Big Data Technologies – CS989 Assignment.

"An analysis of speed-dating data with regard to its application in modern-day dating apps."

By: Frank Mitchell
Course: MSc Artificial Intelligence and Applications.

Contents

List of Figures and Tables

Chapter One: Introduction

The dataset I have chosen to analyse focuses on experimental speed-dating information, consisting of answered questions from 8378 attendees between 2002-2004. Each attendee had a four-minute “first date” with every other member of the opposite sex. Once they had completed their short date, attendees were asked to rate how likely it was (between 0 and 10) that they would see their partner again. Each individual was also asked to rate their partner on 6 subjective attributes. As a result of my initial analysis, these subjective attributes - together with ratings on self-perception, actual age and whether they matched with the other person – will form the basis for my inquiry. The attributes are as follows:

i. Attractiveness
ii. Sincerity
iii. Shared Interests
iv. Intelligence
v. Ambition
vi. Humour

As mentioned, each attendee was tasked with rating themselves and how much they value each attribute in a partner, while their partner was tasked with rating the participant. All ratings, considering the above subjective attributes, constituted the best variables to answer my question:

“Can I use these subjective ratings to improve matching for users of a modern-day dating application?”

Chapter Two: Key Challenges and Problems

With the advent of powerful mobile technology enhancing global connectivity, it was clear that dating culture would change significantly. Speed dating, singles nights and the era of blind dates are a thing of the past, with people preferring these days to meet a partner online (and increasingly through mobile apps such as Tinder, Bumble and Hinge).

However, the ability of these apps to accurately match people has always been questionable. More recently, dating apps have used heuristic algorithms for the benefit of the company and not the users (J.Clement, 2019). Hence, having discovered the above dataset, I seen this as an opportunity to apply machine learning techniques to increase the effectiveness of dating applications.

The key to any successful date is matching with your partner, accounted for with the "Match" column indicting a 0 for "Yes" and a 1 for "No". As we'll see, what's important during the date are:

i. How you feel about yourself
ii. How you feel about the other person
iii. How they feel about you.

These components of the dating model are framed within the subjective attributes described above. These will form the basis of my PCA (Principle Component Analyses) dimensionality reduction, before using that analysis to inform K-Means clustering and Support Vector Machines.

There were multiple transformations required before beginning analysis, such as expunging null or redundant values, as well as converting all my numbers (represented as strings) into integers. However, the columns that contained the "preferred partner" ratings posed a particular problem.

[0-15]
[0-15]
[0-15]
[0-15]
[0-15]
[0-15]
[0-15]
[0-15]
[21-100]
[21-100]
[21-100]
[21-100]
[21-100]
[21-100]
[21-100]
[21-100]
[21-100]
[21-100]

Table 2.1 An extract from the column "you_pref_attr" which rates how much the participant rates attractiveness in a partner.

The values contained within these columns (figure 2.1) contain a set of range values. The ranges are as follows:

i. [0 – 15]
ii. [16 – 20]
iii. [21 – 100]

Due to the peculiar nature of these figures I tried to discover their meaning but without success. Rather than remove the feature altogether I felt it suitable to choose a random value within this range to conduct the transformation.

Another point to note is the effect that this transformation had on the distribution of values contained within these columns.

