VIRGINIA COMMONWEALTH UNIVERSITY

STATISTICAL ANALYSIS & MODELING

Multivariate Analysis and Business Analytics Applications (Python and R)

Kirthan Shaker Iyangar
V01072700

Date of Submission: 30/06/2023

CONTENTS

Multivariate Analysis and Business Analytics Applications (Python and R)

INTRODUCTION

This study conducts a Multivariate Analysis and Business Analytics Applications on the assigned dataset, aiming to validate underlying assumptions, evaluate model performance, and interpret the results. Multivariate Analysis is a widely used statistical method for modelling binary outcomes based on predictor variables. In this analysis, we will preprocess the dataset, handle missing values, and ensure that the assumptions of logistic regression are met.

We will explore the relationships between the dependent variable and various independent variables, identifying significant predictors. The confusion will be utilised to assess the model's accuracy and predictive power. Furthermore, we will perform a decision tree analysis on the same dataset and compare its performance to the logistic regression model.

By systematically analyzing the data using these methods, we aim to provide comprehensive insights that can guide decision-making processes. The findings will highlight the strengths and weaknesses of each modeling approach and contribute to a better understanding of the data's underlying structure.

OBJECTIVES

1. Perform Principal Component Analysis and Factor Analysis to identify data dimensions (Survey.csv Download Survey.csv)

2. Conduct Cluster Analysis to characterize respondents based on background variables (Survey.csv Download Survey.csv)

3. Apply Multidimensional Scaling and interpret the results (icecream.csv Download icecream.csv)

4. Conjoint Analysis (pizza_data.csv)Download pizza_data.csv)

BUSINESS SIGNIFICANCE

Using these advanced statistical models like logistic regression, decision trees, probit, and Tobit regressions can significantly enhance business decision-making. For instance, logistic regression and decision trees can predict customer churn, enabling targeted retention strategies to maintain revenue streams. Probit regression can analyze consumer behavior, helping businesses tailor product offerings and marketing campaigns to specific demographics, such as dietary preferences. Tobit regression is crucial for assessing loan default risks, allowing financial institutions to implement risk-based pricing and targeted interventions. These models provide actionable insights, optimizing strategies across customer retention, product development, and risk management.

RESULTS AND INTERPRETATION

1. Perform Principal Component Analysis and Factor Analysis to identify data dimensions (Survey.csv Download Survey.csv)

#Identifying logistic regression analysis

Code and Result:

A) Do principal component analysis and factor analysis and identify the dimensions in the data.

```
is.na(survey_df)
sum(is.na(survey_df))
sur_int=survey_df[,20:46]
str(sur_int)
dim(sur_int)
library(GPArotation)
pca <- principal(sur_int,5,n.obs =162, rotate ="promax")
pca

om.h<-omega(sur_int,n.obs=162,sl=FALSE)
op<-par(mfrow=c(1,1))
om<-omega(sur_int,n.obs=162)
library(FactoMineR)
pca<-PCA(sur_int,scale.unit = TRUE)
summary(pca)
biplot(pca, scale = 0)
str(sur_int)
dim(sur_int)
show(sur_int)
```

Result:

1. Call and Setup:

- The PCA was performed on a correlation matrix with five factors, using Promax rotation and 162 observations.

2. Standardized Loadings (Pattern Matrix):

The table shows the loadings of each variable on the five retained factors (RC1 to RC5), communalities (h2), uniqueness (u2), and complexity (com).

Key Points:

- **Loadings:** These represent the correlation between each variable and the factor. Higher absolute values indicate a stronger relationship.
- **Communalities (h2):** The proportion of each variable's variance explained by the factors.
- **Uniqueness (u2):** The proportion of each variable's variance not explained by the factors (1 - h2).
- **Complexity (com):** Indicates how many factors are needed to explain a variable.

Example Interpretation of Loadings:

- **X3..Proximity.to.transport:** High loading on RC3 (0.77), meaning this variable is strongly related to RC3.
- **X4..Proximity.to.work.place:** High loadings on RC4 (0.82) and negative on RC5 (-0.46), indicating it is strongly related to RC4 and negatively to RC5.

3. SS Loadings and Proportion of Variance:

- **SS Loadings:** Sum of squared loadings for each factor.
- **Proportion Var:** Proportion of the total variance explained by each factor.
- **Cumulative Var:** Cumulative proportion of variance explained by the factors.

