PS0002 Course Project

U2040600D Lee Dongheng: Introduction, Data Preparation and Preliminary Analysis
U2040020G Oh Chun Rong: ML for Clustering, Conclusion and Discussion
U2040410H Chan Luo Xi: ML for Classification, Conclusion and Discussion

1 Introduction

Tourism has become one of the world's most important growth engines and was unfortunately heavily strived by the COVID-19 pandemic. Although countries are slowly moving toward a phased relaxation of measures, the risk of sudden policy changes around air travel, visas, and quarantine requirements still remain elevated. (Manuela et al., 2021)

According to the Singapore Tourism Board, Singapore's international visitor arrivals and tourism receipts reached 330,000 and an estimated $1.9 billion respectively in 2021. While these numbers only represent a fraction of Singapore's tourism performance prior to the pandemic, there have been positive signs of recovery in the tourism sector in the last three quarters of 2021. The introduction of various travel arrangements, such as Vaccinated Travel Lanes, has encouraged the gradual return of international travellers.(Singapore Tourism Board, 2022) In line with the recovery of tourism, we expect to see numerous tour packages introduced by various tourism companies in the near future. Therefore, forming a well-targeted tour package is crucial before the launch of new packages.

In the dataset given, a tourism company named "Lets Travel" is looking forward to launching a new product i.e. Wellness Tourism Package to expand the customer base. There are currently 5 types of packages the company is offering - Basic, Standard, Deluxe, Super Deluxe, King. Prior to the launch, the company decided to harness the data of existing and potential customers to predict which of them are more likely to purchase the newly introduced travel package. The key aim of this study is to analyse customers' data to provide recommendations on which segment of customers should be targeted more, and to determine which package should be pitched to a customer based on their basic information (e.g. age and monthly income) by performing exploratory data analysis rigorously.

2 Methods

2.1 Data Preparation

We extract the dataset tour_package with 4888 observations and 20 variables *(**CustomerID***: Unique customer ID; ***ProdTaken***: Whether the customer has purchased a package or not (0: No, 1: Yes); ***Age***: Age of customer; ***TypeofContact***: How the customer was contacted (Company Invited or Self Inquiry); ***CityTier***: City tier depends on the development of a city, population, facilities, and living standards. The categories are ordered, i.e. Tier 1 > Tier 2 > Tier 3; ***DurationOfPitch***: Duration of the pitch by a salesperson to the customer; ***Occupation***: Occupation of customer; ***Gender***: Gender of customer; ***NumberOfPersonVisiting***: Total number of persons planning to take the trip with the customer; ***NumberOfFollowups***: Total number of follow-ups has been done by salesperson after the sales pitch; ***ProductPitched***: Product pitched by the salesperson; ***PreferredPropertyStar***: Preferred hotel property rating by customer; ***MaritalStatus***: Marital status of customer; ***NumberOfTrips***: Average number of trips in a year by customer; ***Passport***: The customer has a passport or not (0: No, 1: Yes); ***PitchSatisfactionScore***: Sales pitch satisfaction score; ***OwnCar***: Whether the customers own a car or not (0: No, 1: Yes); ***NumberOfChildrenVisiting***: Total number of children with age less than 5 planning to take the trip; ***Designation***: Designation of the customer in the current organisation; ***MonthlyIncome***: Gross monthly income of the customer).

We begin our data crunching by excluding the first variable, *CustomerID*, since they are impractical to our further analysis. We have also fixed the observations in the variable *Gender* by assigning "Fe Male" to "Female" to avoid improper analysis. Then, we remove the remaining observations that contain "NA". To deal with categorical variables *(ProdTaken, CityTier, TypeofContact*, *Occupation*, *Gender*, *ProductPitched*, *MaritalStatus*, *Passport, OwnCar, Designation)* in the dataset, we translate them to numerical format using the function as.factor() in R.

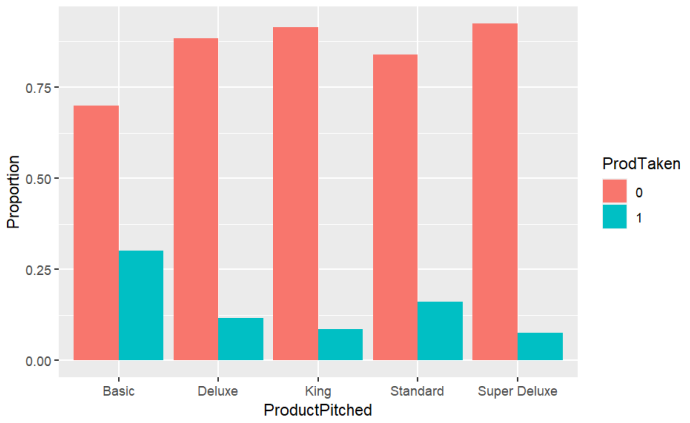
After all, we have 4128 observations and 19 variables in our dataset. There are 9 numerical variables and 10 categorical variables respectively, with *ProdTaken* being the outcome variable.

