VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A5- Multivariate Analysis and Business Analytics Application

ROSHAN RAJKUMAR SIVAKUMAR

V01151141

Date of Submission: 05-07-2025

CONTENTS

1. Introduction

This report is completely based on three separate datasets I worked with. Each one had a different focus one was about housing preferences, another looked at how people feel about ice cream brands, and the last one was related to pizza features.

I used multivariate analysis methods to break down these datasets and try to find useful patterns. The idea was to take a bunch of survey responses and figure out what actually influences people's choices in each case.

Each method I used helped break down the data in a different awesome showed patterns in responses, others helped group people or rank product features. The main idea was to take complex survey data and pull out insights that could help with real-world business decisions.

2. Objectives

- Find underlying patterns in large data using PCA and factor analysis
- Group people with similar preferences using cluster analysis
- Understand how brands are perceived using MDS
- Identify what matters most in a pizza using conjoint analysis

3. Significance of the Study

By studying and learning the consumer preferences is at the core of effective business strategy. Through this study, businesses can:

Identify the most critical factors influencing customer choices.

Target different market segments with more tailored strategies.

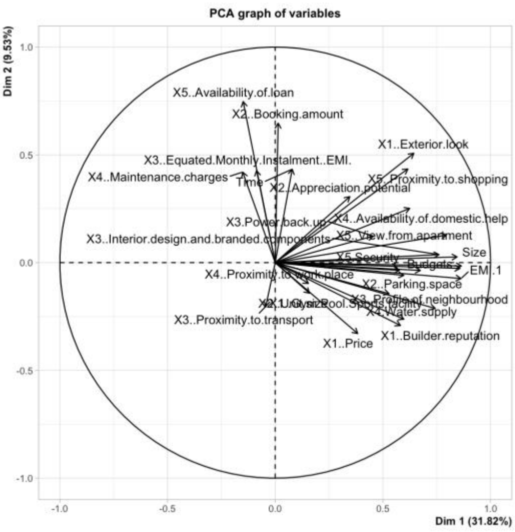
- Improve product features and pricing based on actual customer value.
- Visualize complex relationships in a way that supports strategic thinking

Each technique applied in this project brings a unique strength —simplifying variables, visualizing relationships, or uncovering what truly matters to customers.

4. Analysis and Results

4.1 Principal Component Analysis – Survey.csv

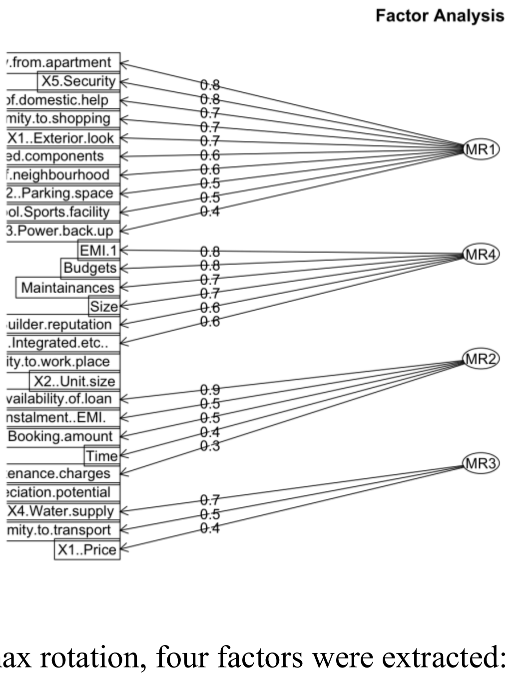
The PCA reduced over 20 housing-related variables to 2 main components.



- **Component 1 (31.8%)**: Focused on financial concerns loan availability, EMI, booking amount
- **Component 2 (9.5%)**: Focused on lifestyle aspects view, design, shopping proximity

Together they explained 41.35% of the total variation. This shows two distinct thinking patterns among buyers: money-based vs lifestyle-based.