4. Component Correlations:

- Shows correlations between the components (RC1 to RC5). High correlations indicate that components share variance.

5. Model Fit:

- **RMSR (Root Mean Square of Residuals):** 0.07
- **Empirical Chi-square:** 252.24 with p < 0.11
- **Fit based on off-diagonal values:** 0.95
- These values indicate how well the model fits the data. A lower RMSR and higher fit value indicate a better fit.

6. Warnings and Factor Scores:

- **Warnings:**
  - The estimated weights for factor scores might be incorrect.
  - An ultra-Heywood case was detected (implies a possible issue with communalities being greater than 1).
- These warnings suggest potential issues with the solution and that careful examination is necessary.

7. PCA (FactoMineR):

- Eigenvalues and the proportion of variance explained by each dimension.
- **Dim.1:** Explains 31.82% of the variance.
- **Dim.1 to Dim.5:** Cumulatively explain 60.69% of the variance.
- **Dim.1 to Dim.10:** Cumulatively explain 80.77% of the variance.
- The first few dimensions explain a significant portion of the variance, which is common in PCA.

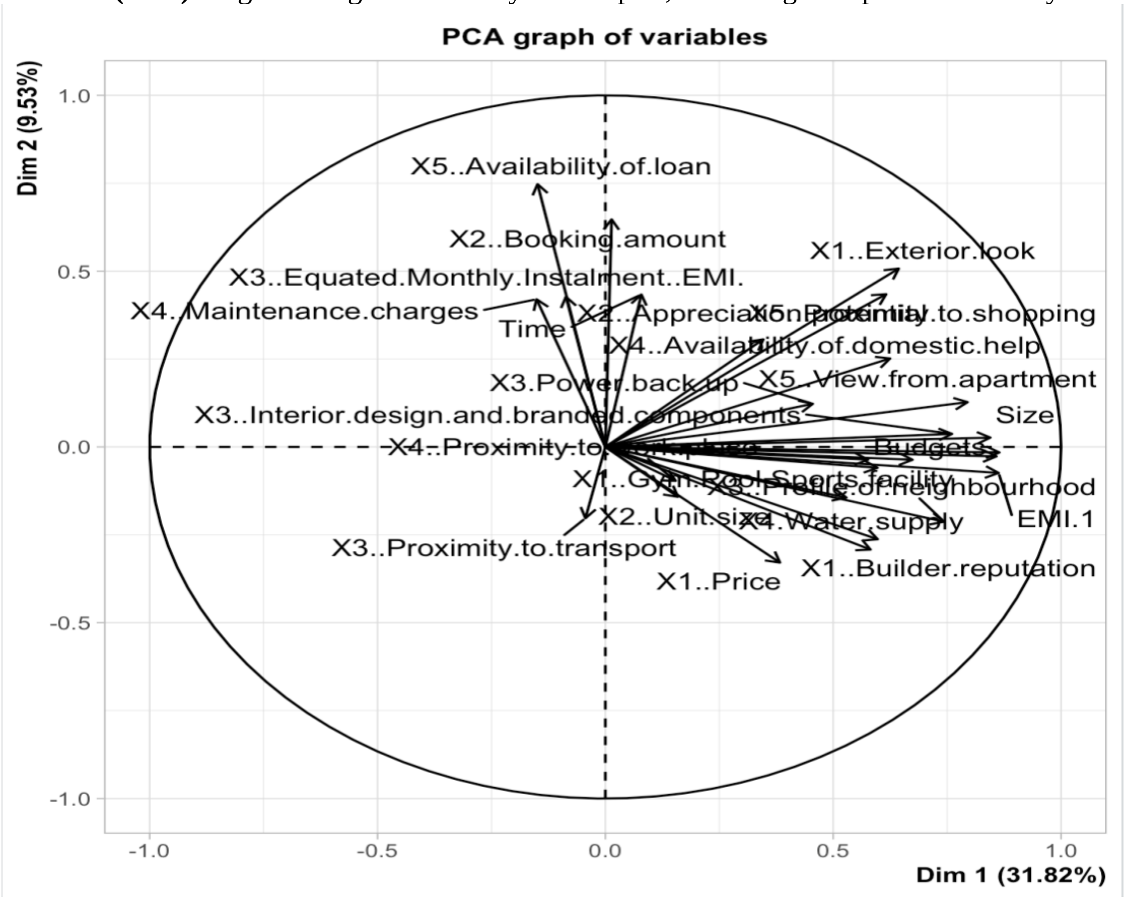
Detailed Variable Contributions (First 10 Variables):

- **X3..Proximity.to.transport:** Contributes more to Dim.3.
- **X4..Proximity.to.work.place:** Higher contribution to Dim.1 and Dim.2.

