(one_row_df) <- column_names
```

```r
# Print the dataframe
print(one_row_df)
```

```
##   month population male female children adult senior average_age area_sq_km
## 1     1      10000 5000   5000     2000  6000   2000          40         10
##   people_per_sq_km bungalow masionette terrace semi_detached detached annexe
## 1             1000      100        200     300           150      250     50
##   other unknown total bungalow_perc masionette_perc terrace_perc
## 1    50     100  1000            10              20           30
##   semi_detached_perc detached_perc annexe_perc other_perc unknown_perc
## 1                 15            25           5          5           10
```

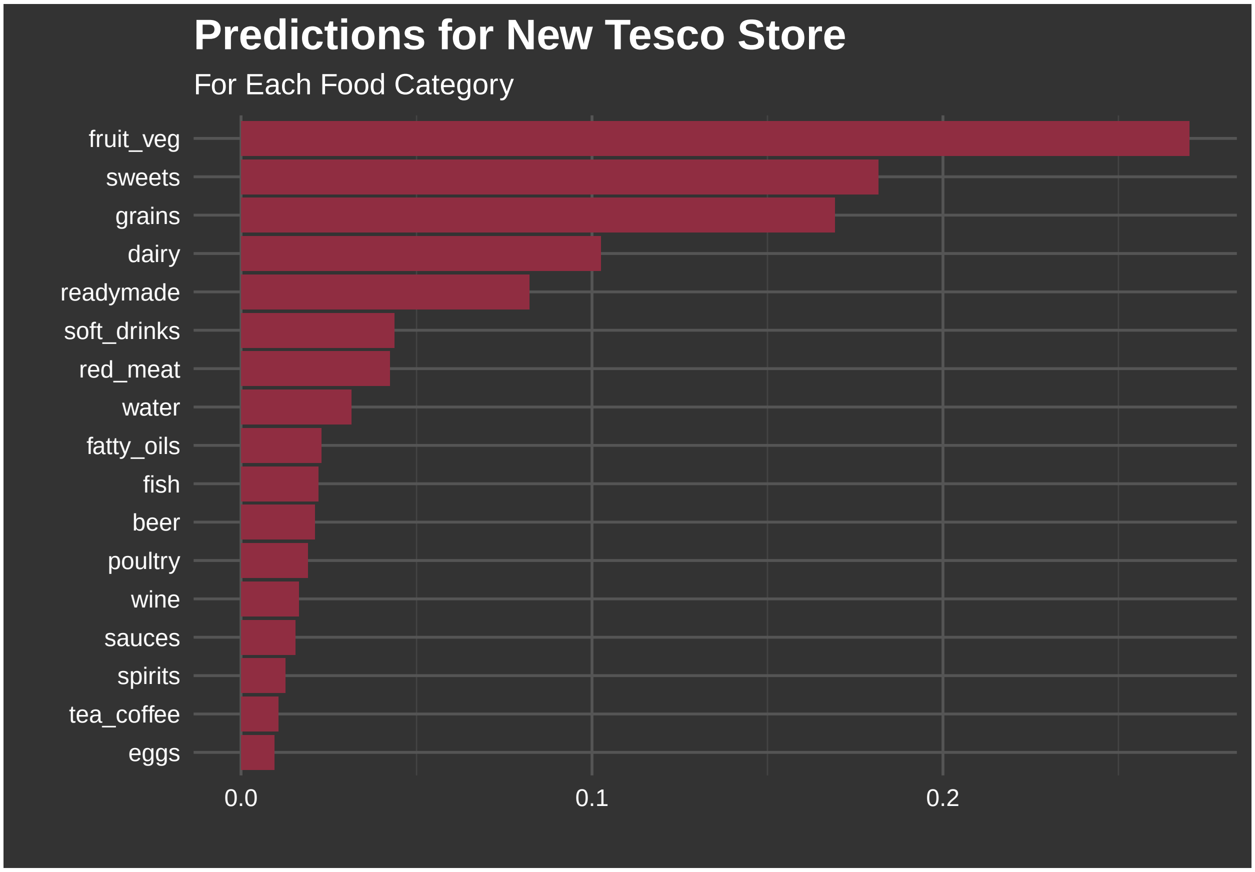
```r
# Initialize lists to store predictions and evaluation metrics for new tesco store
predictions_new_tesco <- list()
```

```r
# Loop through each target column to predict and evaluate for dataset of new tesco store
for (target_name in colnames(train_targets_no_area_name)) {
  # Load the trained model for datasets without 'area_name'
  model_no_area_name <- readRDS(paste0("model_no_area_name_", target_name, ".rds"))

  # Predict on the new tesco store data
  prediction_new_tesco <- predict(model_no_area_name, data = one_row_df)$predictions
  predictions_new_tesco[[target_name]] <- prediction_new_tesco
}

