STAT 517 – STATISTICAL PREDICTIVE MODELLING

FINAL PROJECT – SPEED DATING DATASET

NAME: VIGNESH J MURALIDHARAN

Contents

1. Introduction:

The data is gathered from participants in experimental speed dating events of the years 2002-2004. All the attendees would have to talk with at least 20 opposite sex and were asked if they would like to see their date again. The questionnaire of the data gathered from participants were similarly asked with both the partners each time when they meet so that to get an idea of the population. During the event attendees would have 4 min with every other participant and they were also asked to rate their date on six attributes like Attractiveness, Sincerity, Intelligence, Fun, Ambition and Shared Interest. The dataset also includes fields like demographics, dating habits, self-perception across key attributes, beliefs on what others find valuable in the partner and lifestyle with preference of yours on the attributes, how do you rate the partner? How does partner gives importance on the attributes and how do you rate yourself on the attributes. At the end of the event the decision on both sides were asked and based on the decision the match or not match has been decided.

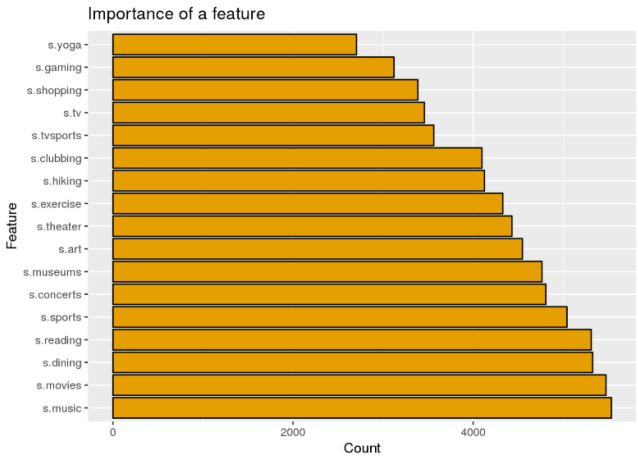
2. Aim:

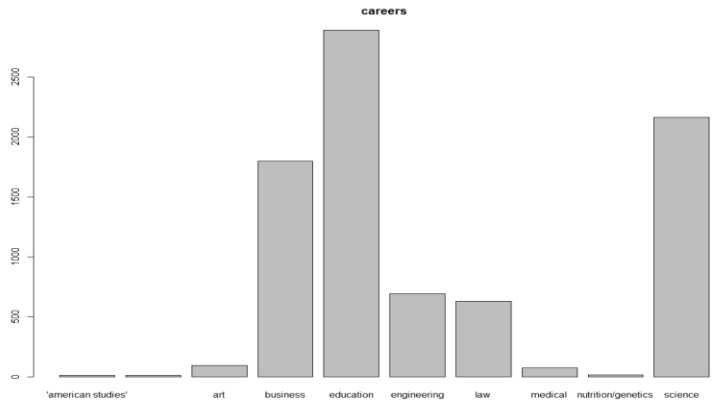
To develop approaches for predicting whether a partner on the dating event is a ("Match" – 1) or ("Not Match" – 0) based on the decisions taken in the event.

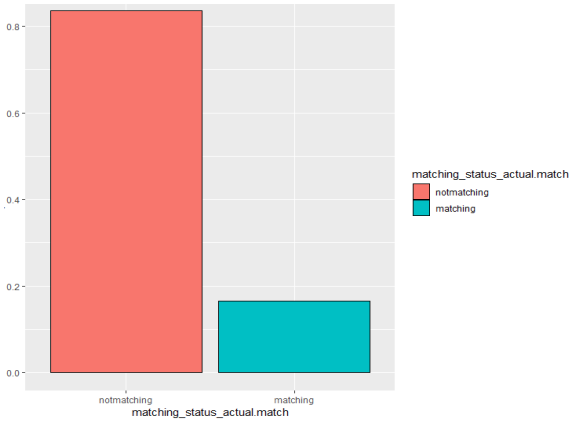
3. Data Description:

Dimension of the actual dataset: 8378 (Rows) & 123 (Features)
List of Textual Variables: Gender, Race, and Field of Study – Total: 3
List of Continuous variables except for Target: Total 122 (Listed already in the Final Draft)
Target: Variable Name – Match => Type binary (0 – Not Match, 1 - Match)

3.1 Data Visualization:







The graphs above mentions the feature importance that can be seen given by people to the attributes were shows that almost 5000 people voted importance for music, movies, dancing and reading. And the least importance was given to yoga, gaming and shopping. Also, people at the age of 27 population were seen the most in the event with careers having in education, science and business. The target response nonmatching was 83% and matching was around 20%

4. Methodology – Preprocessing: Used both R & Python

1. Starting with in "R", the dataset was having both "?" and the "NA" is almost 62 variables. I first revalued all the "?" to the "NA" in the complete dataset. Then I used imputation technique to replace "NA" with the highest number of repeated levels. The image below shows how we can check levels & impute for each variable. I did it for 62 variables but not listed here.

```
578  table(data$age) # Checking for the maxi levels in the data for variable age
579  data$age=revalue(data$age,c("NA"="27")) #Imputing the "?" of the age variable to 27 which is listed higher
```

2. The variable "Field of Study" had total of 262 levels and for easy analysis later for association the levels had been decreased to 10 Major fields namely Art, Education, Business, Engineering, Medical, Law, Science. I did this in R by basically mean aggregating by using the codes shown below

```
84  data$field=recode(data$field,"'Economics; English'"="Education")
85  data$field=recode(data$field,"'Economics; Sociology'"="Education")
86  data$field=recode(data$field,"'Ed.D. in higher education policy at TC'"="Education")
```

3. Next there were some factor levels in the data survey based on the survey taken from other side of the person these variables which has these kind of numbers were changed according to the mostly available levels and imputed them to it. The image below shows how I revalued them according to the levels of factor available.

```
309  #data$d_importance_same_race - Renaming levels of a factor
310  #str(data$d_importance_same_race)
311  data$d_importance_same_race=revalue(data$d_importance_same_race,c("[0-1]"="1"))
312  data$d_importance_same_race=revalue(data$d_importance_same_race,c("[2-5]"="5"))
313  data$d_importance_same_race=revalue(data$d_importance_same_race,c("[6-10]"="8"))
```

   After the data has been revalued and unwanted variable was dropped the new data from "R" was exported into CSV format to do preprocessing in python. The dimension of data after imputation and dropping is 8378 (R) & 118 (C).

4. In "Python" I checked for the data types first so that I can change the Target variable to category type and assigning and adding encoders to new columns. Here I will try to have object type "uint8", "int64", "int8" & "float64" data types in the dataset for the analysis of the supervised learning. After I select the variables the dimension of the dataset is 8378 (R) & 117 (C) for "X" variables and is 8378 (R) & 1 (C) for "Target" variable

```
]: data["match"]=data["match"].astype('category') # Changing "Y"-Salary from Object type to Category type
   data["match_category"]=data["match"].cat.codes # Assigned encoders adding new columns
   data=data.select_dtypes(include=['uint8','int64','int8','float64']).copy() # Here i drop the object type to add int and uint type
```

Now the dataset is ready for the analysis.

5. Supervised Learning: Used Python

The target variable ("Y") in the dataset is either Match – 0 or Not Match – 1 this means that classification will work much better. Regression will not work in this target because the variable is not continuous. For classification the Y and the X variable has been assigned and the Train-Test split has been done with 75% & 25% split. The split and the dimension is shown in the image below. Later the accuracy is tested with the models listed KNearest Neighbors, Naïve Bayes, Logistic Regression, Decision Tree, Random Forest, Gradient Boosting Classifier, SVM and Neural Networks. The Evaluation of the models were made to see which performed better and the performance of the models were analyzed with Accuracy score and the ROC Curve.

```
y_data = pd.DataFrame([data.match_category]).T # TAKING THE SALARY AS THE Y VARIABLE OR TARGET
x_data = pd.DataFrame(data) # CONSIDERING ALL THE VARIABLES TO AS X VARIABLES AT FIRST
x_data = data.drop(['match_category'], axis = 1) # DROPPING THE SALARY FROM THE X VARIABLES
print (x_data.shape) ; print (y_data.shape) # PRINTING THE DIMENTION OF THE BOTH LEFTOVER "Y" & "X" VARIABLES
from sklearn.cross_validation import train_test_split # TRAINING AND TEST SPLIT CROSS VALIDATAION
xtrain_data, xtest_data, ytrain_data, ytest_data = train_test_split(
    x_data, y_data, random_state = 1, test_size = 0.25) # THE TRAIN AND TEST SPLIT IS 75% AND 25 %
print (xtrain_data.shape) ; print (xtest_data.shape) # PRINTING ALL THE X_TRAIN, X_TEST
print (ytrain_data.shape) ; print (ytest_data.shape) # PRINTING ALL THE Y_TRAIN, Y_TEST
```

```
(8378, 117)
(8378, 1)
(6283, 117)
(2095, 117)
(6283, 1)
(2095, 1)
```

<= dim(X_train)
<= dim(X_test)
<=dim(Y_train)
<=dim(Y_test)

5.1 K Nearest Neighbors:

K Nearest Neighbors (KNN) is a radically different approach. To make a prediction for X = x KNN uses training observations and identifies those closes to x. X is then attributed to the class that is the most prominent amongst the neighbors. It is expected for KNN to perform well in situations where the decision boundary is highly non-linear. The major determining factor for the prediction in KNN is the number of nearest neighbors of a similar class.

```
#Evaluating the model on the test data set
ymodel_KNN = model.predict(xtest_data)
acc_knn1 = round(model.score(xtrain_data, ytrain_data) * 100, 2)
print ('Training accuracy = {}'.format(acc_knn1))
acc_knn = round(accuracy_score(ytest_data, ymodel_KNN) * 100, 2)
print ('Testing accuracy = {}'.format(acc_knn))
```

```
Training accuracy = 86.15
```

```
In [20]: # KNeighborsClassifier is a class --> go to the source
         # http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.ht
         for k in [1, 3, 5, 7, 9,11,15,17,19,21]:
             knn = KNeighborsClassifier(n_neighbors = k)
             knn.fit(xtrain_data, ytrain_data) #construct the BallTree or KDTree
             y_test_pred = knn.predict(xtest_data)
             cfm = confusion_matrix(ytest_data, y_test_pred)
             #print('KNN confusion matrix for test set\n', cfm)
             mean_acc = knn.score(xtest_data, ytest_data)
             print('mean accuracy of {} nearest neighbor(s) is {} '.format(k, round(mean_acc, 4)))
```

```
mean accuracy of 1 nearest neighbor(s) is 0.7599
mean accuracy of 3 nearest neighbor(s) is 0.7952
mean accuracy of 5 nearest neighbor(s) is 0.8095
mean accuracy of 7 nearest neighbor(s) is 0.8215
mean accuracy of 9 nearest neighbor(s) is 0.8239
mean accuracy of 11 nearest neighbor(s) is 0.8258
mean accuracy of 15 nearest neighbor(s) is 0.8329
mean accuracy of 17 nearest neighbor(s) is 0.8339
mean accuracy of 19 nearest neighbor(s) is 0.8315
mean accuracy of 21 nearest neighbor(s) is 0.831
```

Testing accuracy for 17 neighbors is highest testing scores increases as we increase the k neighbors

```
In [21]: #get area under roc curve from model used above
         roc_auc_score(ytest_data, ymodel_KNN)
```

```
Out[21]: 0.5269482107280825
```