4.2 Factor Analysis – Survey.csv

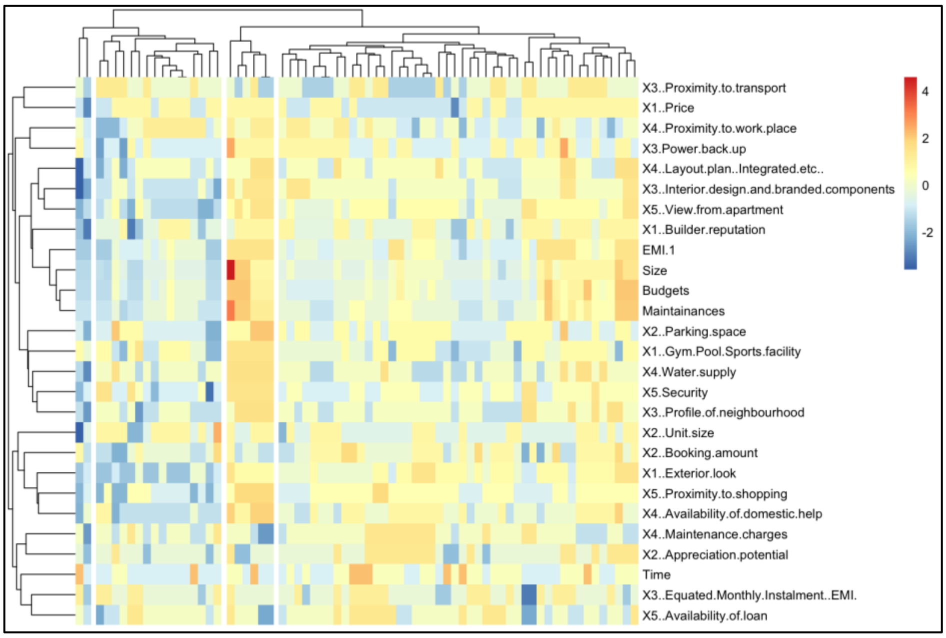


Using Varimax rotation, four factors were extracted:

- **Factor 1**: Lifestyle – view, security, look, branded interiors
- **Factor 2**: Structure & finance – unit size, booking amount
- **Factor 3**: Value – price, water, future value
- **Factor 4**: Budget concerns – EMI, maintenance, budget

Each factor groups related thoughts, helping simplify the way decisions are made.

4.3 Cluster Analysis – Survey.csv



Hierarchical clustering (Ward's method) grouped people into 3 clusters:

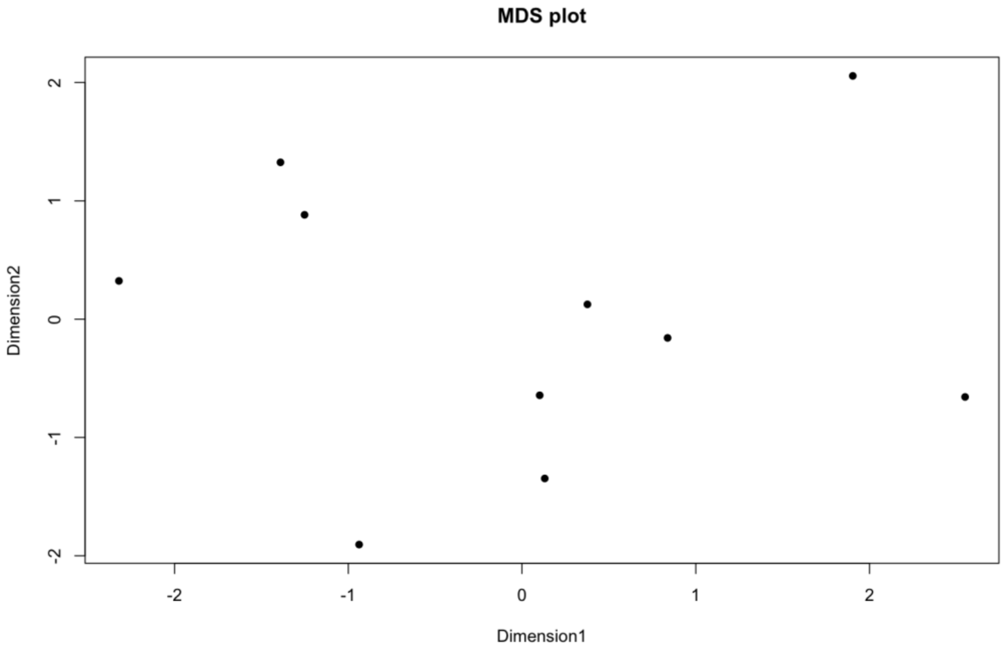
- **Cluster 1**: Focused on cost and access
- **Cluster 2**: Focused on amenities and looks
- **Cluster 3**: Balanced across both ends