- Each variable's contributions help understand which dimensions they influence most.

General Interpretation:

- **Factor 1 (RC1):** Related to variables like Builder reputation, Size, Budgets, Maintainances, and EMI.1. This could represent an overarching factor related to the overall quality and cost aspects of the apartments.
- **Factor 2 (RC5):** Strongly related to Proximity to shopping, Security, Availability of domestic help. This might represent convenience and security.
- **Factor 3 (RC2):** Related to financial aspects like Booking amount, EMI, Availability of loan.
- **Factor 4 (RC4):** Influenced by factors like Proximity to work place, Power backup.
- **Factor 5 (RC3):** High loading on Proximity to transport, indicating transport accessibility.



The provided image is a PCA (Principal Component Analysis) biplot of variables. PCA is a statistical technique used to emphasize variation and bring out strong patterns in a dataset. It does this by transforming the data into a set of linearly uncorrelated variables called principal components.

Here's an interpretation of the PCA biplot:

1. **Axes and Variance Explained:**
   - The x-axis (Dim 1) explains 31.82% of the total variance in the data.
   - The y-axis (Dim 2) explains 9.53% of the total variance in the data.
   - Together, these two dimensions explain 41.35% of the variance in the dataset.
2. **Length and Direction of Arrows:**
   - Each arrow represents a variable in the dataset.

    - The length of the arrow indicates how much that variable contributes to the principal components. Longer arrows indicate variables that have a stronger influence on the principal components.
    - The direction of the arrow indicates the correlation of the variable with the principal components. Variables pointing in the same direction are positively correlated, while those pointing in opposite directions are negatively correlated.
3. **Variable Clusters:**
    - Variables that are close to each other are positively correlated.
    - For example, "X5..Availability.of.loan" and "X2..Booking.amount" are close to each other, indicating that these two variables are positively correlated.
    - Variables that form a 90-degree angle are uncorrelated.
    - Variables on opposite sides of the origin are negatively correlated.
4. **Interpretation of Specific Variables:**
    - Variables such as "X1..Price", "X1..Builder.reputation", and "X1..Neighbourhood" are clustered together, suggesting that they are positively correlated and contribute similarly to the principal components.
    - Variables like "X4..Proximity.to.transport" and "X3..Interior.design.and.branded.components" are positioned far from each other, indicating a potential negative correlation.
5. **Principal Components:**
    - Dim 1 (x-axis) might be representing a gradient of variables related to financial and proximity factors, as it has high contributions from "X5..Availability.of.loan" and "X2..Booking.amount".
    - Dim 2 (y-axis) might represent a gradient of aesthetic and convenience factors, with high contributions from variables like "X1..Exterior.look" and "X3..Equated.Monthly.Instalment..EMI".

The PCA biplot helps in understanding the relationships between variables and how they contribute to the principal components, providing insights into the underlying structure of the data.

Conclusion:

- The analysis extracted five significant factors explaining about 61% of the variance.
- Each factor represents a cluster of related variables, suggesting underlying dimensions in the data.

The PCA provides insights into the underlying structure of the data, helping to reduce its complexity and identify key patterns and relationships among the variables. If you have specific questions or need further details on any part of the analysis, please let me know!

Interpretation:

2. Conduct Cluster Analysis to characterise respondents based on background variables (Survey.csv Download Survey.csv)

Code and Result

```
# Function to auto-install and load packages
install_and_load <- function(packages) {
  for (package in packages) {
    if (!require(package, character.only = TRUE)) {
      install.packages(package, dependencies = TRUE)
    }
    library(package, character.only = TRUE)
  }
}

# List of packages to install and load
packages <- c("cluster", "FactoMineR", "factoextra", "pheatmap")

install_and_load(packages)
setwd('/Users/kirthanshaker/Desktop/SCMA 631 Data Files ')
survey_df<-read.csv('/Users/kirthanshaker/Desktop/SCMA 631 Data Files
/Survey.csv',header=TRUE)
sur_int=survey_df[,20:46]


#B) Carry our cluster analysis and characterize the respondents based on their
background variables.
library(cluster)
library(factoextra)
show(sur_int)
fviz_nbclust(sur_int,kmeans,method = "gap_stat")
set.seed(123)
km.res<-kmeans(sur_int,4,nstart = 25)
fviz_cluster(km.res,data=sur_int,palette="jco",
             ggtheme = theme_minimal())
res.hc <- hclust(dist(sur_int), method = "ward.D2")
fviz_dend(res.hc,cex=0.5,k=4,palette = "jco")
library(pheatmap)
pheatmap(t(sur_int),cutree_cols = 4)
```