5.2 Naïve Bayes:

The classifier assumes that the joint probability density function is the product of the marginal probability density function for each xi. This presumes that the probability density function of each class is independent from each other.

$$Posterior = ((Likelihood) * (Proposition prior probability)) / Evidence prior probability$$

$$P(xj|Ci) = 1/(\sqrt{2 * pi * sigma^2 * ci}) * exp(-(xj - \mu * cj)^2 / 2\sigma^2 * ci)$$

```
model = GaussianNB()
model.fit(xtrain_data, ytrain_data)
ymodel_NB = model.predict(xtest_data)
acc_gauss1 = round(model.score(xtrain_data, ytrain_data) * 100, 2)
print ('Training accuracy = {}'.format(acc_gauss1))
acc_gauss = round(accuracy_score(ytest_data, ymodel_NB) * 100, 2)
print ('Testing accuracy = {}'.format(acc_gauss))
#get area under roc curve from model used above
roc_auc_score(ytest_data, ymodel_NB)
```

```
Training accuracy = 97.98
Testing accuracy = 98.14
```

Naïve Bayes performed better with this dataset and also ROC score is good

```
0.9441260744985673
```

5.3 Logistic Regression:

Logistic Regression Assumptions (1) Binary logistic regression requires the dependent variable to be binary (2) For a binary regression, the factor level 1 of the dependent variable should represent the desired outcome.

$$p(X) = e^{\beta 0 + \beta 1X} / 1 + e^{\beta 0 + \beta 1X}$$

```
logreg = LogisticRegression().fit(xtrain_data, ytrain_data)
y_model_log = logreg.predict(xtest_data)
print("Training set score: {:.3f}".format(logreg.score(xtrain_data, ytrain_data)))
print("Test set score: {:.3f}".format(logreg.score(xtest_data, ytest_data)))
#get area under roc curve from model used above
roc_auc_score(ytest_data, y_model_log)
```

```
Training set score: 1.000
Test set score: 1.000
```

LR performed outstanding herewith 100% in both train and test also ROC performs good

```
1.0
```

But when I did the 3 fold cross validation for the logistic regression, the train and test score went little down

```
rs = 0
logitR = LogisticRegression(random_state = rs)

param_grid = {
            'penalty' : ['l2','l1'],
            'C' : [0.001, 0.01, 0.1, 1, 10, 100, 1000]}
clf = GridSearchCV(estimator = logitR, cv = 3, param_grid = param_grid , scoring = 'accuracy', verbose = 1)
clf.fit(xtrain_data, ytrain_data)
```

3-fold CV

```
Fitting 3 folds for each of 14 candidates, totalling 42 fits
```

```
# 2. Parameters and methods in GridSearchCV class
# 2.1 parameters
clf.cv_results_  # output the results (average and all folds)
print('mean test scores: ', clf.cv_results_['mean_test_score']) # average test score on all folds
print('best score: ', clf.best_score_)
print('best  parameters: ', clf.best_params_)
print('best  parameters: ', clf.scorer_)
# 2.2 methods of class GridSearchCV
# fit(X,y) # train models for all combinations of parameters using k-fold cross-validation
# predict(X,y) # use the model with the best found parameter
```

```
def show_metrics(cm):
    '''This function calculates the accuracy, precision, recall ratio and F1 score using the
    confusion matrix of a two-class problems.
    Malignant tumors and benign tumors are negative

    :param cm: 2*2 confision matrix
    '''

    tp = cm[1,1] #true postive (true label)1-->1(prediction)
    fn = cm[1,0] #false negative 1-->0
    fp = cm[0,1] #false positive 0-->1
    tn = cm[0,0] #true negative  0-->0

    precision = tp/(tp+fp)
    recallR = tp/(tp+fn)
    acc = (tp+tn)/(tp+tn+fp+fn)
    print('Accuracy  =    {:.3f}'.format(acc))
    print('Precision =    {:.3f}'.format(precision))
    print('Recall ratio = {:.3f}'.format(recallR))
    print('F1_score =    {:.3f}'.format(2*(recallR * precision)/(recallR + precision)))
```

Explanation of the Confusion matrix

```
best_ps =clf.best_params_
best_clf = LogisticRegression(C = best_ps['C'],
                              penalty = best_ps['penalty'],
                              random_state = rs)
best_clf.fit(xtrain_data, ytrain_data)
y_test_pred = best_clf.predict(xtest_data)
cm = confusion_matrix(ytest_data, y_test_pred)
print(cm)
show_metrics(cm)
```

```
[[1746    0]
 [   0  349]]
Accuracy  =    1.000
Precision =    1.000
Recall ratio = 1.000
F1_score =    1.000
```

The confusion matrix with all other scores were 100% on the test data

```
logreg001 = LogisticRegression(C=0.01).fit(xtrain_data, ytrain_data)
y_model_log_001 = logreg.predict(xtest_data)
print("Training set score: {:.3f}".format(logreg001.score(xtrain_data, ytrain_data)))
print("Test set score: {:.3f}".format(logreg001.score(xtest_data, ytest_data)))
#get area under roc curve from model used above
roc_auc_score(ytest_data, y_model_log_001)
```

```
Training set score: 0.963
Test set score: 0.958
```

The decrement in the train test score is noted after 3-fold cross validation

```
1.0
```

5.4 Decision Tree:

```
tree = DecisionTreeClassifier(random_state=0)
tree.fit(xtrain_data, ytrain_data)
y_model_dectree = tree.predict(xtest_data)
print("Accuracy on training set: {:.3f}".format(tree.score(xtrain_data, ytrain_data)))
print("Accuracy on test set: {:.3f}".format(tree.score(xtest_data, ytest_data)))
#get area under roc curve from model used above
roc_auc_score(ytest_data, y_model_dectree)
```

```
Accuracy on training set: 1.000
Accuracy on test set: 1.000
```

The decision tree did very good for all train, test and roc curve

```
1.0
```

5.5 Random Forest:

" nestimators " This is the number of trees we want to build before taking the maximum voting or averages of predictions. Higher number of trees gives us better performance but makes our code slower. We should choose as high value as your processor can handle because this makes our predictions stronger and more stable.

```
forest = RandomForestClassifier(n_estimators=100, random_state=0)
forest.fit(xtrain_data, ytrain_data)
y_model_randomfor = forest.predict(xtest_data)
print("Accuracy on training set: {:.3f}".format(forest.score(xtrain_data, ytrain_data)))
print("Accuracy on test set: {:.3f}".format(forest.score(xtest_data, ytest_data)))
roc_auc_score(ytest_data, y_model_randomfor)
```

```
Accuracy on training set: 1.000
Accuracy on test set: 0.994
```

The test shows 99.4% which is very good with also good roc curve

```
0.9828080229226361
```

5.6 Support Vector Machine (SVM):

Is an extension of the support vector classifier that results from enlarging the feature space in a specific way using kernels. The linear support vector classifier can be represented as

$$f(X) = (\beta)0 + \Sigma\alpha(x, xi)$$ Where n parameters αi i=1,2,3...n are per training observations.

```
svc = SVC(C=1000)
svc.fit(xtrain_data, ytrain_data)
print("Accuracy on training set: {:.2f}".format(svc.score(xtrain_data, ytrain_data)))
print("Accuracy on test set: {:.2f}".format(svc.score(xtest_data, ytest_data)))
```

```
Accuracy on training set: 1.00
Accuracy on test set: 0.83
```

The test performed good with 83%

```
y_test_pred = clf.predict(xtest_data)
cm = confusion_matrix(ytest_data, y_test_pred)
print('Confusion matrix:\n', cm)
show_metrics(cm)
roc_auc_score(ytest_data,y_test_pred)
```

```
Confusion matrix:
 [[1746    0]
 [   0  349]]
Accuracy  =    1.000
Precision =    1.000
Recall ratio = 1.000
F1_score =    1.000
```

The confusion matrix and other scores performed like the logistic regression on the test data

```
1.0
```

5.7 Neural Networks (NN):

```
mlp = MLPClassifier(random_state=42)
mlp.fit(xtrain_data, ytrain_data)
y_model_NN = mlp.predict(xtest_data)
print("Accuracy on training set: {:.2f}".format(mlp.score(xtrain_data, ytrain_data)))
print("Accuracy on test set: {:.2f}".format(mlp.score(xtest_data, ytest_data)))
roc_auc_score(ytest_data, y_model_NN)
```

```
Accuracy on training set: 0.82
Accuracy on test set: 0.81
```

The Neural Network performed better with train and test score but still little lower than the other models

```
0.8172482661966609
```

5.8 Gradient Boosting Classifier:

```
gbrt = GradientBoostingClassifier(random_state=0)
gbrt.fit(xtrain_data, ytrain_data)
y_model = gbrt.predict(xtest_data)
print("Accuracy on training set: {:.3f}".format(gbrt.score(xtrain_data, ytrain_data)))
print("Accuracy on test set: {:.3f}".format(gbrt.score(xtest_data, ytest_data)))
#get area under roc curve from model used above
roc_auc_score(ytest_data, y_model)
```

```
Accuracy on training set: 1.000
Accuracy on test set: 1.000
```

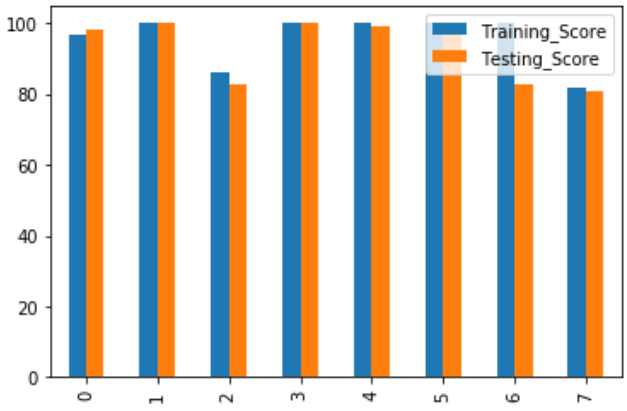
The Gradient boosting did exceptionally well for both train and test

```
1.0
```

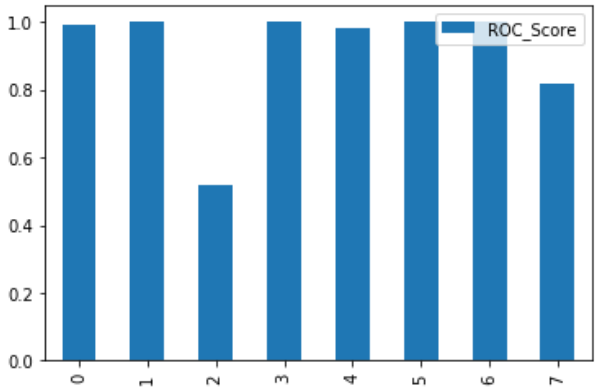
5.9 Summary: Classification

Overall, the classification performed well with Logistic, Decision Tree and Gradient Boosting Classifier performed better than other models. KNN and Neural Networks performed little bit behind. The reason why most of the predictions were attained well was may be because the not match was 83% response and other 27% was only match which was like the predicting almost same with all the classification models.