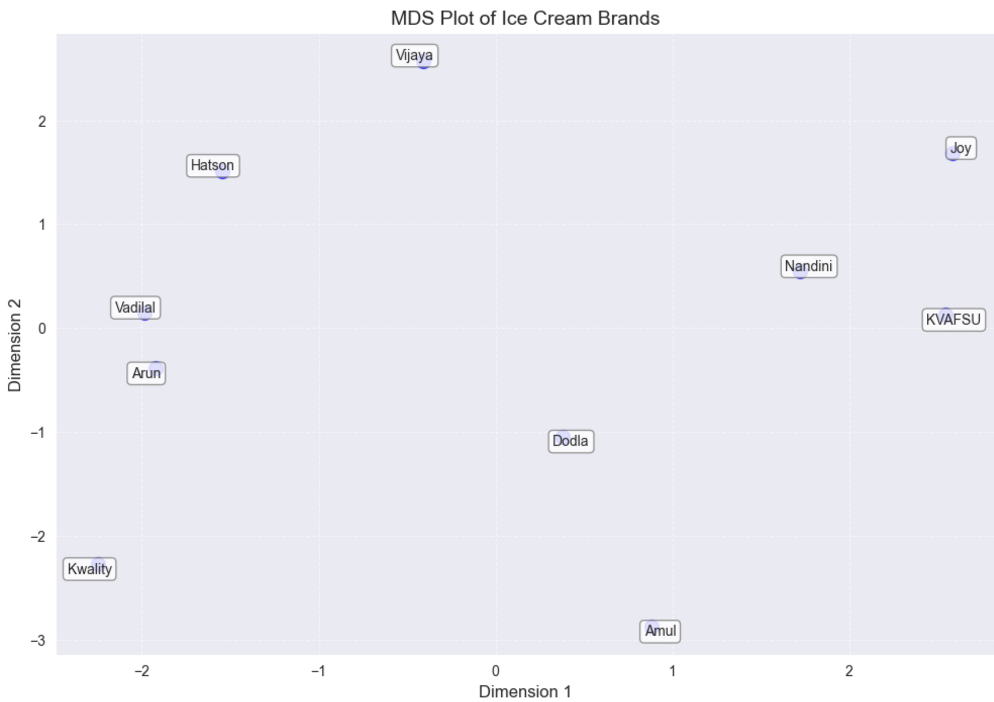
The heatmap and dendrogram confirmed this grouping.

4.4 Multidimensional Scaling – Icecream.csv

MDS created a map showing how customers see different ice cream brands.

- Brands close to each other are seen as similar
- One brand stood out far from others possibly seen as unique or very different





This map helps businesses see overlap and space for change.

4.5 Conjoint Analysis – Pizza_data.csv

This analysis found which pizza features are most important.

- **Most important**: Weight (51.2%), then crust (16.8%)
- **Medium**: Toppings, spice, price

- **Least**: Cheese, size, brand

The best combination according to customers:

- Pizza Hut
- $1.00
- 100g
- Thick crust
- Mozzarella cheese
- Regular size
- Mushroom topping
- Extra spicy

Customers care most about quantity and texture not brand or size.