Result:

Interpretation:

3. Apply Multidimensional Scaling and interpret the results (icecream.csv Download icecream.csv4.

Code and Result:

C) Do multidimensional scaling and interpret the results.

icecream_df<-read.csv('E:\\Code\\Notebooks\\Classes\\Icecream.csv',header=TRUE)
dim(icecream_df)

names(icecream_df)

ice<-subset(icecream_df,select = -c(Brand))
distance_matrix<-dist(ice)

mds_result<-cmdscale(distance_matrix,k=2)

```
plot(mds_result[,1],mds_result[,2],pch=16,xlab="Dimension1",ylab="Dimension2",main="MDS plot")
```

4. Conjoint Analysis (pizza_data.csv)Download pizza_data.csv)

```
Code : # Function to auto-install and load packages
install_and_load <- function(packages) {
  for (package in packages) {
    if (!require(package, character.only = TRUE)) {
      install.packages(package, dependencies = TRUE)
    }
    library(package, character.only = TRUE)
  }
}

# List of packages to install and load
packages <- c("dplyr", "psych", "tidyr", "GPArotation", "FactoMineR",
"factoextra", "pheatmap")

# Call the function
install_and_load(packages)

survey_df<-
read.csv('E:\\Code\\Notebooks\\Classes\\Survey.csv',header=TRUE)
sur_int=survey_df[,20:46]

#Factor Analysis

factor_analysis<-fa(sur_int,nfactors = 4,rotate = "varimax")
names(factor_analysis)
print(factor_analysis$loadings,reorder=TRUE)
fa.diagram(factor_analysis)
print(factor_analysis$communality)
print(factor_analysis$scores)
```

Results :

Interpretation :

```
import pandas as pd, numpy as np
df=pd.read_csv('pizza_data.csv')
```

Conjoint Analysis

We want to understand which combination of attributes & levels of pizza is most and least preferred by customers while choosing or ordering pizza so that the marketing team can enter the market with the best combinations.

The first step is to define the attributes and levels of the product.

We will take eight different attributes, namely 'brand,' 'price,' 'weight,' 'crust,' 'cheese,' 'size,' 'toppings,' and 'spicy,' where brand, price, and weight have four levels each and rest of the attributes have two levels.

The next step is to select the number of combinations or profiles. Here, we have a total 4*4*4*2*2*2*2*2 number of combinations. But we will not use all combinations since the company may not be able to produce some combinations, and the customers may not prefer some combinations. So, we will go with the selected 16 combinations and their rankings from a survey. We will load the dataset in the proper format.

In [14]:

```
import pandas as pd, numpy as np
df=pd.read_csv('pizza_data.csv')
```

We will now estimate each attribute level's effects using Linear Regression Model.