	NameModel	Training_Score	Testing_Score
0	Gaussian Naive Bayes	97	98
1	Logistic Regression	100	100
2	KNN	86	83
3	Decision Tree	100	100
4	Random Forest	100	99
5	Gradient Boosting Classifier	100	100
6	Support Vector Machine	100	83
7	Neural Networks	82	81



	NameModel	ROC_Score
0	Gaussian Naive Bayes	0.994
1	Logistic Regression	1.000
2	KNN	0.520
3	Decision Tree	1.000
4	Random Forest	0.982
5	Gradient Boosting Classifier	1.000
6	Support Vector Machine	1.000
7	Neural Networks	0.817



6. Association Rules: Used R Program

General facts known: The association rules were conducted to see the relationship among the variables. To understand better with the rules and what the support confidence and lift mean in the table here are few facts. Considering the part of the LHS first rule as “A” another part of the rule is “B”.

Support: The first measure called the support is the number of transactions that include items in the {A} & {B} parts of the rule as a percentage of the total number of transactions. It is a measure of how frequently the collection of items occur together as a percentage of all transactions.

$$Support = \frac{(A + B)}{Total\ Transactions}$$

Confidence: The second measure is the ratio of the number of transactions that include all items in {B} as well as the number of transactions that include all items in {A} to the number of the transactions that include all items in {A}.

$$Confidence = \frac{(A + B)}{A}$$

Lift: Lift ratio is the ratio of confidence to expected confidence. Expected confidence is the confidence divided by the frequency of B. the lift tells us how much better a rule is at predicting the result than just assuming the result in the first place. Greater the lift values indicate stronger associations. In common if the lifts is equal to 1, it means that A and B are independent. If the lift is higher than 1, it means that A & B are positively correlated. If the lift is lower than 1, it means that A & B are negatively correlated.

$$Lift = \frac{(\frac{A + B}{A})}{\frac{B}{Total}}$$

For this analysis I removed the decision for both the partners so that we will know much better what the attributes with levels are makes the actual match and not match in the survey. The relationship was analyzed in different manner to check for different attributes in connection with the target variable and what makes the people think why the partners match.

6.1 Association among the actual full dataset (In General):

The transactions for association rules were first generated by installing the library's arules, arulesViz and arulesCBA. After I apply the data as a factor the transactions were analyzed with listing the item frequency more than 0.5. The total transactions were listed based on the support 0.001 and confidence of 0.80. The LHS is assigned as the all X variables and RHS is assigned as the Y variable (Match =1). I inspected first 20 top rules in this.

```
> data2.rules1 <- apriori(data = data2.arules , parameter = list( supp = 0.001 , conf = 0.80) ,
 appearance = list(default = "lhs" , rhs = "match=1") )
Apriori

Parameter specification:

Algorithmic control:

Absolute minimum support count: 8

set item appearances ...[1 item(s)] done [0.00s].
set transactions ...[1677 item(s), 8378 transaction(s)] done [0.33s].
sorting and recoding items ... [1583 item(s)] done [0.03s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3Mining stopped (time limit reached). Only patterns up to a length
 of 3 returned! done [6.88s].
writing ... [86 rule(s)] done [0.56s].
creating S4 object  ... done [0.18s].
```

The count 8 with Total Transactions 8378 and 1677 items. Total 86 rules

```
     lhs                                          rhs       support     confidence lift     count
[1]  {shared_interests_o=10,funny_partner=9}  => {match=1} 0.002267844 0.9047619  5.492823 19
[2]  {funny_o=9,shared_interests_partner=10}  => {match=1} 0.002267844 0.9047619  5.492823 19
[3]  {pref_o_ambitious=3,d_funny_partner=9}   => {match=1} 0.001074242 0.9000000  5.463913  9
[4]  {d_funny_o=9,ambtition_important=3}      => {match=1} 0.001074242 0.9000000  5.463913  9
[5]  {sincere_important=0,theater=1}          => {match=1} 0.001074242 0.9000000  5.463913  9
[6]  {attractive=10,theater=1}                => {match=1} 0.001074242 0.9000000  5.463913  9
[7]  {tvsports=10,theater=1}                  => {match=1} 0.001074242 0.9000000  5.463913  9
[8]  {exercise=10,theater=1}                  => {match=1} 0.001074242 0.9000000  5.463913  9
[9]  {gaming=7,theater=1}                     => {match=1} 0.001074242 0.9000000  5.463913  9
[10] {intelligence=10,theater=1}              => {match=1} 0.001074242 0.9000000  5.463913  9
[11] {clubbing=9,theater=1}                   => {match=1} 0.001074242 0.9000000  5.463913  9
[12] {d_attractive=9,theater=1}               => {match=1} 0.001074242 0.9000000  5.463913  9
[13] {art=5,theater=1}                        => {match=1} 0.001074242 0.9000000  5.463913  9
[14] {theater=1,d_clubbing=9}                 => {match=1} 0.001074242 0.9000000  5.463913  9
[15] {funny_important=10,theater=1}           => {match=1} 0.001074242 0.9000000  5.463913  9
[16] {theater=1,shopping=7}                   => {match=1} 0.001074242 0.9000000  5.463913  9
[17] {dining=10,theater=1}                    => {match=1} 0.001074242 0.9000000  5.463913  9
[18] {theater=1,concerts=7}                   => {match=1} 0.001074242 0.9000000  5.463913  9
[19] {theater=1,d_gaming=7}                   => {match=1} 0.001074242 0.9000000  5.463913  9
[20] {sincere_important=0,gaming=7}           => {match=1} 0.001074242 0.9000000  5.463913  9
```

Here for confidence >0.8 the first confidence rule listed is 0.907 with 19 counts.

In the above association result “Shared interest of partner with 10 score” and “Partner who are funny” tops the rules with highest confidence and lift score. Here,

$$Confidence\ (rule1) = \frac{Shared\ interest + Funny\ partner}{Total\ \#\ of\ transactions\ that\ include\ all\ Shared\ interest} = 0.9047619$$

$$Lift = \frac{\frac{Confidence(rule1)}{Funny\ partner}}{Total\ frequency\ of\ Funny\ partner} = 5.49823$$

Thus, with the lift higher than 1it means that all the inspected rules in these transactions were positively correlated. Which makes sense that people who have shared interest and funny partner or people who are sincere or who like theater, tv sports, exercise clubbing all these people are getting their match during the event.

Comparing this match with the not matching group and the variables associated with it would be interesting and thus I inspected the rules with (Not match=0) group and inspecting with all LHS as same as earlier analysis.

```
> data2.rules1 <- apriori(data = data2.arules , parameter = list( supp = 0.001 , conf = 0.80) ,
appearance = list(default = "lhs" , rhs = "match=0") )
Apriori

Parameter specification:

Algorithmic control:

Absolute minimum support count: 8

set item appearances ...[1 item(s)] done [0.00s].
set transactions ...[1677 item(s), 8378 transaction(s)] done [0.35s].
sorting and recoding items ... [1583 item(s)] done [0.03s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3Mining stopped (time limit reached). Only patterns
of 3 returned! done [7.40s].
writing ... [211203 rule(s)] done [0.90s].
creating S4 object  ... done [0.28s].
```

The count 8 with Total Transactions 8378 and 1677 items. Total of 211203 rules. This clearly indicates that the non-match response is more in the target variable.

```
     lhs                                rhs       support     confidence lift     count
[1]  {pref_o_shared_interests=21}    => {match=0} 0.001074242 1          1.197199  9
[2]  {shared_interests_important=21} => {match=0} 0.001074242 1          1.197199  9
[3]  {age_o=18}                      => {match=0} 0.001074242 1          1.197199  9
[4]  {pref_o_ambitious=17.86}        => {match=0} 0.001193602 1          1.197199 10
[5]  {pref_o_funny=17.86}            => {match=0} 0.001193602 1          1.197199 10
[6]  {pref_o_intelligence=17.86}     => {match=0} 0.001193602 1          1.197199 10
[7]  {pref_o_sincere=17.86}          => {match=0} 0.001193602 1          1.197199 10
[8]  {pref_o_attractive=16.07}       => {match=0} 0.001193602 1          1.197199 10
[9]  {age=18}                        => {match=0} 0.001193602 1          1.197199 10
[10] {ambtition_important=17.86}     => {match=0} 0.001193602 1          1.197199 10
[11] {funny_important=17.86}         => {match=0} 0.001193602 1          1.197199 10
[12] {intellicence_important=17.86}  => {match=0} 0.001193602 1          1.197199 10
[13] {sincere_important=17.86}       => {match=0} 0.001193602 1          1.197199 10
[14] {attractive_important=16.07}    => {match=0} 0.001193602 1          1.197199 10
[15] {funny=4}                       => {match=0} 0.001193602 1          1.197199 10
[16] {d_age=16}                      => {match=0} 0.001432323 1          1.197199 12
[17] {d_age=26}                      => {match=0} 0.001432323 1          1.197199 12
[18] {d_age=15}                      => {match=0} 0.001432323 1          1.197199 12
[19] {intelligence_partner=1}        => {match=0} 0.001551683 1          1.197199 13
[20] {intelligence_o=1}              => {match=0} 0.001551683 1          1.197199 13
```

People who prefer a shared interest and people who are the age of 16,18,15 and very ambitions and who prefer so much intelligence, sincerity, attractiveness are not getting the match. In this case we can clearly see that people wanted to see the partner first know about them and then the match is good. So, meeting people is one thing which makes the match better than not meeting people.

6.2 Association based on the personal and the partner preference:

For this analysis the whole dataset has been divided into two based on the survey for personal and the partner side so that we can compare the similar variables for the target.

Dimension of the data Personal preference side without the decision variable: 8378 (rows) & 60 (col)

Dimension of the data Partner preference side without the decision variable: 8378 (rows) & 60 (col)

6.2.1 Personal Side Preference:

On personal side, people expectations to and how people with these preference got the match or not is analyzed here.

```
oneside[,names(oneside)]<-lapply(oneside[,names(oneside)],factor)
oneside<-as(oneside,"transactions")
itemFrequencyPlot(oneside[, itemFrequency(oneside) > 0.5], cex.names = 0.6)

rules1.oneside <- apriori(data = oneside , parameter = list( supp = 0.001 , conf = 0.8) , appearance = list(default = "lhs" , rhs = "match=1") )
rules1<-sort(rules1.oneside , decreasing = TRUE , by = 'lift')
## Sorting rules based on confidence,printing the rules
inspect(rules1[1:10])
```

I change the Rhs = Match=1 or 0 to get the association rules for both likes and dislikes between the partners

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.8    0.1    1 none FALSE            TRUE       5   0.001      1     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 8

set item appearances ...[1 item(s)] done [0.01s].
set transactions ...[1559 item(s), 8378 transaction(s)] done [0.11s].
sorting and recoding items ... [1453 item(s)] done [0.02s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4Mining stopped (time limit reached). Only patterns up to a length of 4 returned! done [46.96s].
writing ... [4651 rule(s)] done [7.46s].
creating S4 object  ... done [1.42s].
```