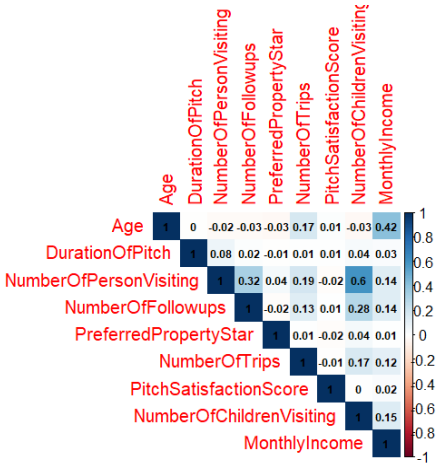
2.2 Preliminary Analysis

We first inspect the data by using the function summary(). We observe that there are 797 customers who took the tour packages.

```
> summary(tour)
 ProdTaken        Age                TypeofContact  CityTier DurationOfPitch           Occupation      Gender     NumberOfPersonVisiting
 0:3331    Min.   :18.00   Company Invited:1210   1:2678   Min.   :  5.00   Free Lancer    :   2   Female:1665   Min.   :1.000
 1: 797    1st Qu.:31.00   Self Enquiry   :2918   2: 162   1st Qu.:  9.00   Large Business: 381   Male  :2463   1st Qu.:2.000
           Median :36.00                          3:1288   Median : 14.00   Salaried       :1999                Median :3.000
           Mean   :37.23                                   Mean   : 15.58   Small Business:1746                 Mean   :2.949
           3rd Qu.:43.00                                   3rd Qu.: 20.00                                       3rd Qu.:3.000
           Max.   :61.00                                   Max.   :127.00                                       Max.   :5.000
 NumberOfFollowups      ProductPitched PreferredPropertyStar   MaritalStatus  NumberOfTrips    Passport PitchSatisfactionScore OwnCar
 Min.   :1.000     Basic       :1615   Min.   :3.000         Divorced : 789   Min.   : 1.000   0:2909   Min.   :1.000          0:1601
 1st Qu.:3.000     Deluxe      :1422   1st Qu.:3.000         Married  :1990   1st Qu.: 2.000   1:1219   1st Qu.:2.000          1:2527
 Median :4.000     King        : 104   Median :3.000         Single   : 667   Median : 3.000            Median :3.000
 Mean   :3.742     Standard    : 737   Mean   :3.578         Unmarried: 682   Mean   : 3.295            Mean   :3.061
 3rd Qu.:4.000     Super Deluxe: 250   3rd Qu.:4.000                          3rd Qu.: 4.000            3rd Qu.:4.000
 Max.   :6.000                         Max.   :5.000                          Max.   :22.000            Max.   :5.000
 NumberOfChildrenVisiting        Designation   MonthlyIncome
 Min.   :0.000            AVP          : 250   Min.   : 1000
 1st Qu.:1.000            Executive     :1615   1st Qu.:20751
 Median :1.000            Manager       :1422   Median :22418
 Mean   :1.224            Senior Manager: 737   Mean   :23178
 3rd Qu.:2.000            VP            : 104   3rd Qu.:25301
 Max.   :3.000                                  Max.   :98678
```

The proportion barplot of *ProdTaken* grouped by *ProductPitched* and the correlation matrix of all numerical variables are also plotted.





Looking at the barplot, we notice that Basic and Standard packages are taken by a relatively higher proportion of customers as compared to other “luxurious” packages (Deluxe, Super Deluxe and King).

Among the input numerical variables, there are a few interesting observations from the correlation matrix:
• *NumberOfPersonVisitings* is positively correlated to *NumberOfChildrenVisiting* (= 0.6) *and NumberOfFollowups* (= 0.32)
• *Age* and *MonthlyIncome* are moderately positively correlated (= 0.42)

2.3 Methodology: ML for Clustering

We used k-means clustering to determine which package should be pitched to a customer based on their age and monthly salary. We first selected the observations of customers that have taken the product. The range of *Age* is 15~65, and the range of *MonthlyIncome* is $15000~40000. We then calculated the mean and standard deviation of *Age* and *MonthlyIncome*, and standardised the values in columns *Age* and *MonthlyIncome*. Starting with the Basic tour packages, we determined the optimal number of clusters using the elbow method and we obtained *Figure 1*.

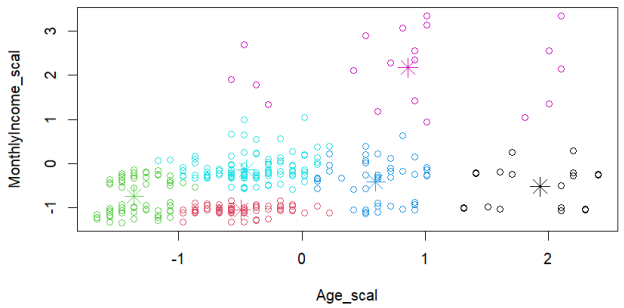


Figure 1. *Clustering for Basic Package (Standardised)*

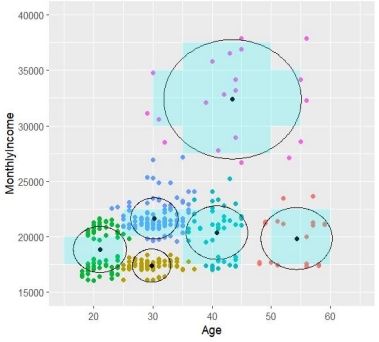


Figure 2. *Summarising Clusters in Basic into Subsections*

We computed the standardised mean and standard deviation (S.D.) of each cluster ($Standard\ Deviation = \sqrt{\frac{Withinss}{Size-1}}$), and scaled them back to the actual mean and standard deviation of the data. For each cluster, we added an ellipse whose centre is the actual mean of the cluster and $a = S.D.\ of\ Age,\ b = S.D.\ of\ MonthlyIncome$. We divided the range of actual *Age* and *MonthlyIncome* into 10 even sections respectively, which yielded 10×10=100 subsections for the whole graph. If at least 50% of the area of a subsection is encompassed by some ellipse, we then highlighted the

particular subsection in blue *(Figure 2)*. Hence, we concluded if the age and monthly income of a customer are in the range of some subsection highlighted in blue, then we should recommend the Basic package to them.

Repeating the procedures above for Standard, Deluxe, Super Deluxe, and King packages, we acquired the results:

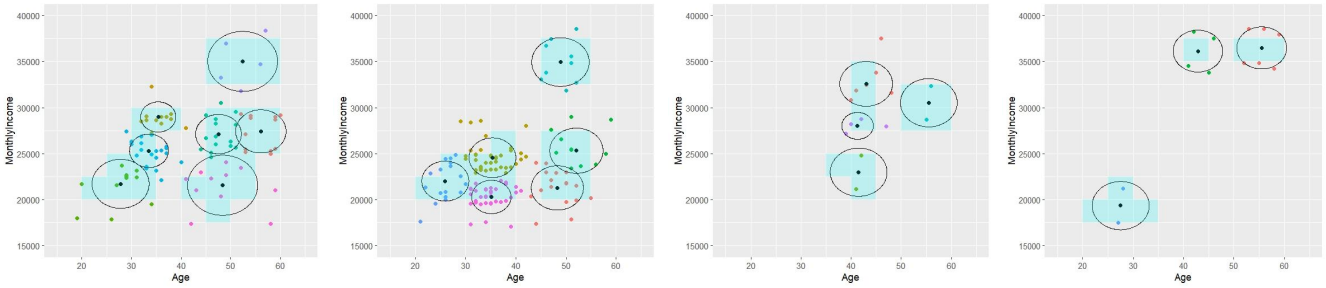


Figure 3. Clustering for Standard, Deluxe, Super Deluxe, and King Packages, Respectively

By combining all the graphs in Figure 3, we obtained the summary graph *(Figure 4*). E.g., if a customer is 42 years old and has a monthly income of $21000, we should recommend Basic, Standard and Super Deluxe packages to them.