# Print the predictions for the new tesco store
print(predictions_new_tesco)
```

```
## $beer
## [1] 0.02103759
##
## $dairy
## [1] 0.1025556
##
## $eggs
## [1] 0.009421906
##
## $fatty_oils
## [1] 0.02286149
##
## $fish
## [1] 0.02201847
##
## $fruit_veg
## [1] 0.2702389
##
## $grains
## [1] 0.1691914
##
## $red_meat
## [1] 0.04233851
##
```

```
## $poultry
## [1] 0.01898717
##
## $readymade
## [1] 0.0821018
##
## $sauces
## [1] 0.01541804
##
## $soft_drinks
## [1] 0.04369309
##
## $spirits
## [1] 0.01266248
##
## $sweets
## [1] 0.1815148
##
## $tea_coffee
## [1] 0.01061526
##
## $water
## [1] 0.03136168
##
## $wine
## [1] 0.01639934
```

```
# Plot the predictions for the new tesco store
predictions_new_tesco_df <- data.frame(Target = names(predictions_new_tesco), Prediction = unlist(predic

ggplot(predictions_new_tesco_df, aes(x = reorder(Target, Prediction), y = Prediction)) +
  geom_bar(stat = "identity", fill = "#902D41") +
  coord_flip() +
  theme_minimal() +
  labs(title = "Predictions for New Tesco Store",
       subtitle = "For Each Food Category",
       x = "",
       y = "") +
  theme_dark_background
```



Observation

Thus for a **hypothetical new TESCO store in Croyden, for the month of January, this would be the predicted distribution of food categories that the store's inventory should have.**

Predicted distribution for each category across the year

This section involves predicting the percentage distribution for each food category across the year for the new Tesco store in Croydon.

```
# Initialize a list to store predictions for each month and category
monthly_predictions <- list()

# Loop through months 1-12
for (month in 1:12) {
  # Update the month value in the dataframe
  one_row_df$month <- month

  # Temporary storage for monthly predictions
  temp_predictions <- list()

  # Loop through each target category
  for (target_name in colnames(train_targets_no_area_name)) {
```

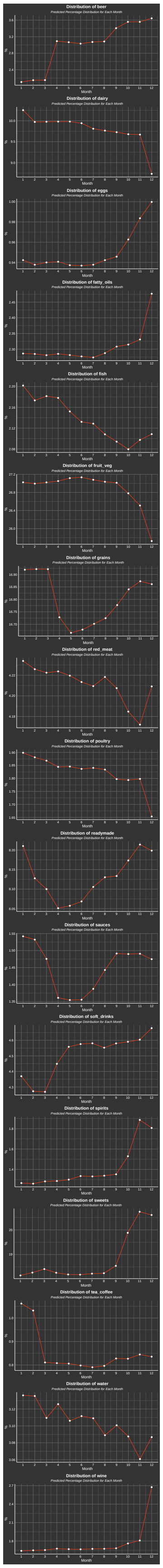
```r
    # Load the trained model
    model_no_area_name <- readRDS(paste0("model_no_area_name_", target_name, ".rds"))

    # Predict on the updated dataframe
    prediction_new_tesco <- predict(model_no_area_name, data = one_row_df)$predictions
    temp_predictions[[target_name]] <- prediction_new_tesco
  }

  # Store the monthly predictions
  monthly_predictions[[month]] <- temp_predictions
}
```

```r
# Initialize an empty dataframe
predictions_df <- data.frame(Month = integer(),
                             Category = character(),
                             Prediction = numeric(),
                             stringsAsFactors = FALSE)

# Populate the dataframe, converting Prediction into a percentage
for (month in 1:length(monthly_predictions)) {
    for (category in names(monthly_predictions[[month]])) {
        predictions_df <- rbind(predictions_df,
                                data.frame(Month = month,
                                           Category = category,
                                           Prediction = unlist(monthly_predictions[[month]][[category]]
                                           stringsAsFactors = FALSE))
    }
}
```

```r
# Unique categories
unique_categories <- unique(predictions_df$Category)

# Generate a list to hold plots
plots_list <- list()
```

```r
# Colors for lines and points that stand out on a dark background
line_color <- "#C93C20"
point_color <- "#FFFFFF"

# Adjust the plotting code within the loop
for (category in unique_categories) {
    p <- ggplot(subset(predictions_df, Category == category), aes(x = Month, y = Prediction)) +
        geom_line(color = line_color, linewidth = 1) + # Line settings
        geom_point(color = point_color, size = 2, shape = 21, fill = "white") + # Point settings
        theme_minimal() +
        theme(plot.background = element_rect(fill = "#333333", color = NA), # Dark background
              panel.background = element_rect(fill = "#333333", color = NA),
              text = element_text(color = "white"), # Text color
              plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
              plot.subtitle = element_text(hjust = 0.5, size = 12, face = "italic"),
              axis.title = element_text(size = 14),
              axis.text = element_text(size = 12, color = "white"),
              axis.line = element_line(color = "white"), # White axis lines
```