```
     lhs                                                                                    rhs       support     confidence lift     count
[1]  {funny_o=9,shared_interests_partner=10,like=10}                                     => {match=1} 0.001193602 1          6.071014 10
[2]  {gender=male,attractive_o=9,like=10}                                                => {match=1} 0.001074242 1          6.071014  9
[3]  {race=European/Caucasian-American,shared_interests_o=9,shared_interests_partner=10} => {match=1} 0.001432323 1          6.071014 12
[4]  {samerace=1,funny_o=9,shared_interests_partner=10}                                  => {match=1} 0.001193602 1          6.071014 10
[5]  {shared_interests_partner=10,movies=9,met=1}                                        => {match=1} 0.001432323 1          6.071014 12
[6]  {pref_o_attractive=20,shared_interests_o=8,shared_interests_partner=10}             => {match=1} 0.001074242 1          6.071014  9
[7]  {pref_o_ambitious=10,attractive_o=8,shared_interests_partner=10}                     => {match=1} 0.001074242 1          6.071014  9
[8]  {shared_interests_o=10,funny_partner=9,like=8}                                      => {match=1} 0.001432323 1          6.071014 12
[9]  {samerace=1,shared_interests_o=10,funny_partner=9}                                  => {match=1} 0.001193602 1          6.071014 10
[10] {shared_interests_o=10,attractive_important=20,shared_interests_partner=8}          => {match=1} 0.001074242 1          6.071014  9
```

6.2.2 Partner Side Preference:

It's always better to check the other side of the preference to get to share the interest and get the match. In this scenario when checking the partner side preference, I could find different results which is very interesting. People who are above 25 age who are sincere and funny gets the top rule. The second rule is interesting, people are giving importance to race is an interesting variable which I have not seen in any of the other association rules. Total of 135 rules have been generated.

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target   ext
        0.9    0.1    1 none FALSE            TRUE       5   0.001      1     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 8

set item appearances ...[1 item(s)] done [0.00s].
set transactions ...[267 item(s), 8378 transaction(s)] done [0.03s].
sorting and recoding items ... [254 item(s)] done [0.02s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4Mining stopped (time limit reached). Only patterns up to a length of 4 returned! done [27.88s].
writing ... [135 rule(s)] done [0.91s].
creating S4 object  ... done [0.16s].
```

```
     lhs                                                                                  rhs       support     confidence lift     count
[1]  {age=25,d_sincere=4,d_funny_partner=7}                                           => {match=1} 0.001074242 1          6.071014  9
[2]  {race_o='Black/African American',d_shared_interests_partner=7,d_like=9}          => {match=1} 0.001074242 1          6.071014  9
[3]  {age_o=23,d_age=0,d_shared_interests_partner=9}                                  => {match=1} 0.001074242 1          6.071014  9
[4]  {age=23,d_age=0,d_shared_interests_partner=9}                                    => {match=1} 0.001074242 1          6.071014  9
[5]  {age=25,d_attractive_important=10,d_shared_interests_partner=9}                  => {match=1} 0.001551683 1          6.071014 13
[6]  {age=23,age_o=23,d_shared_interests_partner=9}                                   => {match=1} 0.001074242 1          6.071014  9
[7]  {age_o=23,d_age=0,d_shared_interests_o=9}                                        => {match=1} 0.001074242 1          6.071014  9
[8]  {age=23,d_age=0,d_shared_interests_o=9}                                          => {match=1} 0.001074242 1          6.071014  9
[9]  {age_o=25,d_pref_o_attractive=10,d_shared_interests_o=9}                         => {match=1} 0.001551683 1          6.071014 13
[10] {age=23,age_o=23,d_shared_interests_o=9}                                         => {match=1} 0.001074242 1          6.071014  9
```

6.3 Association based on specific interest attributes:

Specific interest attributes includes how much people like Sports, TV, exercise, dining, museums, art, hiking, gaming, clubbing, reading, theater, movies, music, shopping, yoga.

These attributes are listed for both partners. And this was included in the past association rules. But did not had much effect or tops the list with even 0.7 confidence. So, I decided to analyze only these specific interests separately to get an idea of the population how they get the match based on these attributes alone.

```
SP=read.csv("https://raw.githubusercontent.com/vigneshjmurali/Statistical-Predictive-Modelling/master/Datasets/SPECIFIC%20INTEREST_ARULES.csv")
#oneside<-oneside[,-c(61)]
dim(SP)
#View(oneside)
SP[,names(SP)]<-lapply(SP[,names(SP)],factor)
SP<-as(SP,"transactions")
itemFrequencyPlot(SP[, itemFrequency(SP) > 0.5], cex.names = 0.6)

rules1.SP <- apriori(data = SP , parameter = list( supp = 0.001 , conf = 0.9) , appearance = list(default = "lhs" , rhs = "match=0") )
rules1.SP<-sort(rules1.SP , decreasing = TRUE , by = 'lift')
## Sorting rules based on confidence,printing the rules
inspect(rules1.SP[1:10])
```

```
 [1] 8378   35

Absolute minimum support count: 8

set item appearances ...[1 item(s)] done [0.00s].
set transactions ...[233 item(s), 8378 transaction(s)] done [0.03s].
sorting and recoding items ... [233 item(s)] done [0.01s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5Mining stopped (time limit reached). Only patterns
up to a length of 5 returned! done [18.35s].
writing ... [25039 rule(s)] done [5.11s].
creating S4 object  ... done [1.22s].
     lhs                                rhs       support     confidence lift
count
[1]  {tvsports=10,theater=1}         => {match=1} 0.001074242 0.9        5.463913 9
[2]  {exercise=10,theater=1}         => {match=1} 0.001074242 0.9        5.463913 9
[3]  {gaming=7,theater=1}            => {match=1} 0.001074242 0.9        5.463913 9
[4]  {clubbing=9,theater=1}          => {match=1} 0.001074242 0.9        5.463913 9
[5]  {art=5,theater=1}               => {match=1} 0.001074242 0.9        5.463913 9
[6]  {theater=1,d_clubbing=9}        => {match=1} 0.001074242 0.9        5.463913 9
[7]  {theater=1,shopping=7}          => {match=1} 0.001074242 0.9        5.463913 9
[8]  {dining=10,theater=1}           => {match=1} 0.001074242 0.9        5.463913 9
[9]  {theater=1,concerts=7}          => {match=1} 0.001074242 0.9        5.463913 9
[10] {theater=1,d_gaming=7}          => {match=1} 0.001074242 0.9        5.463913 9
[11] {tvsports=10,tv=1}              => {match=1} 0.001074242 0.9        5.463913 9
```

25039 rules generated

People who like tvsports, exercise, clubbing, dining have better chance of finding a match in the event from the result on left

People who don't like dining, music, movies, clubbing have no chance of getting a match in the event from result below

```
set item appearances ...[1 item(s)] done [0.01s].
set transactions ...[233 item(s), 8378 transaction(s)] done [0.04s].
sorting and recoding items ... [233 item(s)] done [0.01s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5Mining stopped (time limit reached). Only patter
up to a length of 5 returned! done [18.70s].
writing ... [4274037 rule(s)] done [5.98s].
creating S4 object  ... done [2.49s].
     lhs                          rhs       support     confidence lift     count
[1]  {dining=1,music=3}        => {match=0} 0.001074242 1          1.197199 9
[2]  {dining=1,movies=3}       => {match=0} 0.001074242 1          1.197199 9
[3]  {dining=1,art=2}          => {match=0} 0.001074242 1          1.197199 9
[4]  {dining=1,clubbing=2}     => {match=0} 0.001074242 1          1.197199 9
[5]  {dining=1,museums=3}      => {match=0} 0.001074242 1          1.197199 9
[6]  {dining=1,concerts=3}     => {match=0} 0.001074242 1          1.197199 9
[7]  {dining=1,theater=3}      => {match=0} 0.001074242 1          1.197199 9
[8]  {dining=1,reading=5}      => {match=0} 0.001074242 1          1.197199 9
[9]  {dining=1,tv=3}           => {match=0} 0.001074242 1          1.197199 9
[10] {dining=1,hiking=9}       => {match=0} 0.001074242 1          1.197199 9
[11] {dining=1,gaming=4}       => {match=0} 0.001074242 1          1.197199 9
```

4274037 rules generated where we can clearly see that non-match response is more in data.

People who don't like dinning, music, movies have lower chance of finding a match in the event from the result on left

6.4 Summary: Associations

Overall, the association rules were able to give some useful insights on the dataset. Lots of things were clarified using associations and understood few interesting rules popped up in the result form the codes. With associations I could change the confidence and inspect more rules.

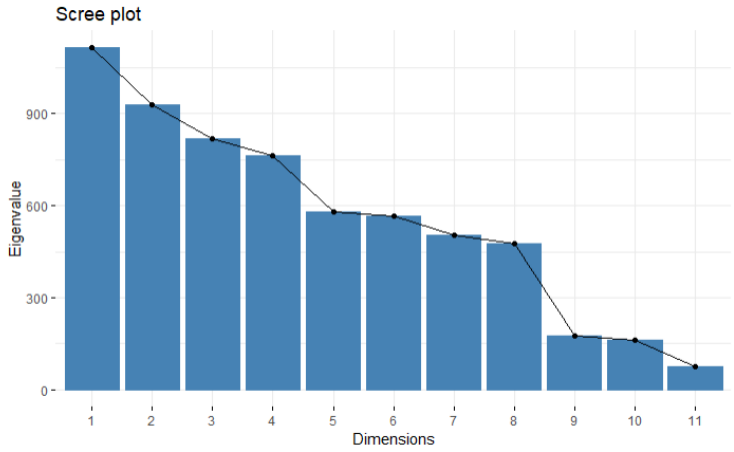
- Being funny, sincere, age above 26 and having same shared interest matched most of the people at the event
- Being age of 18 & difference in age of more than 16 with specific race was both disliked and liked by most of the people at the event.
- Having specific interest of tvsports, theater & exercise got the match easy rather than not liking music, movies and dining.

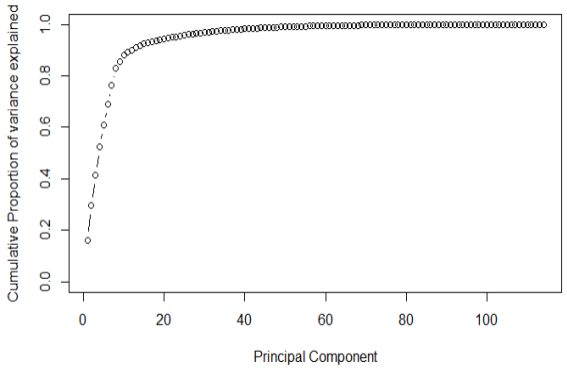
7. Unsupervised learning: Cluster Analysis: Used R Program

The cluster analysis was used without the target variable in this to check how the observations were grouped or formed in how many clusters and to check if its making any sense to do it.

Dimension of the actual dataset = 8378 (rows) and 123 (col)
The target variable and other string variables were removed to process the clustering
Dimension of the dataset after removing string variables + Target: 8378 (rows) and 114 (col)
I used PCA for dimensionality reduction so, first I scaled the data and then did the pca with function prcomp in R.