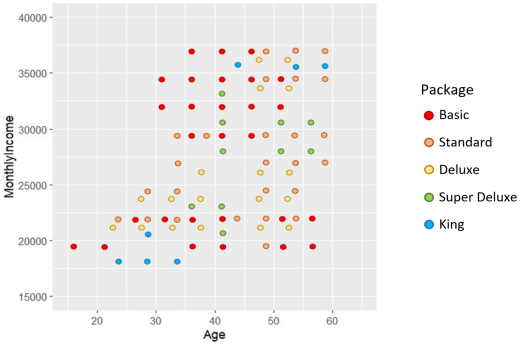


Figure 4. Summary Graph for Clustering

2.4 Methodology: ML for Classification

For classification, we seek to find the best classification model for this dataset. We performed our prediction using Logistic Regression, k-Nearest Neighbour(kNN) and Support Vector Machine(SVM) models and compared their performance. We adopted the 80/20 splitting rule, which selected 80% of data into the training set to build a predictive model, while the remaining 20% will be treated as a test set for evaluating the model.

We started with the logistic regression model and kNN model.

```
Call:
glm(formula = ProdTaken ~ ., family = "binomial", data = train.data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9655  -0.6147  -0.4114  -0.2060   3.1244

Coefficients:
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)              -3.143e+00  6.661e-01  -4.718 2.38e-06 ***
Age                      -2.665e-02  6.453e-03  -4.129 3.64e-05 ***
TypeofContact            -2.368e-01  1.074e-01  -2.205  0.02748 *
CityTier                  4.223e-01  5.488e-02   7.695 1.42e-14 ***
DurationOfPitch           3.259e-02  5.838e-03   5.582 2.38e-08 ***
Occupation               -2.091e-01  7.729e-02  -2.705  0.00683 **
Gender                    2.886e-01  1.028e-01   2.808  0.00499 **
NumberOfPersonVisiting   -1.332e-01  9.282e-02  -1.435  0.15137
NumberOfFollowups         3.652e-01  5.635e-02   6.481 9.09e-11 ***
ProductPitched           -1.826e-01  7.978e-02  -2.289  0.02208 *
PreferredPropertyStar     4.138e-01  6.028e-02   6.865 6.67e-12 ***
MaritalStatus             3.643e-01  5.145e-02   7.082 1.42e-12 ***
NumberOfTrips             8.583e-02  2.776e-02   3.092  0.00199 **
Passport                  1.610e+00  1.019e-01  15.798  < 2e-16 ***
PitchSatisfactionScore    1.023e-01  3.667e-02   2.789  0.00528 **
OwnCar                    4.848e-02  1.021e-01   0.475  0.63494
NumberOfChildrenVisiting -6.988e-02  7.489e-02  -0.933  0.35078
Designation              -2.261e-01  7.823e-02  -2.890  0.00385 **
MonthlyIncome            -5.879e-05  2.308e-05  -2.547  0.01087 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3244.5  on 3301  degrees of freedom
Residual deviance: 2608.6  on 3283  degrees of freedom
AIC: 2646.6

Number of Fisher Scoring iterations: 5

                ProdTaken_pred   0   1
                             0 650 108
                             1  18  50
```

Figure 5. *Summary and confusion matrix of Logistic Regression Model*

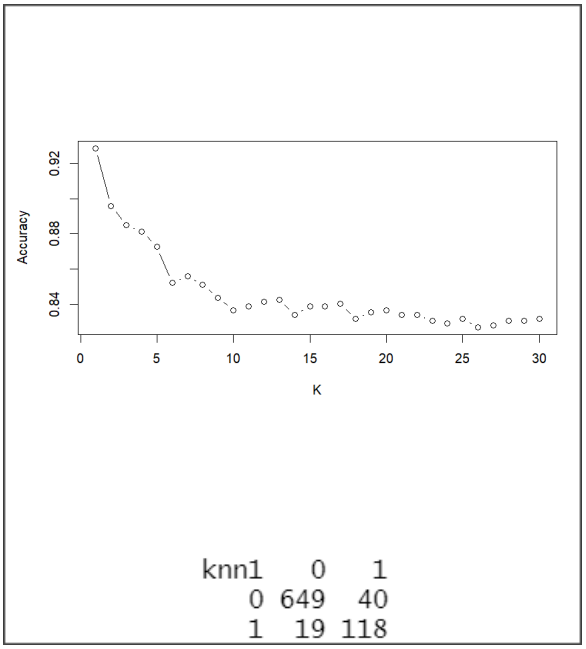


Figure 6. *Accuracy plot and Confusion Matrix for kNN model*

In Logistic Regression, we found that *Age, CityTier, DurationOfPitch, NumberOfFollowups, PreferredPropertyStar, MaritalStatus,* and *Passport* are significantly important variables that have an impact on *ProdTaken*. The accuracy of

this classification model is 84.75%. While in the kNN model, the plot above showed that k=1 results in the highest accuracy of 92.86%.

Then, we performed the classification using the SVM model. We evaluated the classification performance and checked its accuracy by calculating its confusion matrix.

```
Call:
svm(formula = ProdTaken ~ ., data = train.data, kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  1504

 ( 639 865 )


Number of Classes:  2

Levels:
 0 1
```

pred.svm	0	1
0	668	158
1	0	0

Figure 7. Summary of SVM Model with linear kernel and its Confusion Matrix

However, the accuracy for this SVM model is only 80.87%. Therefore, we tried to improve its performance using the radial kernel function.