In [10]:

```
import statsmodels.api as sm
import statsmodels.formula.api as smf

model='ranking ~
C(brand,Sum)+C(price,Sum)+C(weight,Sum)+C(crust,Sum)+C(cheese,Sum)+C(size,Sum)+C(to
ppings,Sum)+C(spicy,Sum)'
model_fit=smf.ols(model,data=df).fit()
print(model_fit.summary())
                            OLS Regression Results
==============================================================================
Dep. Variable:                ranking   R-squared:                       0.999
Model:                            OLS   Adj. R-squared:                  0.989
Method:                 Least Squares   F-statistic:                     97.07
Date:                Sat, 06 Jul 2024   Prob (F-statistic):             0.0794
Time:                        01:13:15   Log-Likelihood:                 10.568
No. Observations:                  16   AIC:                             8.864
Df Residuals:                       1   BIC:                             20.45
Df Model:                          14
Covariance Type:            nonrobust
===================================================================================
=============
                                   coef    std err          t      P>|t|      [0.02
5      0.975]
-----------------------------------------------------------------------------------
-------------
Intercept                        8.5000      0.125     68.000      0.009       6.91
2      10.088
C(brand, Sum)[S.Dominos]       2.22e-15      0.217   1.03e-14      1.000      -2.75
1       2.751
C(brand, Sum)[S.Onesta]       8.882e-15      0.217    4.1e-14      1.000      -2.75
1       2.751
```

```
C(brand, Sum)[S.Oven Story]      -0.2500      0.217     -1.155      0.454      -3.00
1       2.501
C(price, Sum)[S.$1.00]            0.7500      0.217      3.464      0.179      -2.00
1       3.501
C(price, Sum)[S.$2.00]         2.109e-15      0.217   9.74e-15      1.000      -2.75
1       2.751
C(price, Sum)[S.$3.00]        -4.885e-15      0.217  -2.26e-14      1.000      -2.75
1       2.751
C(weight, Sum)[S.100g]            5.0000      0.217     23.094      0.028       2.24
9       7.751
C(weight, Sum)[S.200g]            2.0000      0.217      9.238      0.069      -0.75
1       4.751
C(weight, Sum)[S.300g]           -1.2500      0.217     -5.774      0.109      -4.00
1       1.501
C(crust, Sum)[S.thick]            1.7500      0.125     14.000      0.045       0.16
2       3.338
C(cheese, Sum)[S.Cheddar]        -0.2500      0.125     -2.000      0.295      -1.83
8       1.338
C(size, Sum)[S.large]            -0.2500      0.125     -2.000      0.295      -1.83
8       1.338
C(toppings, Sum)[S.mushroom]      1.1250      0.125      9.000      0.070      -0.46
3       2.713
C(spicy, Sum)[S.extra]            0.7500      0.125      6.000      0.105      -0.83
8       2.338
==============================================================================
Omnibus:                       29.718   Durbin-Watson:                   2.000
Prob(Omnibus):                  0.000   Jarque-Bera (JB):                2.667
Skew:                           0.000   Prob(JB):                        0.264
Kurtosis:                       1.000   Cond. No.                         2.00
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly sp
ecified.
C:\Users\Patil\AppData\Local\Programs\Python\Python312\Lib\site-packages\scipy\stat
s\_axis_nan_policy.py:418: UserWarning: `kurtosistest` p-value may be inaccurate wi
th fewer than 20 observations; only n=16 observations were given.
  return hypotest_fun_in(*args, **kwds)
```

We can analyze the model's fitness using parameters like R-squared, p-values, etc. The coefficients of each attribute level define its effect on the overall choice model.

Now, we will create the list of conjoint attributes.

In [11]:

```
conjoint_attributes =
['brand','price','weight','crust','cheese','size','toppings','spicy']
```

Before going ahead, we need to understand these conjoint analysis terminologies:

Relative importance: It depicts which attributes are more or less important when purchasing. E.g., a Mobile Phone's Relative importance could be Brand 30%, Price 30%, Size 20%, Battery Life 10%, and Color 10%.

Part-Worths/Utility values: The amount of weight an attribute level carries with a respondent. These factors lead to a product's overall value to consumers.

Next, we will build part-worths information and calculate attribute-wise importance level.