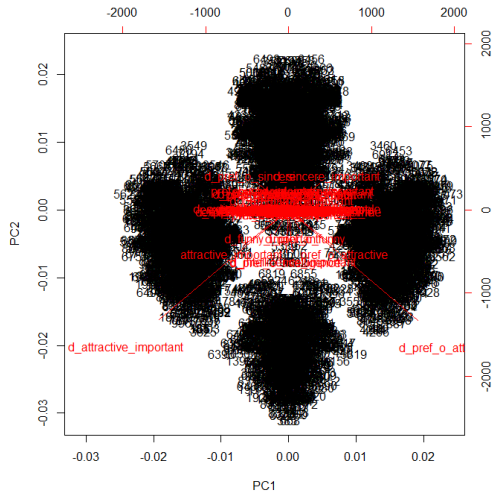
```
data.clus.s=scale(data.clus) #Scalling the data
data.clus.pca=prcomp(data.clus) #Performing the PCA
data.clus.pca$rotation[1:5,1:5]
data.clus.sd=data.clus.pca$sdev #determining the standard deviation
data.clus.var=data.clus.pca$sdev^2
data.clus.var[1:10] #listing of the variance
# Proportion of variance
pve=data.clus.var/sum(data.clus.var)
which.max(cumsum(pve)[cumsum(pve)<0.90]) # determining 90% of information in what PC
cumsum(pve[11])
fviz_screeplot(data.clus.pca,ncp=11,choice="eigenvalue") #Ploting 90% pc
plot(cumsum(pve),xlab="Principal Component", ylab="Cumulative Proportion of variance
explained",ylim=c(0,1),type='b')
biplot(data.clus.pca,arrow.len=0)
```





From this graph we can clearly see that 11 PC's comprise of 90% of the information.

I performed the biplot to see how all information on both samples and variables of data matrix is displayed graphically.

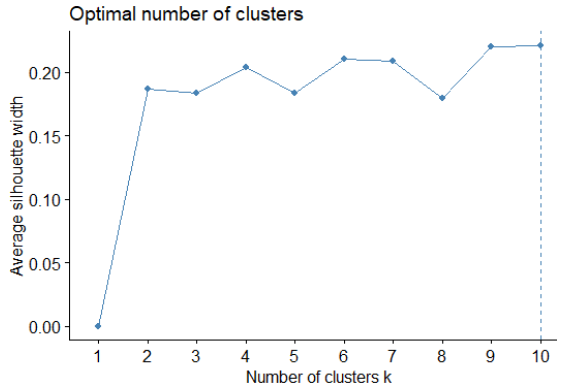


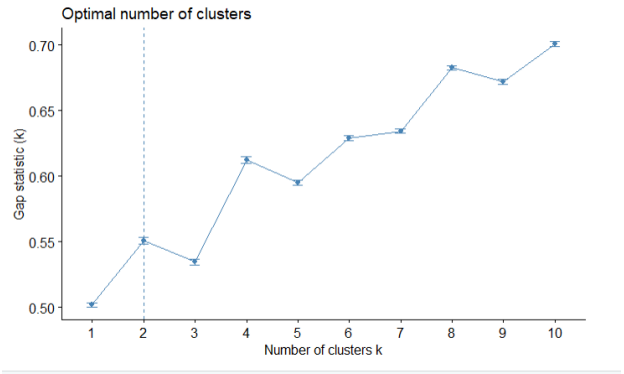
The biplot generated here with the actual dataset is kind of different and looks totally not understandable. This might be the reason because the data not only includes the personal preference and like of the partner, but it also includes the partner preference and like of the partner which is basically dividing the groups like this in the dataset. Also, the variables is divided like positive and negative is almost equal size of all the groups in the data. Hopefully I can get more insights in how it is divided after evaluating the number of clusters needed for the dataset.

7.1 Determining the number of clusters:

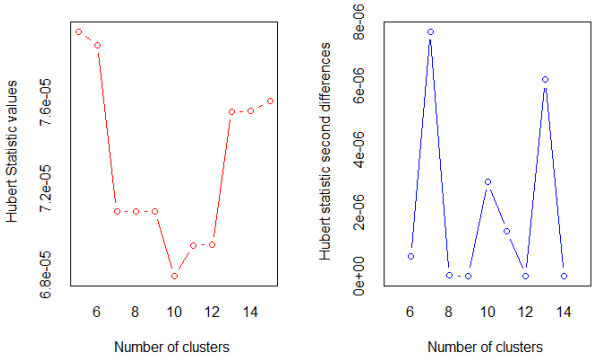
The optimal number of clusters is determined with models like Kmeans (WSS, Silhouette, Gap stat), NBClust and MClust.

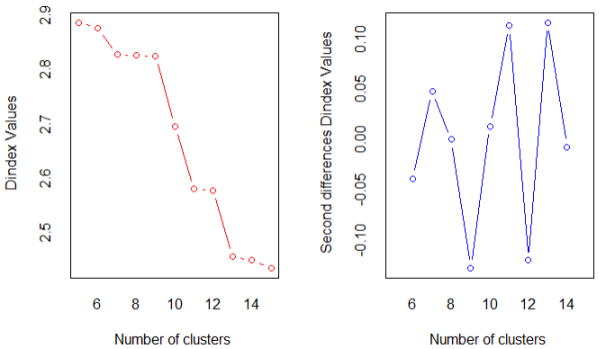
```
# OPTIMAL NUMBER OF CLUSTERS - (1) WSS, (2) Silhouette, (3) Gap_stat, (4) NbClust # For PC 1 to PC 11
data.clus_new=data.clus.pca$x[,1:11]
data.clus_new.s=scale(data.clus_new)
set.seed(10)
#Kmeans
fviz_nbclust(data.clus_new,kmeans,method="wss") # Using elbow method - wss #k=4
fviz_nbclust(data.clus_new,kmeans,method="gap_stat") #k=2
fviz_nbclust(data.clus_new,kmeans,method="silhouette") # k=10
#NBClust
data.clus.nbclust<-data.clus_new %>% #Using NbClust # k=5
 scale() %>%
  NbClust(distance="euclidean",min.nc=5,max.nc=15,method="complete",index="all")
#MClust
data.clus.mclust<-Mclust(data.clus_new.s)  # n = 9
summary(data.clus.mclust)
fviz_mclust(data.clus.mclust,"BIC",palette="jco")
fviz_mclust(data.clus.mclust,"classification",geom="point",pointsize=1.5, palette="jco")
```

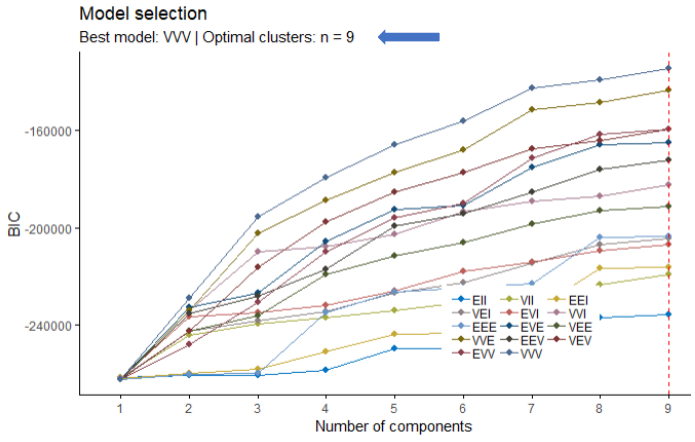





```
                second differences plot) that corresponds to a significan
increase of the value of
                the measure.

*****************************************************************
* Among all indices:
* 4 proposed 5 as the best number of clusters
* 8 proposed 6 as the best number of clusters
* 2 proposed 10 as the best number of clusters
* 1 proposed 11 as the best number of clusters
* 1 proposed 12 as the best number of clusters
* 6 proposed 13 as the best number of clusters
* 1 proposed 15 as the best number of clusters

                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is  6

*****************************************************************
```

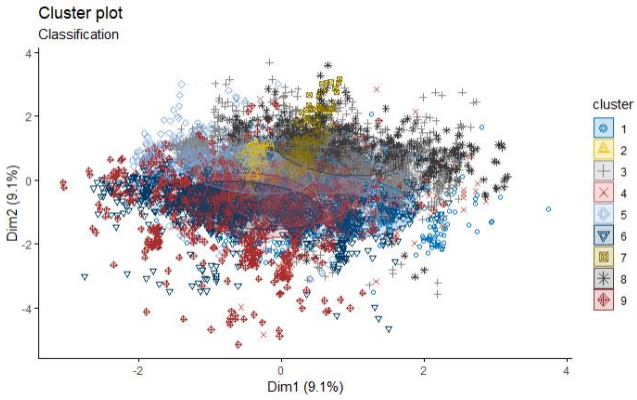


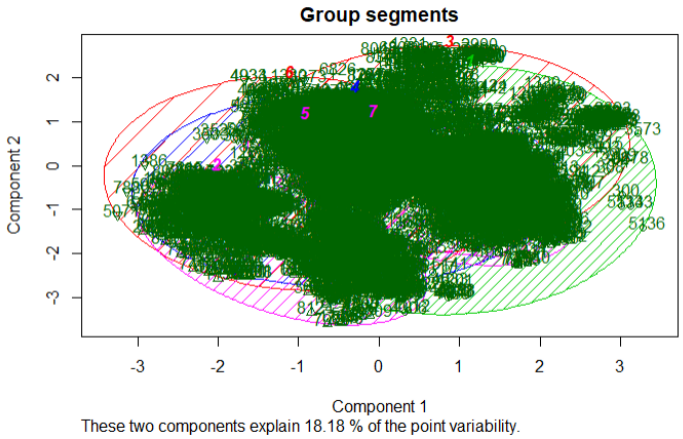




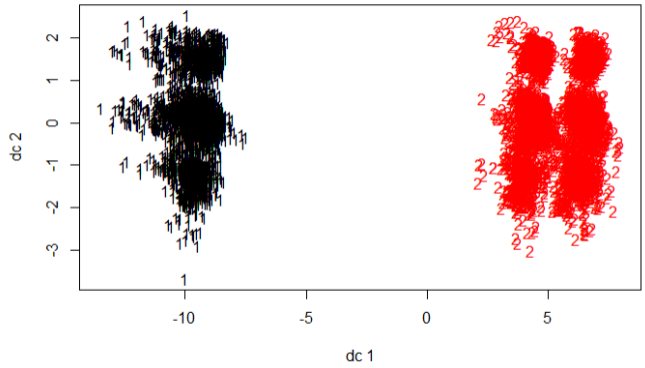
MODEL NAME	# OF CLUSTERS
SILHOUETTE	10
GAP_STAT	2
WSS	8
NBCLUST	5
MCLUST	9

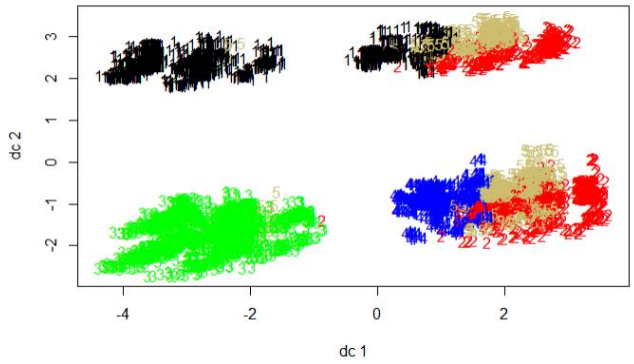
The number of clusters is determined for the actual dataset. Here since the kmeans (Silhouette, Gap Stat), NBClust and MClust have majority more than 5 I will use average number of clusters as K=7. Using this I am going to do the fuzzy cluster.





By seeing this the groups of clusters are not separated or clearly see it's like a fuzzy one with lots of overlapping points in the graph. This indicates that number of clusters 7 doesn't work that good to see the distinct groups of clusters.





When I tried only with 2 clusters in the above graph as suggested by Gap stat it divides the group very good. Any number of clusters more than 2 having difficulty in distinguishing the groups among the dataset. I constructed hierarchical clustering using complete linkage with 7 clusters but, it doesn't make good sense which is not including in report.