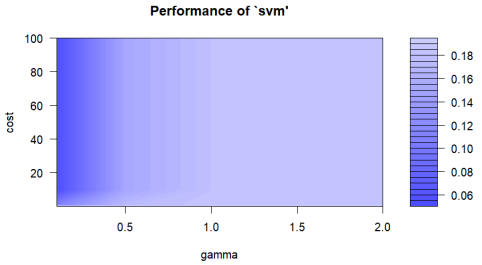
```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 gamma cost
   0.1   10

- best performance: 0.05421221

- Detailed performance results:
   gamma  cost      error  dispersion
1    0.1   0.1 0.19351186 0.031742737
2    0.5   0.1 0.19351186 0.031742737
3    1.0   0.1 0.19351186 0.031742737
4    2.0   0.1 0.19351186 0.031742737
5    0.1   1.0 0.11083677 0.020815765
6    0.5   1.0 0.17352467 0.029646539
7    1.0   1.0 0.18321615 0.029756912
8    2.0   1.0 0.18382129 0.030390376
9    0.1  10.0 0.05421221 0.010734612
10   0.5  10.0 0.15171931 0.028966934
11   1.0  10.0 0.18018768 0.029028646
12   2.0  10.0 0.18321615 0.029756912
13   0.1 100.0 0.05451341 0.008327203
14   0.5 100.0 0.15171931 0.028966934
15   1.0 100.0 0.18018768 0.029028646
16   2.0 100.0 0.18321615 0.029756912
```



pred.svm.tune	0	1
0	660	25
1	8	133

Figure 8. Summary of SVM Model with radial kernel and its Confusion Matrix

We noticed that the performance of this SVM classification model has been improved drastically, achieving an accuracy of 96.00% by applying the radial kernel function.

Comparing the logistic regression, kNN and SVM model, we observed that SVM model using radial function has the highest accuracy of 96.00%, followed by kNN model, logistic regression and SVM model using linear kernel function with accuracy 92.86%, 84.75% and 80.87% respectively. For the confusion matrices, the kNN model yields the most false positive, while the SVM model using the linear kernel function yields the most false negative.

3 Conclusion & Discussion

Using techniques of classification, we did achieve our objective in determining which variables are considerably significant in predicting which segment of customer is more likely to purchase the newly introduced travel package. We observed that *Age, CityTier, DurationOfPitch, NumberOfFollowups, PreferredPropertyStar, MaritalStatus, and Passport* contribute the most in the prediction. For this project, we found out that SVM model with radial kernel is the most suitable model for prediction with 96.00% accuracy, which achieves the highest accurary among the other models.

Through k-means clustering, we discovered the grouping of customers that have accepted a specific package type based on their age and monthly salary. With this, we can recommend a package to a new customer whose age and monthly salary are close to those of previous customers in some cluster. K-means clustering is a logical way to achieve our goal. However, by using the adjusted standard deviations in each cluster as the radius of the ellipse, some ellipses covered more area than the cluster implied (see Clustering for Super Deluxe and King Packages). Furthermore, in *Figure 2*, due to the peculiar position of the ellipse corresponding to the cluster in yellow, the ellipse does not encompass the area of any subsection for at least 50%, which results in the cluster not yielding a significant outcome in the findings. To improve on this issue, using smaller subsections may yield more accurate findings.

To conclude, the summary graph (*Figure 4*) from clustering provides a rough guide on which packages to recommend to a new customer based on their age and monthly income. Furthermore, we could predict if that particular customer will purchase the new tour package using the radial kernel SVM model once we are able to collect further customers' details.

4 Appendix

4.1 Plots and Graphs

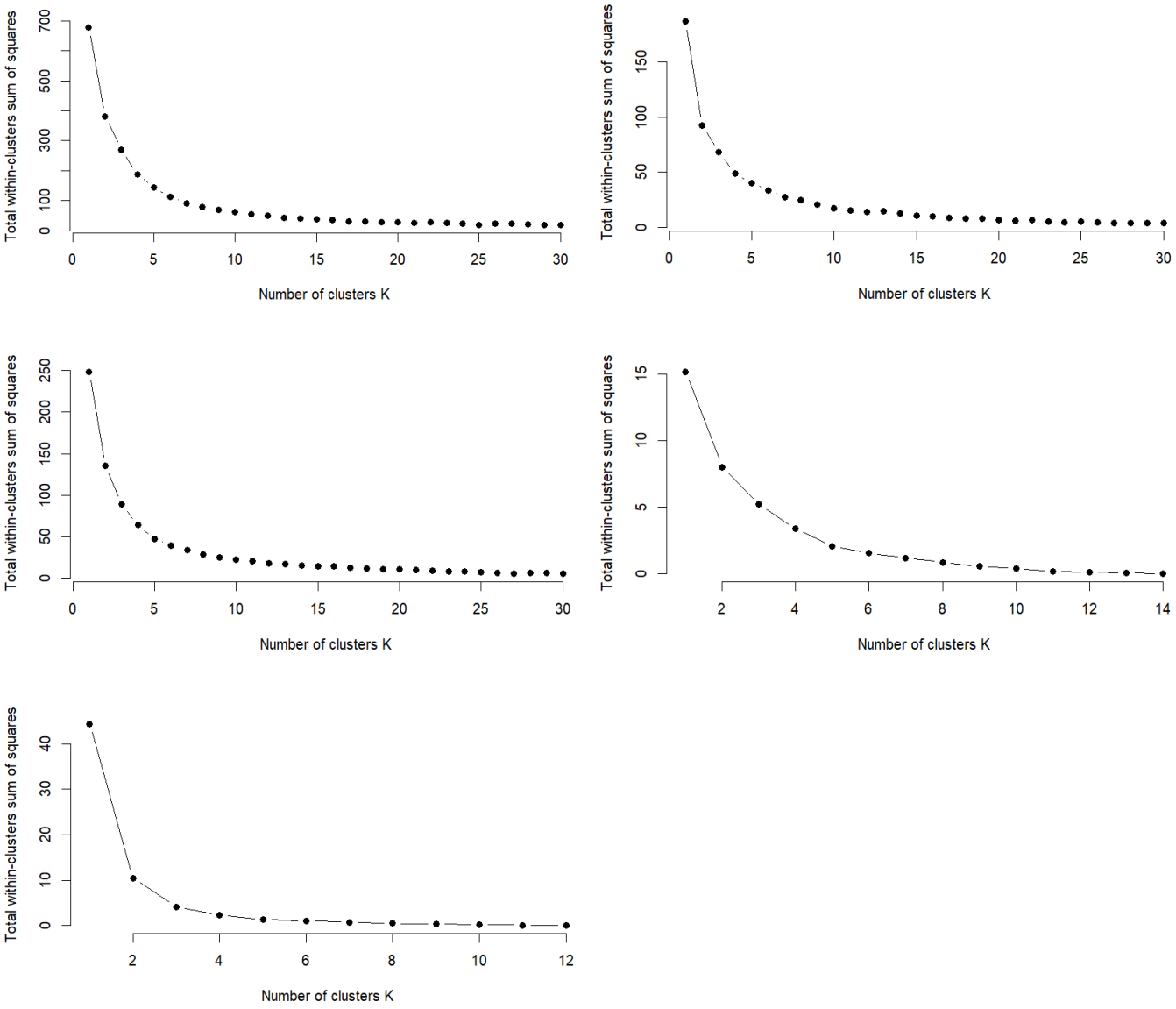


Figure 9.: Finding Optimal n for Clustering of Basic (n=6), Standard (n=7), Deluxe (n=6), Super Deluxe (n=4), King (n=3) Packages (Left to right, then top to bottom)