```
              panel.grid.major = element_line(color = "#656565"), # Lighter grid lines for contrast
              panel.grid.minor = element_line(color = "#505050")) + # Lighter grid lines for contrast
        labs(title = paste("Distribution of", category),
             subtitle = "Predicted Percentage Distribution for Each Month",
             x = "Month",
             y = "%") +
        scale_x_continuous(breaks = 1:12) # Ensure x-axis shows only whole month numbers

    # Store the plot in the list
    plots_list[[category]] <- p
}

# Arrange and display the plots vertically
do.call(grid.arrange, c(plots_list, ncol = 1))
```


Distribution of beer
Predicted Percentage Distribution for Each Month
Distribution of dairy
Predicted Percentage Distribution for Each Month
Distribution of eggs
Predicted Percentage Distribution for Each Month
Distribution of fatty_oils
Predicted Percentage Distribution for Each Month
Distribution of fish
Predicted Percentage Distribution for Each Month
Distribution of fruit_veg
Predicted Percentage Distribution for Each Month
Distribution of grains
Predicted Percentage Distribution for Each Month
Distribution of red_meat
Predicted Percentage Distribution for Each Month
Distribution of poultry
Predicted Percentage Distribution for Each Month
Distribution of readymade
Predicted Percentage Distribution for Each Month
Distribution of sauces
Predicted Percentage Distribution for Each Month
Distribution of soft_drinks
Predicted Percentage Distribution for Each Month
Distribution of spirits
Predicted Percentage Distribution for Each Month
Distribution of sweets
Predicted Percentage Distribution for Each Month
Distribution of tea_coffee
Predicted Percentage Distribution for Each Month
Distribution of water
Predicted Percentage Distribution for Each Month
Distribution of wine
Predicted Percentage Distribution for Each Month
Month
%

Observation

The above plots show the **predicted percentage distribution for each food category across the year**. This would help the new TESCO store in Croydon to plan their inventory based on the expected demand for each food category.

Interestingly, the relationships for several categories can be seen in their individual plots.

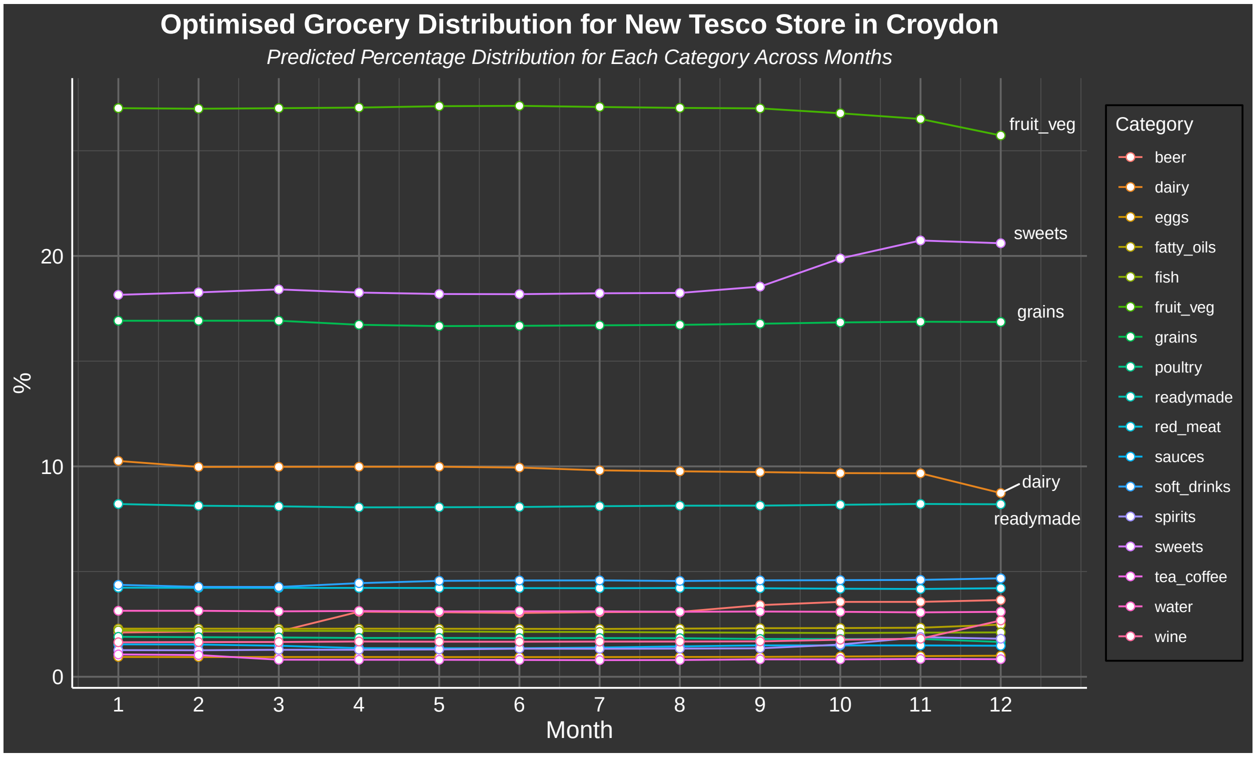
Distribution of:

1. Beer, eggs, fatty oil, and soft drinks **increases and peaks at the end of the year**.
2. Diary, fruits, poultry **decreases and has a sharp drop in december (winter).**
3. Fish, grain, readymade, sauces, water decreases, **hits bottoms and peaks again** at the end of the year.
4. Red meat **decreases but increases sharply during winter month** of december.
5. Tea and coffee distribution **follows the winter months** (high demand in november to february).

Combined Line Graph