7.2 Determining Cluster Analysis in Personal side preference and ratings:

The personal side preference data is taken separately as in association done earlier. The cluster analysis was used without the target variable in this to check how the observations were grouped only for the personal side

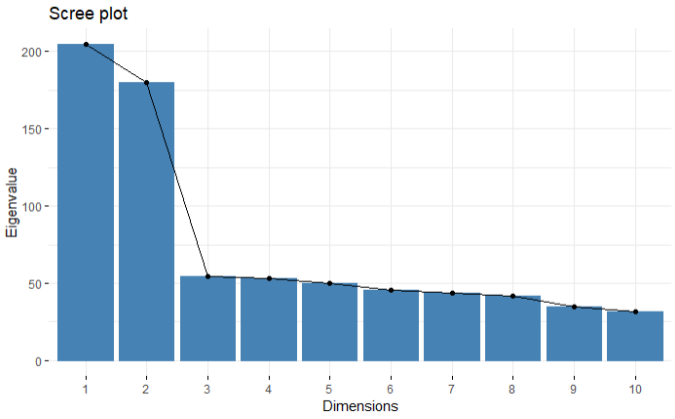
Dimension of the dataset = 8378 (rows) and 62 (col)

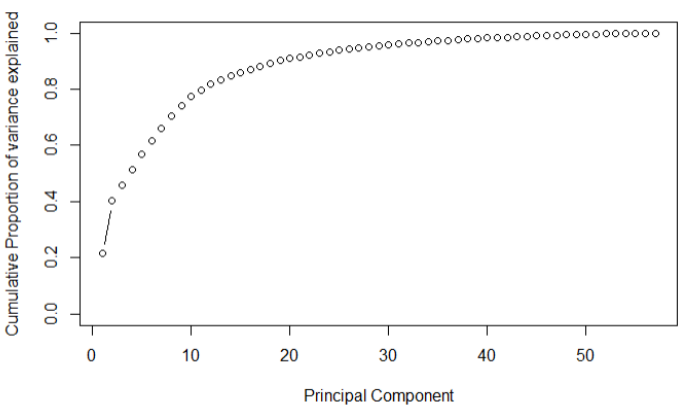
The target variable and other string variables were removed to process the clustering

Dimension of the dataset after removing string variables + Target: 8378 (rows) and 57 (col)

```
attributes=read.csv("https://raw.githubusercontent.com/vigneshjmurali/Statistical-Predictive-Modellir
Datasets/attributes%20of%20data.csv")
dim(attributes)
data.attri<-attributes[,-c(1,5,9,61,62)] ; dim(data.attri)
data.attri.mat<-as.matrix(data.attri)
data.attri.pca<-prcomp(data.attri.mat)
data.attri.pca$rotation[1:5,1:5]
data.attri.sd=data.attri.pca$sdev; data.attri.var=data.attri.pca$sdev^2 #To compute variance
data.attri.var[1:5] ; pve=data.attri.var/sum(data.attri.var) ; cumsum(pve[1:10])
fviz_screeplot(data.attri.pca,np=10,choice="eigenvalue")
which.max(cumsum(pve)[cumsum(pve)<0.90]) ; biplot(data.attri.pca)
#Choosing the principle components as new variables
data.attri.new=data.attri.pca$x[,1:18] ; data.attri.new.s=scale(data.attri.new)
```

Dimensionality reduction is done through PCA and 18 PC's were selected which has 90% of information

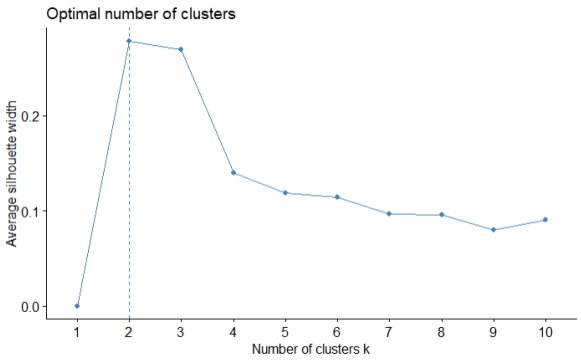




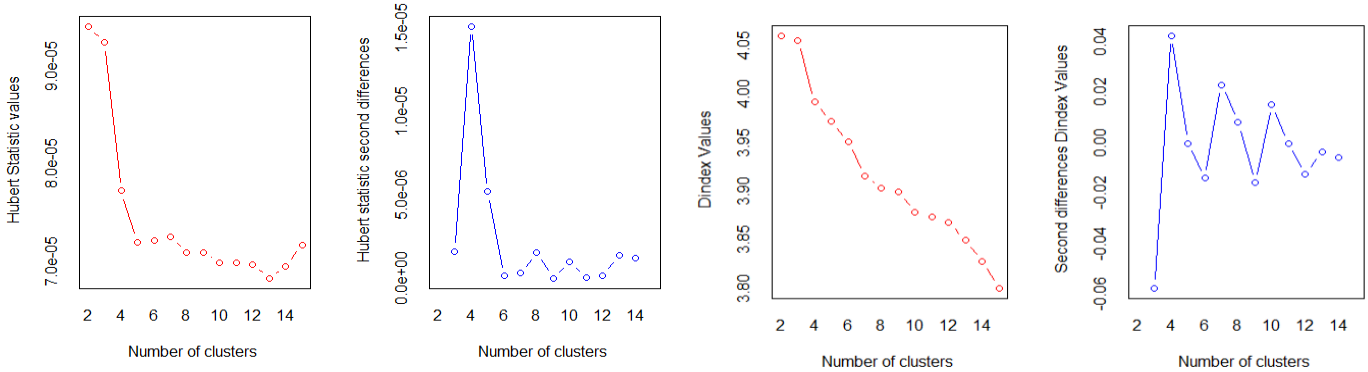
After the PCs are determined to determine the number of clusters I used kmeans, NBClust, and Mclust

```
#K means clustering
set.seed(10)
#fviz_nbclust(data.attri.new,kmeans,method="wss")
fviz_nbclust(data.attri.new,kmeans,method="silhouette")
fviz_nbclust(data.attri.new,kmeans,method="gap_stat")
#NBCLUST - use unless you want to have to wait for long time
data.attri.nbclust<-data.attri.new %>% #Using NbClust
  scale() %>%
  NbClust(distance="euclidean",min.nc=2,max.nc=15,method="complete",index="all")
#fuzzy kmeans
set.seed(10)
km_11_2.fit=kmeans(data.attri.new,3,nstart=50)
plotcluster(data.attri.new,km_11_2.fit$cluster)
fviz_cluster(km_11_2.fit,data=data.attri.new,ellipse.type="convex",palette="jco",ggtheme=theme_minimal())

#MCLUST
data.attri.fit<-Mclust(data.attri.new.s[,0:18])
summary(data.attri.fit)
fviz_mclust(data.attri.fit,"BIC",palette="jco")
fviz_mclust(data.attri.fit,"classification",geom="point",pointsize=1.5, palette="jco")
fviz_mclust(data.attri.fit,"uncertainty", palette="jco")
```

Model Name	# of Clusters
Silhouette	2
NBClust	4
MClust	9

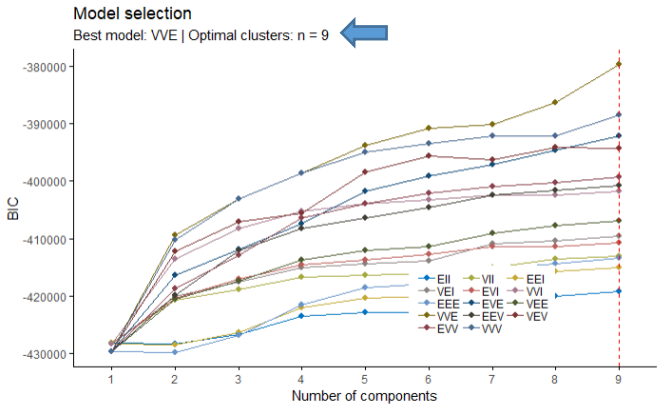


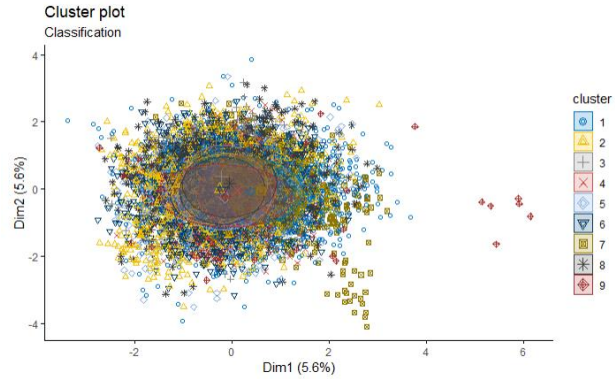

```
*******************************************************************
* Among all indices:
* 7 proposed 2 as the best number of clusters
* 3 proposed 3 as the best number of clusters
* 8 proposed 4 as the best number of clusters
* 3 proposed 5 as the best number of clusters
* 1 proposed 7 as the best number of clusters
* 1 proposed 9 as the best number of clusters

                 ***** Conclusion *****

* According to the majority rule, the best number of clusters is  4

*******************************************************************
```

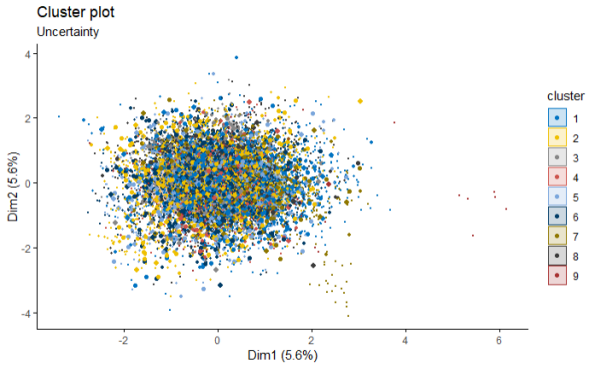


Both Silhouette and NBClust suggested 2 clusters

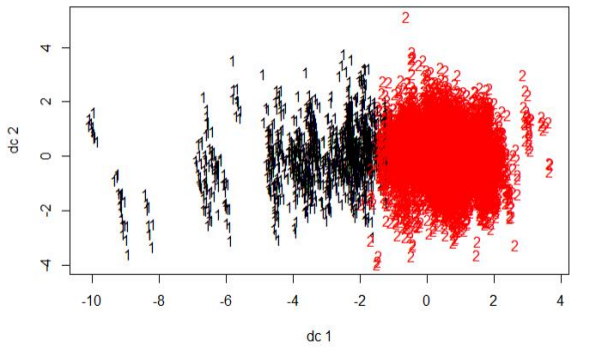
```
> summary(data.attri.fit)
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust VVE (ellipsoidal, equal orientation) model with 9 co

 log.likelihood    n  df       BIC       ICL
      -187602.4 8378 485 -379585.9 -382085.8

Clustering table:
   1    2    3    4    5    6    7    8    9
1537 1514  811  290  953 1199  775 1081  218
> fviz_mclust(data.attri.fit,"BIC",palette="jco")
```