4.2 References

Goretti, M., Salinas, G., Nadeem, S., Dirk, D., Kaendera, S., Cevik, S., Babii, A., & Leigh, L. (2021). Tourism in the post-pandemic world: Economic challenges and opportunities for Asia-Pacific and the Western Hemisphere. Retrieved April 10, 2022, from https://www.imf.org/en/Publications/Departmental-Papers-Policy-Papers/Issues/2021/02/19/Tourism-in-the-Post-Pandemic-World-Economic-Challenges-and-Opportunities-for-Asia-Pacific-49915

Singapore Tourism Board. (2022, January 25). Singapore's tourism sector remained resilient in 2021, ready for recovery in 2022 and beyond. Retrieved April 10, 2022, from https://www.stb.gov.sg/content/stb/en/media-centre/media-releases/Singapores-tourism-sector-remained-resilient-in-2021-ready-for-recovery-in-2022-and-beyond.html

4.3 Codes

```
library(dplyr)
library(ggplot2)
library(corrplot)

setwd("C:/Users/DHLee/OneDrive/Desktop/Y2S2/PS0002 Intro To Data Science & Artificial Intelligence/Projects")
tour_raw <-read.csv("tour_package.csv", header = TRUE, sep=",")
str(tour_raw)

tour = tour_raw %>% select(-ï..CustomerID) %>% na.omit()
tour[tour=="Fe Male"] ="Female"

#change categorical variables to numeric values
tour$ProdTaken = as.factor(tour$ProdTaken)
tour$CityTier = as.factor(tour$CityTier)
tour$TypeofContact = as.factor(tour$TypeofContact)
tour$Occupation = as.factor(tour$Occupation)
tour$Gender = as.factor(tour$Gender)
tour$ProductPitched = as.factor(tour$ProductPitched)
tour$MaritalStatus = as.factor(tour$MaritalStatus)
tour$Passport = as.factor(tour$Passport)
tour$OwnCar = as.factor(tour$OwnCar)
tour$Designation = as.factor(tour$Designation)

dim(tour)# 4128 19
summary(tour)

attach(tour)

data = tour %>% group_by(ProductPitched,ProdTaken) %>% summarize(n=n()) %>% mutate(Proportion=n/sum(n))
ggplot(data, aes(x=ProductPitched, fill = ProdTaken, group = ProdTaken)) + geom_bar(aes(y=Proportion),
stat="identity", position = "dodge")

tour_num = tour %>% select(Age,DurationOfPitch,NumberOfPersonVisiting,NumberOfFollowups,
PreferredPropertyStar,NumberOfTrips,PitchSatisfactionScore,NumberOfChildrenVisiting,MonthlyIncome)
corrplot(cor(tour_num), type="upper", method="color",addCoef.col = "black",number.cex = 0.6)
```