In [15]:

```
level_name = []
```

```
part_worth = []
part_worth_range = []
important_levels = {}
end = 1  # Initialize index for coefficient in params

for item in conjoint_attributes:
    nlevels = len(list(np.unique(df[item])))
    level_name.append(list(np.unique(df[item])))

    begin = end
    end = begin + nlevels -1

    new_part_worth = list(model_fit.params[begin:end])
    new_part_worth.append((-1)*sum(new_part_worth))
    important_levels[item] = np.argmax(new_part_worth)
    part_worth.append(new_part_worth)
    print(item)
    #print(part_worth)
    part_worth_range.append(max(new_part_worth) - min(new_part_worth))
    # next iteration
print("-------------------------------------------------------------")
print("level name:")
print(level_name)
print("npw with sum element:")
print(new_part_worth)
print("imp level:")
print(important_levels)
print("part worth:")
print(part_worth)
print("part_worth_range:")
print(part_worth_range)
print(len(part_worth))
print("important levels:")
print(important_levels)
brand
price
weight
crust
cheese
size
toppings
spicy
-------------------------------------------------------------
level name:
[['Dominos', 'Onesta', 'Oven Story', 'Pizza hut'], ['$1.00', '$2.00', '$3.00', '$4.
00'], ['100g', '200g', '300g', '400g'], ['thick', 'thin'], ['Cheddar', 'Mozzarella'
], ['large', 'regular'], ['mushroom', 'paneer'], ['extra', 'normal']]
npw with sum element:
[0.7499999999999993, -0.7499999999999993]
imp level:
{'brand': np.int64(3), 'price': np.int64(0), 'weight': np.int64(0), 'crust': np.int
64(0), 'cheese': np.int64(1), 'size': np.int64(1), 'toppings': np.int64(0), 'spicy'
: np.int64(0)}
part worth:
[[2.220446049250313e-15, 8.881784197001252e-15, -0.25000000000000827, 0.24999999999
999717], [0.7500000000000013, 2.1094237467877974e-15, -4.884981308350689e-15, -0.74
99999999999986], [5.000000000000002, 2.0000000000000018, -1.2500000000000004, -5.75
00000000000036], [1.7499999999999996, -1.7499999999999996], [-0.2500000000000009, 0
.2500000000000009], [-0.2500000000000018, 0.2500000000000018], [1.1249999999999991,
-1.1249999999999991], [0.7499999999999993, -0.7499999999999993]]
part_worth_range:
[0.5000000000000054, 1.5, 10.750000000000005, 3.499999999999999, 0.5000000000000018
, 0.5000000000000036, 2.2499999999999982, 1.4999999999999987]
8
important levels:
{'brand': np.int64(3), 'price': np.int64(0), 'weight': np.int64(0), 'crust': np.int
64(0), 'cheese': np.int64(1), 'size': np.int64(1), 'toppings': np.int64(0), 'spicy'
: np.int64(0)}
```

Now, we will calculate the importance of each attribute.

In [16]:

```
attribute_importance = []
for i in part_worth_range:
    #print(i)
    attribute_importance.append(round(100*(i/sum(part_worth_range)),2))
print(attribute_importance)
[2.38, 7.14, 51.19, 16.67, 2.38, 2.38, 10.71, 7.14]
```

Now, we will calculate the part-worths of each attribute level.

In [30]:

```
part_worth_dict={}
attrib_level={}
for item,i in zip(conjoint_attributes,range(0,len(conjoint_attributes))):
    print("Attribute :",item)
    print("    Relative importance of attribute ",attribute_importance[i])
    print("    Level wise part worths: ")
    for j in range(0,len(level_name[i])):
        print(i)
        print(j)
        print("          {}:{}".format(level_name[i][j],part_worth[i][j]))
        part_worth_dict[level_name[i][j]]=part_worth[i][j]
        attrib_level[item]=(level_name[i])
        #print(j)
part_worth_dict
Attribute : brand
    Relative importance of attribute  2.38
    Level wise part worths:
0
0
          Dominos:2.220446049250313e-15
0
1
          Onesta:8.881784197001252e-15
0
2
          Oven Story:-0.25000000000000827
0
3
          Pizza hut:0.24999999999999717
Attribute : price
    Relative importance of attribute  7.14
    Level wise part worths:
1
0
          $1.00:0.7500000000000013
```

```
1
1
          $2.00:2.1094237467877974e-15
1
2
          $3.00:-4.884981308350689e-15
1
3
          $4.00:-0.7499999999999986
Attribute : weight
    Relative importance of attribute  51.19
    Level wise part worths:
2
0
          100g:5.000000000000002
2
1
          200g:2.0000000000000018
2
2
          300g:-1.2500000000000004
2
3
          400g:-5.7500000000000036
Attribute : crust
    Relative importance of attribute  16.67
    Level wise part worths:
3
0
          thick:1.7499999999999996
3
1
          thin:-1.7499999999999996
Attribute : cheese
    Relative importance of attribute  2.38
    Level wise part worths:
4
0
          Cheddar:-0.2500000000000009
4
1
          Mozzarella:0.2500000000000009
Attribute : size
    Relative importance of attribute  2.38
    Level wise part worths:
5
0
          large:-0.2500000000000018
5
1
          regular:0.2500000000000018
Attribute : toppings
    Relative importance of attribute  10.71
    Level wise part worths:
6
0
          mushroom:1.1249999999999991
```

```
6
1
          paneer:-1.1249999999999991
Attribute : spicy
    Relative importance of attribute  7.14
    Level wise part worths:
7
0
          extra:0.7499999999999993
7
1
          normal:-0.7499999999999993
```