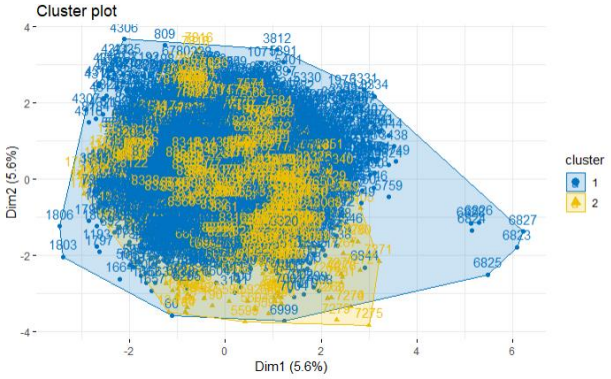


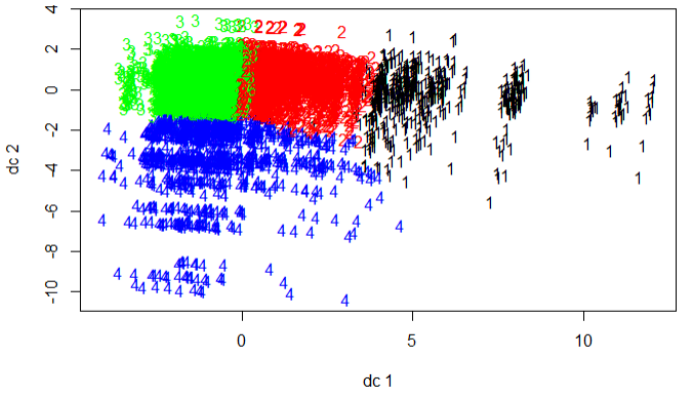


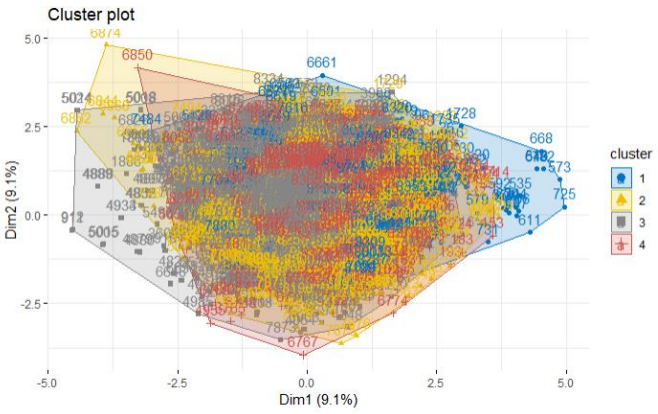


By determining with K= 2 when I plot the cluster the 2 groups are distincly see in the plot below. But when I use more than 2 clusters the plot still makes good groupings.









From this, I would really go ahead with k=2 & K=4 clusters for Personal side preference and ratings subdivision

7.3 Determining Cluster Analysis on Partner side preference and ratings:

The Partner side preference data is taken separately as in association done earlier. The cluster analysis was used without the target variable in this to check how the observations were grouped only for the Partner side

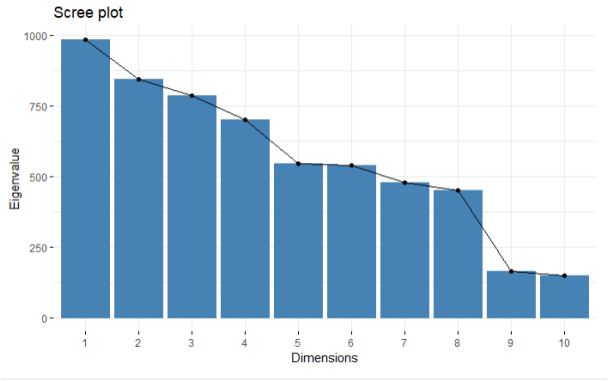
Dimension of the dataset = 8378 (rows) and 62 (col)

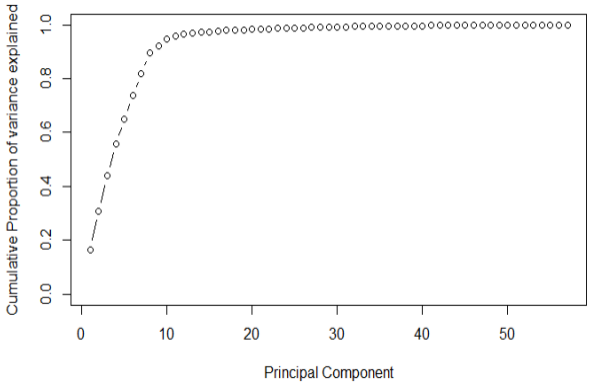
The target variable and other string variables were removed to process the clustering

Dimension of the dataset after removing string variables + Target: 8378 (rows) and 57 (col)

```
attributes1=read.csv("https://raw.githubusercontent.com/vigneshjmurali/Statistical-Predictive-Modellir
/Datasets/attributes%20of%20data1.csv")
dim(attributes1)
data.attri1<-attributes1[,-c(1,5,9,61,62)]
dim(data.attri1)
data.attri1.mat<-as.matrix(data.attri1)
data.attri1.pca<-prcomp(data.attri1.mat)
data.attri1.pca$rotation[1:5,1:5]
data.attri1.sd=data.attri1.pca$sdev; data.attri1.var=data.attri1.pca$sdev^2 #To compute variance
data.attri1.var[1:5]
pve=data.attri1.var/sum(data.attri1.var) ; cumsum(pve[1:10])
fviz_screeplot(data.attri1.pca,np=10,choice="eigenvalue")
plot(cumsum(pve),xlab="Principal Component", ylab="Cumulative Proportion of variance
explained",ylim=c(0,1),type='b')
which.max(cumsum(pve)[cumsum(pve)<0.90])
```

Dimensionality reduction is done through PCA and 8 PC's were selected which has 90% of information

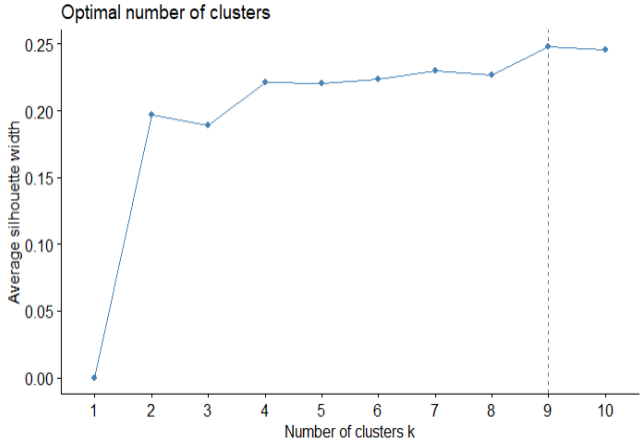


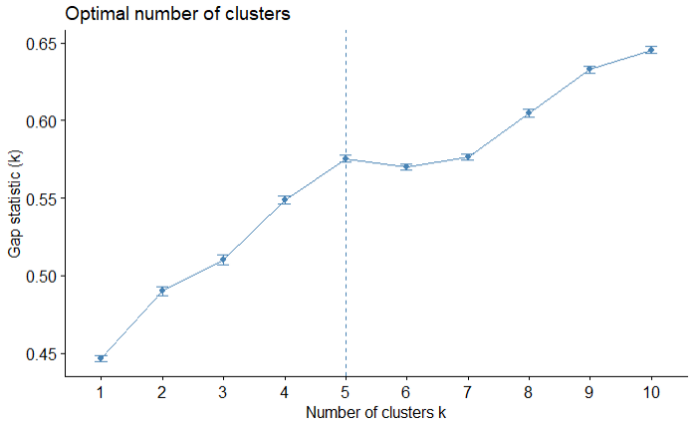


After # of PCs are determined, to find number of clusters I used kmeans (Silhouette & Gap stat), NBClust, & Mclust

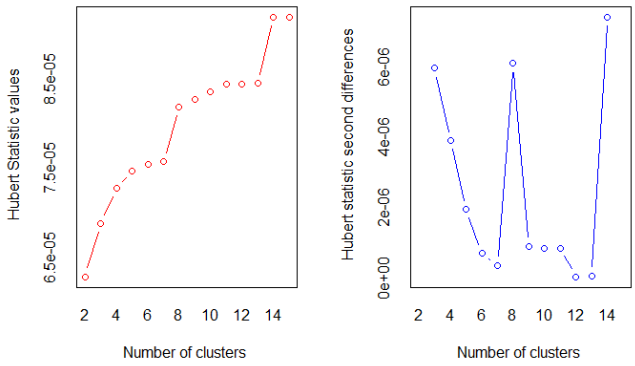
```
data.attri1.new=data.attri1.pca$x[,1:8]
data.attri1.new.s=scale(data.attri1.new)
#K means clustering
set.seed(10)
fviz_nbclust(data.attri1.new,kmeans,method="wss") #k=4
fviz_nbclust(data.attri1.new,kmeans,method="silhouette") # k=9
fviz_nbclust(data.attri1.new,kmeans,method="gap_stat") #k=5

#NBCLUST - use unless you want to have to wait for long time
data.attri1.nbclust<-data.attri1.new %>% #Using NbClust
  scale() %>%
  NbClust(distance="euclidean",min.nc=2,max.nc=15,method="complete",index="all")

#fuzzy kmeans
set.seed(10)
km_11_2_attri1.fit=kmeans(data.attri1.new,2,nstart=50)
plotcluster(data.attri1.new,km_11_2_attri1.fit$cluster)
fviz_cluster(km_11_2_attri1.fit,data=data.attri1.new,ellipse.type="convex",palette="jco",ggtheme=theme_minimal())

#MCLUST
data.attri1.fit<-Mclust(data.attri1.new.s[,0:8])
summary(data.attri1.fit)
data.attri1.fit$modelName
fviz_mclust(data.attri1.fit,"BIC",palette="jco")
fviz_mclust(data.attri1.fit,"classification",geom="point",pointsize=1.5, palette="jco")
fviz_mclust(data.attri1.fit,"uncertainty", palette="jco")

data.attri1.group<-data.frame(data.attri1.new,km_11_8_attri1.fit$cluster)
#write.csv(data.attri1.group$km_11_8_attri1.fit.cluster,"attri_group1_8clus.csv")
```

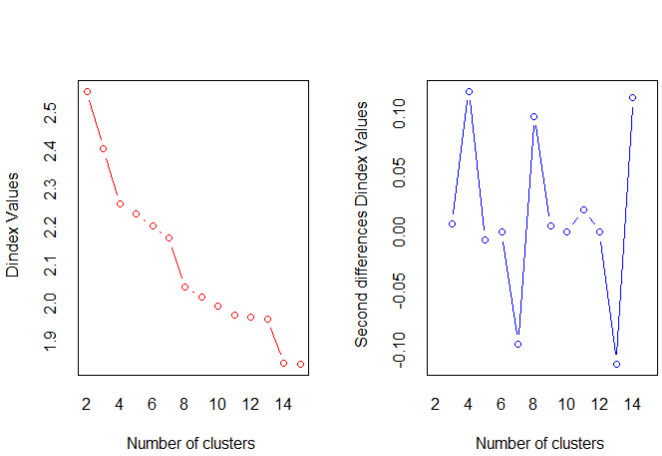
Model Name	# of Clusters
Silhouette	9
Gap_Stat	5
NBClust	4
MClust	9