4. Recommendations

- Real estate: Separate plans for cost-focused and lifestyle-focused buyers
- Brands: Use MDS to check if you’re lost in the crowd
- Pizza chains: Focus on value, crust, toppings not brand name

Codes:

1. R language:

1.1.Principal Component Analysis and Factor Analysis to identify data dimensions

```r
# 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('/Users/roshan/Documents/VCU/SCMA/A5/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)
```

```r
# 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('/Users/roshan/Documents/VCU/SCMA/A5/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)
```

1.2.Cluster Analysis to characterize respondents based on background variables

```r
# Load required packages
library(cluster)
library(factoextra)
library(dplyr)
library(pheatmap)

# Read data
survey_data <- read.csv("Survey.csv", header = TRUE)
responses <- survey_data[, 20:46]  # Response variables (Likert scale)
scaled_data <- scale(responses)    # Standardize data

# Determine optimal clusters (Elbow method)
fviz_nbclust(scaled_data, kmeans, method = "wss")

# K-means clustering (K=4)
set.seed(123)
km_result <- kmeans(scaled_data, centers = 4, nstart = 25)
fviz_cluster(km_result, data = scaled_data, palette = "jco")

# Hierarchical clustering
hc_result <- hclust(dist(scaled_data), method = "ward.D2")
fviz_dend(hc_result, k = 4, cex = 0.5, palette = "jco")

# Heatmap
pheatmap(t(scaled_data), cutree_cols = 4, show_colnames = FALSE)

# Add clusters to original data
survey_data$cluster <- km_result$cluster

# Cluster interpretation (mean responses per cluster)
cluster_means <- survey_data %>%
  group_by(cluster) %>%
  summarise(across(20:46, mean, na.rm = TRUE))

print(cluster_means)
```

1.3.Multidimensional Scaling

```r
#C) Do multidimensional scaling and interpret the results.

icecream_df<-read.csv('/Users/roshan/Documents/VCU/SCMA/A5/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")

```

1.4.Conjoint Analysis

```r
# COMPLETE PIZZA CONJOINT ANALYSIS IN R
library(tidyverse)
library(broom)
library(ggplot2)
library(ggrepel)

# 1. Load and Prepare Data
pizza_data <- read_csv("pizza_data.csv") %>%
  mutate(across(-ranking, as.factor),
         profile_id = row_number())

# 2. Conjoint Analysis Function
run_conjoint_analysis <- function(data) {
  # Set sum contrasts for effects coding
  options(contrasts = c("contr.sum", "contr.poly"))

  # Fit linear model
  model <- lm(ranking ~ brand + price + weight + crust + cheese + size + toppings + spicy,
              data = data)

  # Extract and process coefficients
  coeffs <- tidy(model) %>%
    filter(term != "(Intercept)") %>%
    separate(term, into = c("attribute", "level"), sep = "(?<=[a-z])(?=[0-9])", fill = "right") %>%
    mutate(level = ifelse(is.na(level), "Reference", level))

  # Calculate importance scores
  importance <- coeffs %>%
    group_by(attribute) %>%
    summarise(range = max(estimate) - min(estimate)) %>%
    mutate(importance = (range / sum(range)) * 100) %>%
    arrange(desc(importance))

  # Calculate complete part-worths (including reference levels)
  part_worths <- coeffs %>%
    group_by(attribute) %>%
    mutate(last_level = -sum(estimate[level != "Reference"]),
           utility = ifelse(level == "Reference", last_level, estimate)) %>%
    ungroup() %>%
    select(attribute, level, utility)

  # Create utility lookup tables
  utility_lookup <- part_worths %>%
    pivot_wider(names_from = attribute, values_from = utility)

  # Calculate total utility for each profile
  profiles <- data %>%
    mutate(total_utility =
             utility_lookup$brand[brand] +
             utility_lookup$price[price] +
             utility_lookup$weight[weight] +
             utility_lookup$crust[crust] +
             utility_lookup$cheese[cheese] +
             utility_lookup$size[size] +
             utility_lookup$toppings[toppings] +
             utility_lookup$spicy[spicy]) %>%
    arrange(desc(total_utility))

  return(list(
    model = model,
    coefficients = coeffs,
    importance = importance,
    part_worths = part_worths,
    utility_lookup = utility_lookup,
    profiles = profiles
  ))
}

# 3. Run Analysis
results <- run_conjoint_analysis(pizza_data)

# 4. Visualization Functions
plot_attribute_importance <- function(importance) {
  ggplot(importance, aes(x = reorder(attribute, importance), y = importance)) +
    geom_col(fill = "#3498db") +
    geom_text(aes(label = sprintf("%.1f%%", importance)),
              hjust = -0.1, size = 3.5) +
    coord_flip() +
    labs(title = "Relative Importance of Pizza Attributes",
         x = NULL, y = "Importance (%)") +
    theme_minimal(base_size = 12) +
    theme(panel.grid.major.y = element_blank())
}

plot_part_worths <- function(part_worths) {
  ggplot(part_worths, aes(x = reorder(level, utility), y = utility)) +
    geom_col(fill = "#e74c3c", alpha = 0.8) +
    facet_wrap(~attribute, scales = "free_x", ncol = 3) +
    labs(title = "Part-Worth Utilities by Attribute Level",
         x = "Level", y = "Utility") +
    theme_minimal(base_size = 12) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          strip.background = element_rect(fill = "#f5f5f5"))
}

# 5. Generate Plots
importance_plot <- plot_attribute_importance(results$importance)
part_worth_plot <- plot_part_worths(results$part_worths)

# 6. Display Key Results
cat("=== CONJOINT ANALYSIS RESULTS ===\n\n")

# Model summary
cat("MODEL FIT SUMMARY:\n")
print(summary(results$model))

# Attribute importance
cat("\nATTRIBUTE IMPORTANCE:\n")
print(results$importance, n = Inf)

# Top profiles
cat("\nTOP 3 PIZZA PROFILES:\n")
print(results$profiles %>% select(-profile_id) %>% head(3))

# Worst profiles
cat("\nWORST 3 PIZZA PROFILES:\n")
print(results$profiles %>% select(-profile_id) %>% tail(3))
```