Listing 1. Data Preparation and Preliminary Analysis

```
library(dplyr)
library(ggplot2)
library(ggforce)

tour_data = summarise(tour,mean(Age),sd(Age),mean(MonthlyIncome),sd(MonthlyIncome))
tour_scal <- tour %>% mutate(Age_scal = scale(Age),
  MonthlyIncome_scal = scale(MonthlyIncome)) %>% select(-c(Age,MonthlyIncome))
str(tour_scal)

#Basic
tour_basic = tour %>% filter(ProductPitched == "Basic") %>%
  select(-ProductPitched)
tour_scal_basic = tour_scal %>% filter(ProductPitched == "Basic") %>%
  select(-ProductPitched)

set.seed(100)
wcss_basic <- function(k) {
  kmeans(tour_scal_basic, k, nstart = 10 )$tot.withinss
}
k_basic.values <- 1:30
wcss_k_basic<-sapply(k_basic.values, wcss_basic)
plot(k_basic.values, wcss_k_basic,
     type="b", pch = 19, frame = FALSE,
     xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares")
set.seed(100)
k_basic <- kmeans(tour_scal_basic, 6, nstart = 25)
k_basic
plot(tour_scal_basic[,1:2], col=k_basic$cluster)
```

```
points(k_basic$centers[,1:2], col=1:6, pch=8, cex=2)

k_basic_df = cbind(data.frame(k_basic$centers),data.frame(k_basic$size),
  data.frame(k_basic$withinss)) %>% mutate("k_basic.SD"=
  sqrt(k_basic.withinss/(k_basic.size-1))) %>% mutate(
  "Actual_Age_Mean" = Age_scal*tour_data[1,2]+tour_data[1,1],
  "Actual_Age_SD" = k_basic.SD*tour_data[1,2],
  "Actual_Income_Mean" = MonthlyIncome_scal*tour_data[1,4]+tour_data[1,3],
  "Actual_Income_SD" = k_basic.SD*tour_data[1,4]) %>%
  select(-c(Age_scal,MonthlyIncome_scal,k_basic.size,k_basic.withinss,k_basic.SD))

ggplot()+
  geom_point(data=tour_basic,aes(Age,MonthlyIncome,
    color=as.character(k_basic$cluster))) +
  geom_point(data=k_basic_df[,c(1,3)],
    aes(x=Actual_Age_Mean,y=Actual_Income_Mean)) +
  geom_ellipse(aes(x0 = k_basic_df[1,1], y0 = k_basic_df[1,3],
    a = k_basic_df[1,2], b = k_basic_df[1,4], angle = 0)) +
  geom_ellipse(aes(x0 = k_basic_df[2,1], y0 = k_basic_df[2,3],
    a = k_basic_df[2,2], b = k_basic_df[2,4], angle = 0)) +
  geom_ellipse(aes(x0 = k_basic_df[3,1], y0 = k_basic_df[3,3],
    a = k_basic_df[3,2], b = k_basic_df[3,4], angle = 0)) +
  geom_ellipse(aes(x0 = k_basic_df[4,1], y0 = k_basic_df[4,3],
    a = k_basic_df[4,2], b = k_basic_df[4,4], angle = 0)) +
  geom_ellipse(aes(x0 = k_basic_df[5,1], y0 = k_basic_df[5,3],
    a = k_basic_df[5,2], b = k_basic_df[5,4], angle = 0)) +
  geom_ellipse(aes(x0 = k_basic_df[6,1], y0 = k_basic_df[6,3],
    a = k_basic_df[6,2], b = k_basic_df[6,4], angle = 0)) +
  geom_rect(aes(xmin=35, xmax=50, ymin=35000, ymax=37500),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=30, xmax=55, ymin=30000, ymax=35000),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=35, xmax=50, ymin=27500, ymax=30000),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=15, xmax=25, ymin=17500, ymax=20000),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=25, xmax=35, ymin=20000, ymax=22500),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=35, xmax=45, ymin=17500, ymax=22500),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=50, xmax=60, ymin=17500, ymax=22500),
    linetype=0, fill="cyan", alpha=0.2) +
  scale_x_continuous(limits = c(15, 65)) +
  scale_y_continuous(limits = c(15000, 40000))

#Standard
tour_standard = tour %>% filter(ProductPitched == "Standard") %>%
  select(-ProductPitched)
tour_scal_standard = tour_scal %>% filter(ProductPitched == "Standard") %>%
  select(-ProductPitched)

set.seed(100)
wcss_standard <- function(k) {
  kmeans(tour_scal_standard, k, nstart = 10 )$tot.withinss
}
k_standard.values <- 1:30
wcss_k_standard<-sapply(k_standard.values, wcss_standard)
plot(k_standard.values, wcss_k_standard,
     type="b", pch = 19, frame = FALSE,
     xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares")
set.seed(100)
k_standard <- kmeans(tour_scal_standard, 7, nstart = 25)
k_standard
plot(tour_scal_standard[,1:2], col=k_standard$cluster)
points(k_standard$centers[,1:2], col=1:7, pch=8, cex=2)

k_standard_df = cbind(data.frame(k_standard$centers),data.frame(k_standard$size),
  data.frame(k_standard$withinss)) %>% mutate("k_standard.SD"=
  sqrt(k_standard.withinss/(k_standard.size-1))) %>% mutate(
```

```
  "Actual_Age_Mean" = Age_scal*tour_data[1,2]+tour_data[1,1],
  "Actual_Age_SD" = k_standard.SD*tour_data[1,2],
  "Actual_Income_Mean" = MonthlyIncome_scal*tour_data[1,4]+tour_data[1,3],
  "Actual_Income_SD" = k_standard.SD*tour_data[1,4]) %>%
  select(-c(Age_scal,MonthlyIncome_scal,k_standard.size,k_standard.withinss,k_standard.SD))

ggplot()+
  geom_point(data=tour_standard,aes(Age,MonthlyIncome,
    color=as.character(k_standard$cluster))) +
  geom_point(data=k_standard_df[,c(1,3)],
    aes(x=Actual_Age_Mean,y=Actual_Income_Mean)) +
  geom_ellipse(aes(x0 = k_standard_df[1,1], y0 = k_standard_df[1,3],
    a = k_standard_df[1,2], b = k_standard_df[1,4], angle = 0)) +
  geom_ellipse(aes(x0 = k_standard_df[2,1], y0 = k_standard_df[2,3],
    a = k_standard_df[2,2], b = k_standard_df[2,4], angle = 0)) +
  geom_ellipse(aes(x0 = k_standard_df[3,1], y0 = k_standard_df[3,3],
    a = k_standard_df[3,2], b = k_standard_df[3,4], angle = 0)) +
  geom_ellipse(aes(x0 = k_standard_df[4,1], y0 = k_standard_df[4,3],
    a = k_standard_df[4,2], b = k_standard_df[4,4], angle = 0)) +
  geom_ellipse(aes(x0 = k_standard_df[5,1], y0 = k_standard_df[5,3],
    a = k_standard_df[5,2], b = k_standard_df[5,4], angle = 0)) +
  geom_ellipse(aes(x0 = k_standard_df[6,1], y0 = k_standard_df[6,3],
    a = k_standard_df[6,2], b = k_standard_df[6,4], angle = 0)) +
  geom_ellipse(aes(x0 = k_standard_df[7,1], y0 = k_standard_df[7,3],
    a = k_standard_df[7,2], b = k_standard_df[7,4], angle = 0)) +
  geom_rect(aes(xmin=45, xmax=60, ymin=32500, ymax=37500),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=45, xmax=60, ymin=25000, ymax=30000),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=45, xmax=55, ymin=20000, ymax=25000),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=40, xmax=45, ymin=20000, ymax=22500),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=45, xmax=50, ymin=17500, ymax=20000),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=35, xmax=40, ymin=27500, ymax=30000),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=30, xmax=35, ymin=20000, ymax=30000),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=25, xmax=30, ymin=20000, ymax=25000),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=20, xmax=25, ymin=20000, ymax=22500),
    linetype=0, fill="cyan", alpha=0.2) +
  scale_x_continuous(limits = c(15, 65)) +
  scale_y_continuous(limits = c(15000, 40000))

#Deluxe
tour_deluxe = tour %>% filter(ProductPitched == "Deluxe") %>%
  select(-ProductPitched)
tour_scal_deluxe = tour_scal %>% filter(ProductPitched == "Deluxe") %>%
  select(-ProductPitched)

set.seed(100)
wcss_deluxe <- function(k) {
  kmeans(tour_scal_deluxe, k, nstart = 10 )$tot.withinss
}
k_deluxe.values <- 1:30
wcss_k_deluxe<-sapply(k_deluxe.values, wcss_deluxe)
plot(k_deluxe.values, wcss_k_deluxe,