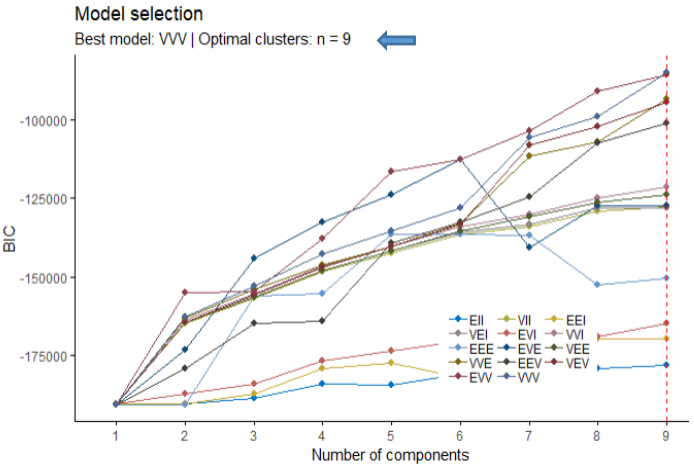


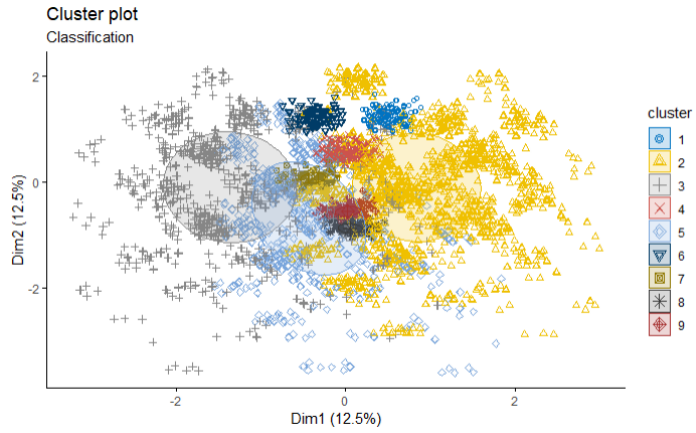


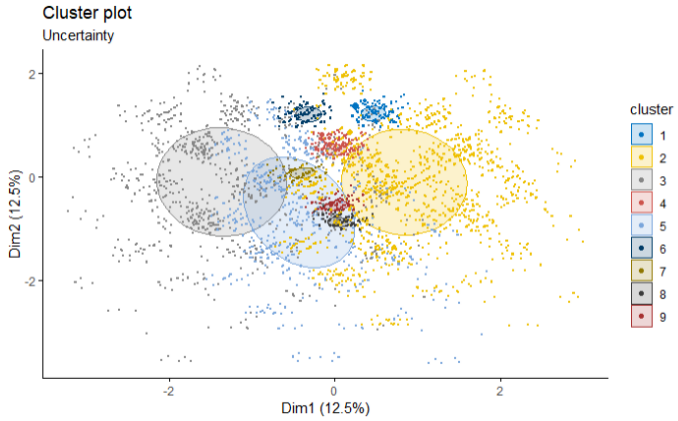
```
*****************************************************************
* Among all indices:
* 2 proposed 2 as the best number of clusters
* 3 proposed 3 as the best number of clusters
* 5 proposed 4 as the best number of clusters
* 2 proposed 5 as the best number of clusters
* 1 proposed 8 as the best number of clusters
* 1 proposed 10 as the best number of clusters
* 3 proposed 11 as the best number of clusters
* 2 proposed 13 as the best number of clusters
* 2 proposed 14 as the best number of clusters
* 2 proposed 15 as the best number of clusters

                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is  4
```



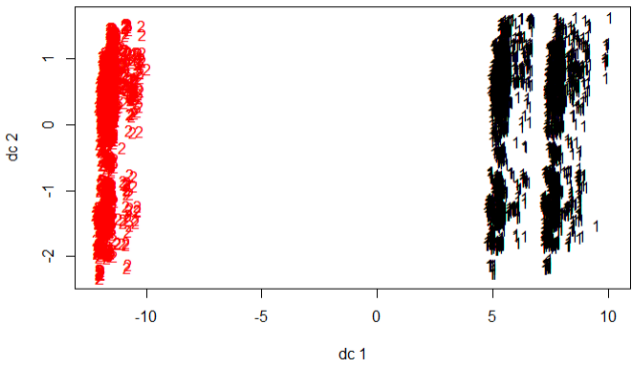


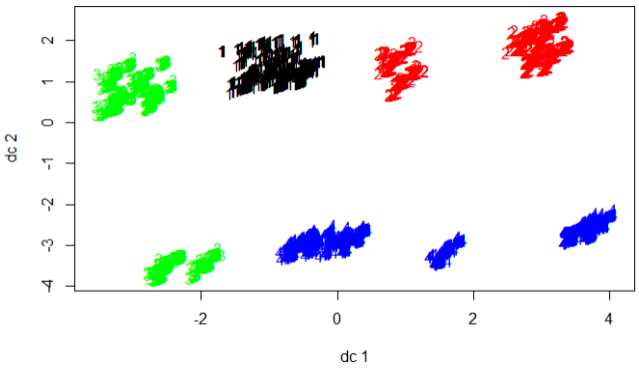


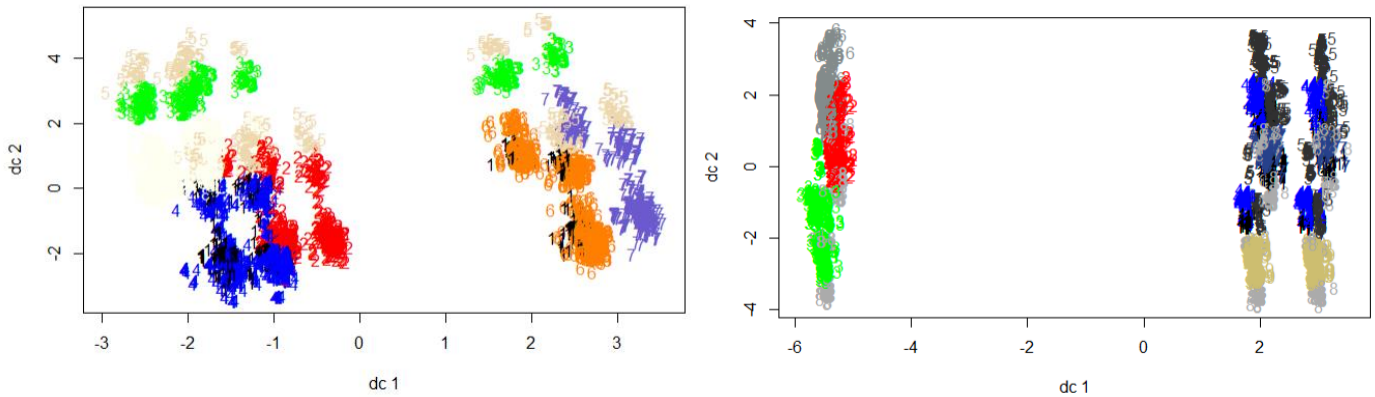


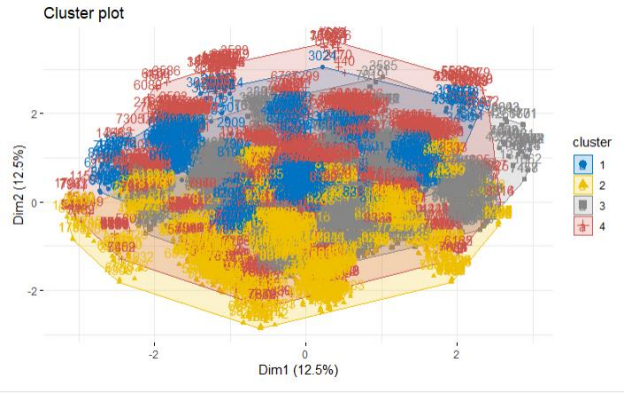


First when I assign K=2 the groupings were really distinguishing and good but as suggested by NBClust if I use K=4 the groupings divide into 4 of different colors but groupings as 8 when I put k=8, it's surprising to see more groupings but same colors. I tried until k=9 but still except for k=2 other groupings doesn't make good sense to me in this analysis.







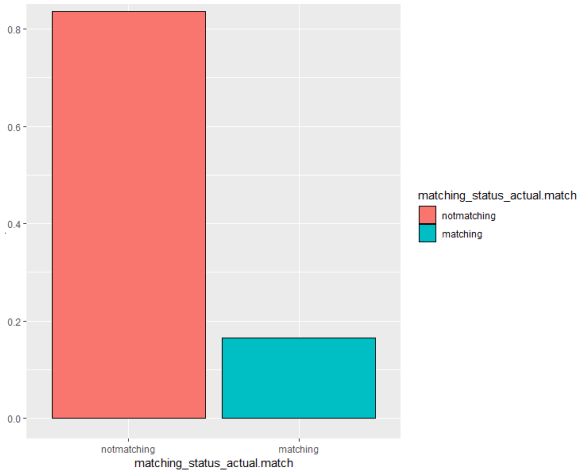


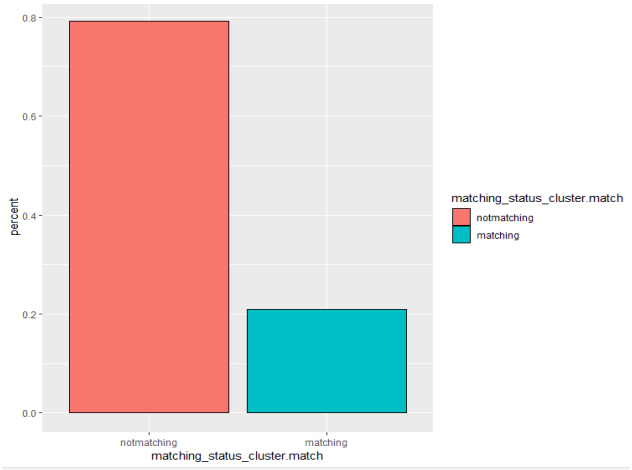
I would really go ahead with k=2 clusters for Partner side preference and ratings subdivision data

7.4 Summary: Cluster Analysis

Overall, the clustering analysis were able to give some useful insights on the different parts of dataset. Lots of things were clarified using this analysis.

- Though the determination of # of clusters were performed and gave results, when I perform I the plot by assigning the # of clusters, the plot doesn't make good sense to the observations with actual dataset.
- When I divide the dataset into two parts based on the survey analysis, the same situation arises and couldn't clarify or group the observations more than k=2.
- By using k=2 for the cluster and adding the group with the actual dataset I used this to see the decision of the personal and partner decision to see if it equals the match variable in the dataset. When I compared it, there was another interesting result known as seen below. The change in "match" "not match" count is differed by decrement in not matching from 83% to 79% and increment of matching category from 17% to 21%





8. Project Conclusion:

In the end this project with the speed dating dataset, I had lots of chances to analyze with all three learning materials used in class like Supervised (Classification), Unsupervised (Clustering) & Association rules. I wouldn't recommend using seriation analysis for this dataset because, the dataset doesn't have anything related to rankings or ordering. From all the analysis the total summary is being the target variable as binary, the classification worked better with having logistic, decision tree and random forest performed excellent, useful variable relationship insights were analyzed to reveal what brings the two people match in the speed dating event like people who love dinning & music also how this whole survey has been divided with group using cluster analysis. The dataset like this will always be useful to analyze using these techniques and getting knowledge from this was very useful to understand better to work for my future.