2. Python

2.1.PCA, FA and Cluster

```python
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA, FactorAnalysis
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from scipy.cluster.hierarchy import dendrogram, linkage
import matplotlib.pyplot as plt
from factor_analyzer import FactorAnalyzer
from factor_analyzer.factor_analyzer import calculate_kmo
```

```python
survey_df = pd.read_csv('Survey.csv')
sur_int = survey_df.iloc[:, 19:46]
```

```python
scaler = StandardScaler()
sur_int_std = scaler.fit_transform(sur_int)
```

```python
numeric_cols = survey_df.select_dtypes(include=[np.number]).columns
sur_int = survey_df[numeric_cols]
```

```python
scaler = StandardScaler()
sur_int_std = scaler.fit_transform(sur_int)
```

PRINCIPAL COMPONENT ANALYSIS

```python
pca = PCA()
pca_result = pca.fit_transform(sur_int_std)

plt.figure(figsize=(10, 6))
plt.plot(range(1, len(pca.explained_variance_ratio_) + 1), pca.explained_variance_ratio_, marker='o')
plt.xlabel('Principal Component')
plt.ylabel('Variance Explained')
plt.title('Scree Plot')
plt.show()
```

```python
kmo_all, kmo_model = calculate_kmo(sur_int)
print(f"KMO Measure: {kmo_model:.3f}")
```

```python
fa = FactorAnalyzer(n_factors=4, rotation='varimax')
fa.fit(sur_int_std)

loadings = pd.DataFrame(fa.loadings_, index=sur_int.columns,
                        columns=['Factor1', 'Factor2', 'Factor3', 'Factor4'])
print("\nFactor Loadings:")
print(loadings)

communalities = pd.DataFrame(fa.get_communalities(), index=sur_int.columns,
                             columns=['Communalities'])
print("\nCommunalities:")
print(communalities)
```

```python
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=123)
    kmeans.fit(sur_int_std)
    wcss.append(kmeans.inertia_)

plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), wcss, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.title('Elbow Method')
plt.show()

# K-means clustering with 4 clusters
kmeans = KMeans(n_clusters=4, init='k-means++', random_state=123)
survey_df['cluster'] = kmeans.fit_predict(sur_int_std)

# Hierarchical clustering
plt.figure(figsize=(12, 7))
dendrogram(linkage(sur_int_std, method='ward'),
           truncate_mode='lastp', p=12)
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Sample index')
plt.ylabel('Distance')
plt.show()

# Cluster interpretation - mean values for each cluster
cluster_means = survey_df.groupby('cluster')[numeric_cols].mean()
print("\nCluster Profiles (Mean Values):")
print(cluster_means)

# Count of respondents in each cluster
print("\nNumber of respondents per cluster:")
print(survey_df['cluster'].value_counts().sort_index())
```