     type="b", pch = 19, frame = FALSE,
     xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares")
set.seed(100)
k_deluxe <- kmeans(tour_scal_deluxe, 6, nstart = 25)
k_deluxe
plot(tour_scal_deluxe[,1:2], col=k_deluxe$cluster)
points(k_deluxe$centers[,1:2], col=1:6, pch=8, cex=2)

k_deluxe_df = cbind(data.frame(k_deluxe$centers),data.frame(k_deluxe$size),
  data.frame(k_deluxe$withinss)) %>% mutate("k_deluxe.SD"=
```

```
  sqrt(k_deluxe.withinss/(k_deluxe.size-1))) %>% mutate(
  "Actual_Age_Mean" = Age_scal*tour_data[1,2]+tour_data[1,1],
  "Actual_Age_SD" = k_deluxe.SD*tour_data[1,2],
  "Actual_Income_Mean" = MonthlyIncome_scal*tour_data[1,4]+tour_data[1,3],
  "Actual_Income_SD" = k_deluxe.SD*tour_data[1,4]) %>%
  select(-c(Age_scal,MonthlyIncome_scal,k_deluxe.size,k_deluxe.withinss,k_deluxe.SD))

ggplot()+
  geom_point(data=tour_deluxe,aes(Age,MonthlyIncome,
    color=as.character(k_deluxe$cluster))) +
  geom_point(data=k_deluxe_df[,c(1,3)],
    aes(x=Actual_Age_Mean,y=Actual_Income_Mean)) +
  geom_ellipse(aes(x0 = k_deluxe_df[1,1], y0 = k_deluxe_df[1,3],
    a = k_deluxe_df[1,2], b = k_deluxe_df[1,4], angle = 0)) +
  geom_ellipse(aes(x0 = k_deluxe_df[2,1], y0 = k_deluxe_df[2,3],
    a = k_deluxe_df[2,2], b = k_deluxe_df[2,4], angle = 0)) +
  geom_ellipse(aes(x0 = k_deluxe_df[3,1], y0 = k_deluxe_df[3,3],
    a = k_deluxe_df[3,2], b = k_deluxe_df[3,4], angle = 0)) +
  geom_ellipse(aes(x0 = k_deluxe_df[4,1], y0 = k_deluxe_df[4,3],
    a = k_deluxe_df[4,2], b = k_deluxe_df[4,4], angle = 0)) +
  geom_ellipse(aes(x0 = k_deluxe_df[5,1], y0 = k_deluxe_df[5,3],
    a = k_deluxe_df[5,2], b = k_deluxe_df[5,4], angle = 0)) +
  geom_ellipse(aes(x0 = k_deluxe_df[6,1], y0 = k_deluxe_df[6,3],
    a = k_deluxe_df[6,2], b = k_deluxe_df[6,4], angle = 0)) +
  geom_rect(aes(xmin=45, xmax=55, ymin=32500, ymax=37500),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=45, xmax=55, ymin=20000, ymax=27500),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=20, xmax=40, ymin=20000, ymax=22500),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=25, xmax=40, ymin=22500, ymax=25000),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=35, xmax=40, ymin=25000, ymax=27500),
    linetype=0, fill="cyan", alpha=0.2) +
  scale_x_continuous(limits = c(15, 65)) +
  scale_y_continuous(limits = c(15000, 40000))

#Super Deluxe
tour_sdeluxe = tour %>% filter(ProductPitched == "Super Deluxe") %>%
  select(-ProductPitched)
tour_scal_sdeluxe = tour_scal %>% filter(ProductPitched == "Super Deluxe") %>%
  select(-ProductPitched)

set.seed(100)
wcss_sdeluxe <- function(k) {
  kmeans(tour_scal_sdeluxe, k, nstart = 10 )$tot.withinss
}
k_sdeluxe.values <- 1:14
wcss_k_sdeluxe<-sapply(k_sdeluxe.values, wcss_sdeluxe)
plot(k_sdeluxe.values, wcss_k_sdeluxe,
     type="b", pch = 19, frame = FALSE,
     xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares")
set.seed(100)
k_sdeluxe <- kmeans(tour_scal_sdeluxe, 4, nstart = 25)
k_sdeluxe
plot(tour_scal_sdeluxe[,1:2], col=k_sdeluxe$cluster)
points(k_sdeluxe$centers[,1:2], col=1:4, pch=8, cex=2)

k_sdeluxe_df = cbind(data.frame(k_sdeluxe$centers),data.frame(k_sdeluxe$size),
  data.frame(k_sdeluxe$withinss)) %>% mutate("k_sdeluxe.SD"=
  sqrt(k_sdeluxe.withinss/(k_sdeluxe.size-1))) %>% mutate(
  "Actual_Age_Mean" = Age_scal*tour_data[1,2]+tour_data[1,1],
  "Actual_Age_SD" = k_sdeluxe.SD*tour_data[1,2],
  "Actual_Income_Mean" = MonthlyIncome_scal*tour_data[1,4]+tour_data[1,3],
  "Actual_Income_SD" = k_sdeluxe.SD*tour_data[1,4]) %>%
  select(-c(Age_scal,MonthlyIncome_scal,k_sdeluxe.size,k_sdeluxe.withinss,k_sdeluxe.SD))

ggplot()+
  geom_point(data=tour_sdeluxe,aes(Age,MonthlyIncome,
```

```
    color=as.character(k_sdeluxe$cluster))) +
  geom_point(data=k_sdeluxe_df[,c(1,3)],
    aes(x=Actual_Age_Mean,y=Actual_Income_Mean)) +
  geom_ellipse(aes(x0 = k_sdeluxe_df[1,1], y0 = k_sdeluxe_df[1,3],
    a = k_sdeluxe_df[1,2], b = k_sdeluxe_df[1,4], angle = 0)) +
  geom_ellipse(aes(x0 = k_sdeluxe_df[2,1], y0 = k_sdeluxe_df[2,3],
    a = k_sdeluxe_df[2,2], b = k_sdeluxe_df[2,4], angle = 0)) +
  geom_ellipse(aes(x0 = k_sdeluxe_df[3,1], y0 = k_sdeluxe_df[3,3],
    a = k_sdeluxe_df[3,2], b = k_sdeluxe_df[3,4], angle = 0)) +
  geom_ellipse(aes(x0 = k_sdeluxe_df[4,1], y0 = k_sdeluxe_df[4,3],
    a = k_sdeluxe_df[4,2], b = k_sdeluxe_df[4,4], angle = 0)) +
  geom_rect(aes(xmin=40, xmax=45, ymin=27500, ymax=35000),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=50, xmax=60, ymin=27500, ymax=32500),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=35, xmax=45, ymin=22500, ymax=25000),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=40, xmax=45, ymin=20000, ymax=22500),
    linetype=0, fill="cyan", alpha=0.2) +
  scale_x_continuous(limits = c(15, 65)) +
  scale_y_continuous(limits = c(15000, 40000))

#King
tour_king = tour %>% filter(ProductPitched == "King") %>%
  select(-ProductPitched)
tour_scal_king = tour_scal %>% filter(ProductPitched == "King") %>%
  select(-ProductPitched)

set.seed(100)
wcss_king <- function(k) {
  kmeans(tour_scal_king, k, nstart = 10 )$tot.withinss
}
k_king.values <- 1:12
wcss_k_king<-sapply(k_king.values, wcss_king)
plot(k_king.values, wcss_k_king,
     type="b", pch = 19, frame = FALSE,
     xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares")
set.seed(100)
k_king <- kmeans(tour_scal_king, 3, nstart = 25)
k_king
plot(tour_scal_king[,1:2], col=k_king$cluster)
points(k_king$centers[,1:2], col=1:3, pch=8, cex=2)

k_king_df = cbind(data.frame(k_king$centers),data.frame(k_king$size),
  data.frame(k_king$withinss)) %>% mutate("k_king.SD"=
  sqrt(k_king.withinss/(k_king.size-1))) %>% mutate(
  "Actual_Age_Mean" = Age_scal*tour_data[1,2]+tour_data[1,1],
  "Actual_Age_SD" = k_king.SD*tour_data[1,2],
  "Actual_Income_Mean" = MonthlyIncome_scal*tour_data[1,4]+tour_data[1,3],
  "Actual_Income_SD" = k_king.SD*tour_data[1,4]) %>%
  select(-c(Age_scal,MonthlyIncome_scal,k_king.size,k_king.withinss,k_king.SD))

ggplot()+
  geom_point(data=tour_king,aes(Age,MonthlyIncome,
    color=as.character(k_king$cluster))) +
  geom_point(data=k_king_df[,c(1,3)],
    aes(x=Actual_Age_Mean,y=Actual_Income_Mean)) +
  geom_ellipse(aes(x0 = k_king_df[1,1], y0 = k_king_df[1,3],
    a = k_king_df[1,2], b = k_king_df[1,4], angle = 0)) +
  geom_ellipse(aes(x0 = k_king_df[2,1], y0 = k_king_df[2,3],
    a = k_king_df[2,2], b = k_king_df[2,4], angle = 0)) +
  geom_ellipse(aes(x0 = k_king_df[3,1], y0 = k_king_df[3,3],
    a = k_king_df[3,2], b = k_king_df[3,4], angle = 0)) +
  geom_rect(aes(xmin=40, xmax=45, ymin=35000, ymax=37500),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=50, xmax=60, ymin=35000, ymax=37500),
    linetype=0, fill="cyan", alpha=0.2) +
  geom_rect(aes(xmin=20, xmax=35, ymin=17500, ymax=20000),
    linetype=0, fill="cyan", alpha=0.2) +
```

```
  geom_rect(aes(xmin=25, xmax=30, ymin=20000, ymax=22500),
    linetype=0, fill="cyan", alpha=0.2) +
  scale_x_continuous(limits = c(15, 65)) +
  scale_y_continuous(limits = c(15000, 40000))
```