```
# Filter predictions_df for the last month and specific categories
label_data <- predictions_df %>%
  group_by(Category) %>%
  filter(Month == max(Month), Category %in% c("fruit_veg", "grains", "sweets", "dairy", "readymade"))

# Update your plot code to include geom_text_repel for labels
p <- ggplot(predictions_df, aes(x = Month, y = Prediction, color = Category)) +
  geom_line() +  # Draw lines
  geom_point(size = 2, shape = 21, fill = "white") +  # Draw points
  geom_text_repel(data = label_data,
                  aes(label = Category),
                  nudge_x = 0.5, nudge_y = 0.5, # Adjust these values as needed
                  size = 3.5, color = "white") + # Ensure text color is visible on dark background
  theme_minimal() +
  theme(plot.background = element_rect(fill = "#333333", color = NA),  # Dark background
        panel.background = element_rect(fill = "#333333", color = NA),
        text = element_text(color = "white"),  # Text color
        plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5, size = 12, face = "italic"),
        axis.title = element_text(size = 14),
        axis.text = element_text(size = 12, color = "white"),
        axis.line = element_line(color = "white"),  # White axis lines
        panel.grid.major = element_line(color = "#656565"),  # Lighter grid lines for contrast
        panel.grid.minor = element_line(color = "#505050"),  # Lighter grid lines for contrast
        legend.background = element_rect(fill = "#333333"),
        legend.text = element_text(color = "white")) +
  labs(title = "Optimised Grocery Distribution for New Tesco Store in Croydon",
       subtitle = "Predicted Percentage Distribution for Each Category Across Months",
       x = "Month",
       y = "%",
       color = "Category") +
  scale_x_continuous(breaks = 1:12)  # Ensure x-axis shows only whole month numbers
```

```r
# Print the plot
print(p)
```



Observation

The top 5 categories for the predicted distribution are: 1. Fruit and Vegetables 2. Sweets 3. Grains 4. Dairy 5. Readymade

Conclusion

From the analysis, **the feature that had the most influence on predicting food categories was month**, and not so much about the property types.

Althought property type features do have an impact on the food categories purchased, the month feature has a higher impact. With this overall insight in mind, perhaps for future analysis, **we could explore the relationship between the month feature, in particular the weather statistics and the food categories purchased.**

Limitations

Bias - data is only for users who frequent TESCOs. This may not be representative of the general population.

Data Quality - The data is based of the consumer behaviours off one year, to make a more robust model, the same data over multiple years would be needed.

References

1. Aiello, L. M., Quercia, D., Schifanella, R., & Del Prete, L. (2020). Tesco Grocery 1.0, a large-scale dataset of grocery purchases in London. Scientific Data, 7(1). https://doi.org/10.1038/s41597-020-0397-7

2. Area type definitions census 2021. Area type definitions Census 2021 - Office for National Statistics. (n.d.). https://www.ons.gov.uk/census/census2021dictionary/areatypedefinitions#:~:text=Middle%20layer%20Super%

Data sets

1. LSOA & MSOA area codes: https://hub.arcgis.com/datasets/0f80c523f3cd4d0fab5111572f84a2fb_0/explore

2. Tesco Grocery Dataset https://figshare.com/collections/Tesco_Grocery_1_0/4769354/2

3. London Property Dataset https://data.london.gov.uk/dataset/property-build-period-lsoa

4. London Shape file https://data.london.gov.uk/dataset/statistical-gis-boundary-files-london