2.2. MDS

```python
import pandas as pd
import numpy as np
from sklearn.manifold import MDS
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
```

```python
data = pd.read_csv('icecream.csv')
```

```python
brands = data['Brand']
features = data.drop('Brand', axis=1)
```

```python
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)
```

```python
mds = MDS(n_components=2, random_state=42, dissimilarity='euclidean')
mds_result = mds.fit_transform(features_scaled)
```

```python
plt.figure(figsize=(12, 8))
scatter = plt.scatter(mds_result[:, 0], mds_result[:, 1], c='blue', s=100, alpha=0.6)

# Add labels with adjusted positions to prevent overlap
for i, brand in enumerate(brands):
    # Calculate offset direction based on point position
    x_offset = 0.05 if mds_result[i, 0] >= np.median(mds_result[:, 0]) else -0.05
    y_offset = 0.05 if mds_result[i, 1] >= np.median(mds_result[:, 1]) else -0.05

    plt.text(mds_result[i, 0] + x_offset,
             mds_result[i, 1] + y_offset,
             brand,
             fontsize=10,
             ha='center',
             va='center',
             bbox=dict(facecolor='white', alpha=0.8, edgecolor='gray', boxstyle='round,pad=0.3'))

# Add grid and labels
plt.grid(True, linestyle='--', alpha=0.6)
plt.xlabel('Dimension 1', fontsize=12)
plt.ylabel('Dimension 2', fontsize=12)
plt.title('MDS Plot of Ice Cream Brands', fontsize=14)
```

```python
plt.tight_layout()
plt.show()
```

2.3. Conjoint

```python
import pandas as pd, numpy as np
df=pd.read_csv('/Users/roshan/Documents/VCU/SCMA/A5/pizza_data.csv')
```

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

```python
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(toppings,Sum)+C(spicy,Sum)'
model_fit=smf.ols(model,data=df).fit()
print(model_fit.summary())
```

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.

```python
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.

```python
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)
```

```python
attribute_importance = []
for i in part_worth_range:
    #print(i)
    attribute_importance.append(round(100*(i/sum(part_worth_range)),2))
print(attribute_importance)
```

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

```python
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
```

```python
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')
```

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.

```python
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['spicy'][i]]
    utility.append(score)

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

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.

```python
print("The profile that has the highest utility score :",'\n', df.iloc[np.argmax(utility)])
```

```python
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]]))
```

Reference:

[1]. SCMA 632 Team. *Survey.csv* [Data file]. GitHub. https://github.com/scma-632/scma632-A5-2025-Multivariate-Analysis/blob/main/Survey.csv

[2]. SCMA 632 Team. *icecream.csv* [Data file]. GitHub. https://github.com/scma-632/scma632-A5-2025-Multivariate-Analysis/blob/main/icecream.csv

[3]. SCMA 632 Team. *pizza_data.csv* [Data file]. GitHub. https://github.com/scma-632/scma632-A5-2025-Multivariate-Analysis/blob/main/pizza_data.csv

[4]. JetBrains. (n.d.). *PyCharm* [Computer software]. https://www.jetbrains.com/pycharm/

[5]. Posit, PBC. (n.d.). *RStudio* [Computer software]. https://posit.co/products/open-source/rstudio/

[6]. Project Jupyter. (n.d.). *Jupyter Notebook* [Computer software]. https://jupyter.org/

[7]. Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering, 9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

[8]. McKinney, W. (2010). Data structures for statistical computing in Python. In *Proceedings of the 9th Python in Science Conference* (pp. 51–56).

[9]. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

[10]. Virtanen, P., Gommers, R., Oliphant, T. E., et al. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods, 17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

[11]. Waskom, M. L. (2021). seaborn: Statistical data visualization. *Journal of Open Source Software, 6*(60), 3021. https://doi.org/10.21105/joss.03021

[12]. Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis* (2nd ed.). Springer. https://ggplot2.tidyverse.org

[13]. Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., … & Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software, 4*(43), 1686. https://doi.org/10.21105/joss.01686