Listing 2. *ML for Clustering*

```
library(dplyr)
library(ggplot2)
library(class)
library(e1071)

tour[,
c("CityTier","TypeofContact","Occupation","Gender","ProductPitched","MaritalStatus","Passport","OwnCar","Designation
")] = sapply(tour[,
c("CityTier","TypeofContact","Occupation","Gender","ProductPitched","MaritalStatus","Passport","OwnCar","Designation
")], unclass)

##Logistic Regression Method

#Split data
set.seed(100)
training.idx = sample(1: nrow(tour), size=nrow(tour)*0.8)
train.data =tour[training.idx, ]
test.data = tour[-training.idx, ]

#Logistic regression
mlogit = glm(ProdTaken ~., data = train.data, family = "binomial")
summary(mlogit)

Pred.p =predict(mlogit, newdata =test.data, type = "response")

ProdTaken_pred_num =ifelse(Pred.p > 0.5, 1, 0)
ProdTaken_pred =factor(ProdTaken_pred_num, levels=c(0, 1))

#Accuracy
mean(ProdTaken_pred ==test.data$ProdTaken)

#Confusion matrix
tab=table(ProdTaken_pred,test.data$ProdTaken)
tab

##kNN Method

tour_kNN=tour

nor=function(x) { (x -min(x))/(max(x)-min(x)) }
tour_kNN[,2:19]=sapply(tour[,2:19], nor)

#Split data
set.seed(100)
training.idx = sample(1: nrow(tour_kNN), size=nrow(tour_kNN)*0.8)
train.data = tour_kNN[training.idx, ]
test.data = tour_kNN[-training.idx, ]

#Try different k to find the best classifier
ac=rep(0, 30)
for(i in 1:30){
  set.seed(101)
  knn.i=knn(train.data[,2:19], test.data[,2:19], cl=train.data$ProdTaken, k=i)
  ac[i]=mean(knn.i ==test.data$ProdTaken)
  cat("k=", i, " accuracy=", ac[i], "\n")
}

#Accuracy plot
plot(ac, type="b", xlab="K",ylab="Accuracy")
set.seed(101)
knn1=knn(train.data[,2:19], test.data[,2:19], cl=train.data$ProdTaken, k=1)
mean(knn1 ==test.data$ProdTaken)
table(knn1,test.data$ProdTaken)
```

```
##SVM Method

#Split data
set.seed(100)
training.idx = sample(1: nrow(tour), size=nrow(tour)*0.8)
train.data = tour[training.idx, ]
test.data = tour[-training.idx, ]

#SVM classification
m.svm=svm(ProdTaken~., data = train.data, kernel="linear")

summary(m.svm)

#Predict newdata in test set
pred.svm = predict(m.svm, newdata=test.data[,2:19])

#Evaluate classification performance and check accuracy
table(pred.svm, test.data$ProdTaken)
mean(pred.svm ==test.data$ProdTaken)

#Set a seed for reproducing results, improve model by using radial kernel
set.seed(123)
m.svm.tune=tune.svm(ProdTaken~., data=train.data, kernel="radial",cost=10^(-1:2), gamma=c(.1,.5,1,2))
summary(m.svm.tune)

#Visualize results of parameter tuning
plot(m.svm.tune)

#Confusion matrix and accuracy
best.svm = m.svm.tune$best.model
pred.svm.tune = predict(best.svm, newdata=test.data[,2:19])
table(pred.svm.tune, test.data$ProdTaken)

mean(pred.svm.tune ==test.data$ProdTaken)
```

Listing 3. *ML for Classification*