Out[30]:

```
{'Dominos': 2.220446049250313e-15,
 'Onesta': 8.881784197001252e-15,
 'Oven Story': -0.25000000000000827,
 'Pizza hut': 0.24999999999999717,
 '$1.00': 0.7500000000000013,
 '$2.00': 2.1094237467877974e-15,
 '$3.00': -4.884981308350689e-15,
 '$4.00': -0.7499999999999986,
 '100g': 5.000000000000002,
 '200g': 2.0000000000000018,
 '300g': -1.2500000000000004,
 '400g': -5.7500000000000036,
 'thick': 1.7499999999999996,
 'thin': -1.7499999999999996,
 'Cheddar': -0.2500000000000009,
 'Mozzarella': 0.2500000000000009,
 'large': -0.2500000000000018,
 'regular': 0.2500000000000018,
 'mushroom': 1.1249999999999991,
 'paneer': -1.1249999999999991,
 'extra': 0.7499999999999993,
 'normal': -0.7499999999999993}
```

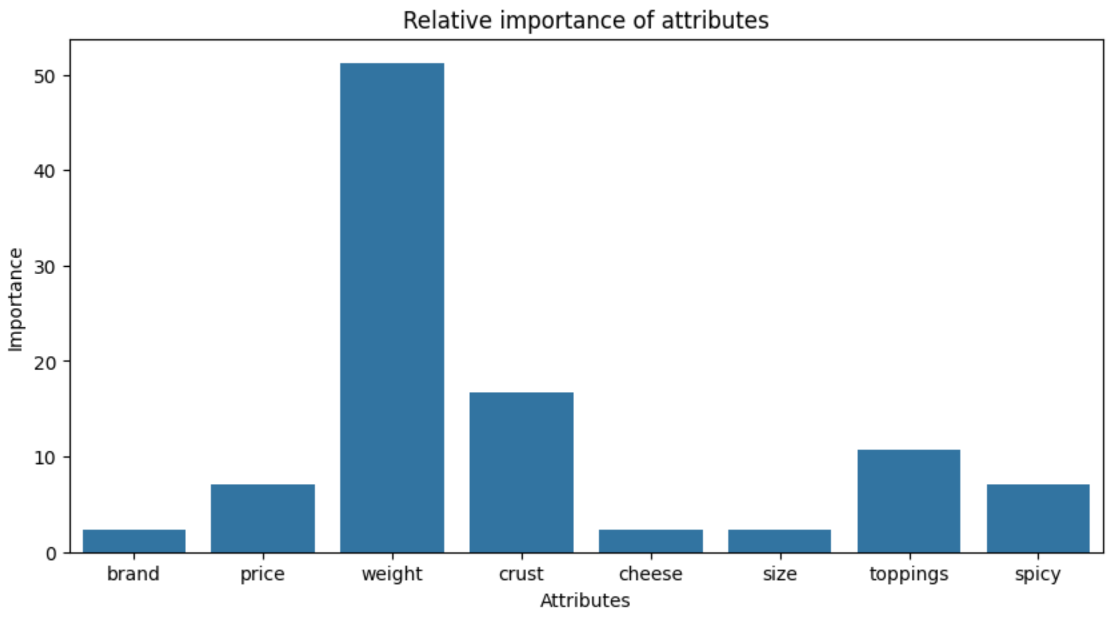
In the next step, we will plot the relative importance of attributes.

In [31]:

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(10,5))
sns.barplot(x=conjoint_attributes,y=attribute_importance)
plt.title('Relative importance of attributes')
plt.xlabel('Attributes')
plt.ylabel('Importance')
```

Out[31]:

```
Text(0, 0.5, 'Importance')
```



We can see that weight is the attribute with the highest relative importance at 51%, followed by crust at 16% and toppings at 10%. Brand, cheese, and size are the least important attributes, each at 2.38%.

Now, we will calculate the utility score for each profile.

In [36]:

```
utility = []
for i in range(df.shape[0]):
    score =
part_worth_dict[df['brand'][i]]+part_worth_dict[df['price'][i]]+part_worth_dict[df[
'weight'][i]]+part_worth_dict[df['crust'][i]]+part_worth_dict[df['cheese'][i]]+part
_worth_dict[df['size'][i]]+part_worth_dict[df['toppings'][i]]+part_worth_dict[df['s
picy'][i]]
    utility.append(score)

df['utility'] = utility
utility
```

Out[36]:

```
[2.6250000000000098,
 3.374999999999992,
 0.3750000000000149,
 -6.375000000000003,
 -0.3749999999999998,
 4.374999999999999,
 -1.3749999999999962,
 -4.624999999999999,
 -3.625,
 7.624999999999992,
 -5.374999999999997,
 -2.3750000000000053,
 1.3749999999999927,
 6.375000000000013,
 -7.625000000000013,
 5.624999999999997]
```

We can see that combination number 9 has the maximum utility, followed by combination numbers 13 and 5. Combination number 14 is the least desirable because of the most negative utility score.

Now, we will find the combination with maximum utility.

In [37]:

```
print("The profile that has the highest utility score :",'\n',
df.iloc[np.argmax(utility)])
The profile that has the highest utility score :
 brand       Oven Story
price            $4.00
weight            100g
crust            thick
cheese      Mozzarella
size             large
toppings      mushroom
spicy            extra
ranking             16
utility          7.625
Name: 9, dtype: object
```

Now, we will determine the levels being preferred in each attribute.

In [38]:

```
for i,j in zip(attrib_level.keys(),range(0,len(conjoint_attributes))):
    #print(i)
    #level_name[j]
    print("Preferred level in {} is ::
{}".format(i,level_name[j][important_levels[i]]))
Preferred level in brand is :: Pizza hut
Preferred level in price is :: $1.00
Preferred level in weight is :: 100g
Preferred level in crust is :: thick
Preferred level in cheese is :: Mozzarella
Preferred level in size is :: regular
Preferred level in toppings is :: mushroom
Preferred level in spicy is :: extra
```

```
# Function to auto-install and load packages
install_and_load <- function(packages) {
  for (package in packages) {
    if (!require(package, character.only = TRUE)) {
      install.packages(package, dependencies = TRUE)
    }
    library(package, character.only = TRUE)
  }
}

# List of packages to install and load
packages <- c("dplyr", "psych", "tidyr", "GPArotation", "FactoMineR", "factoextra", "pheatmap")

# Call the function
install_and_load(packages)

survey_df<-read.csv('E:\\Code\\Notebooks\\Classes\\Survey.csv',header=TRUE)
dim(survey_df)
names(survey_df)
head(survey_df)
str(survey_df)


#A)Do principal component analysis and factor analysis and identify the dimensions in the data.
```

```
is.na(survey_df)
sum(is.na(survey_df))
sur_int=survey_df[,20:46]
str(sur_int)
dim(sur_int)
library(GPArotation)
pca <- principal(sur_int,5,n.obs =162, rotate ="promax")
pca

om.h<-omega(sur_int,n.obs=162,sl=FALSE)
op<-par(mfrow=c(1,1))
om<-omega(sur_int,n.obs=162)
library(FactoMineR)
pca<-PCA(sur_int,scale.unit = TRUE)
summary(pca)
biplot(pca, scale = 0)
str(sur_int)
dim(sur_int)
show(sur_int)




# Function to auto-install and load packages
install_and_load <- function(packages) {
  for (package in packages) {
    if (!require(package, character.only = TRUE)) {
      install.packages(package, dependencies = TRUE)
    }
    library(package, character.only = TRUE)
  }
}

# List of packages to install and load
packages <- c("dplyr", "psych", "tidyr", "GPArotation", "FactoMineR", "factoextra", "pheatmap")

# Call the function
install_and_load(packages)

survey_df<-read.csv('E:\\Code\\Notebooks\\Classes\\Survey.csv',header=TRUE)
sur_int=survey_df[,20:46]

#Factor Analysis

factor_analysis<-fa(sur_int,nfactors = 4,rotate = "varimax")
names(factor_analysis)
print(factor_analysis$loadings,reorder=TRUE)
fa.diagram(factor_analysis)
print(factor_analysis$communality)
print(factor_analysis$scores)
#C) Do multidimensional scaling and interpret the results.

icecream_df<-read.csv('E:\\Code\\Notebooks\\Classes\\Icecream.csv',header=TRUE)
dim(icecream_df)

names(icecream_df)

ice<-subset(icecream_df,select = -c(Brand))
distance_matrix<-dist(ice)
```

```
mds_result<-cmdscale(distance_matrix,k=2)

plot(mds_result[,1],mds_result[,2],pch=16,xlab="Dimension1",ylab="Dimension2",main="MDS plot")
```