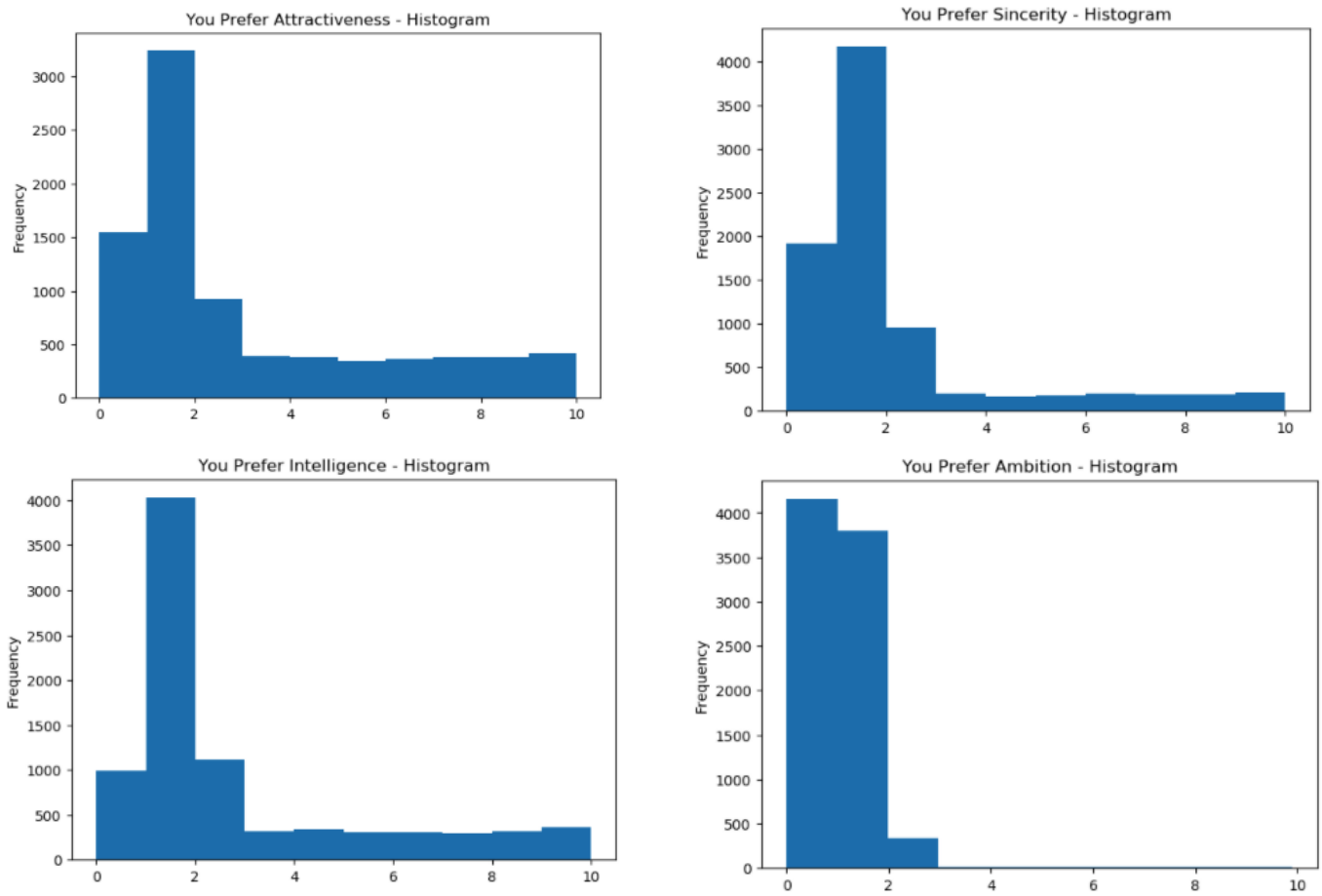

Figure 2.1 Distribution of participant's preference after range value transformation.

As per the histograms above, we can see that after transformation the data is heavily right-hand skewed. Having only affected a small proportion of the features though, I felt the skewed distribution would be negligible to my results. In a real-world example this would not be an issue, as the user of the dating app would be asked to give a discrete rating from 0 – 10.

Chapter Three: Summary Statistics

The purpose of this report is to improve matching and grouping criteria in dating apps, and this all starts with the user. As such it was important to understand the distribution of people – ages, races and gender – and of course the frequency of matches within the data. I began with an examination of participants race.

```
r = baseline_speed_date['race'].unique()
print(r)
```

```
['Asian' 'European' 'Other' 'South American' 'African' None]
```

Table 3.1 – Examining unique values in the 'Race' column.

As demonstrated (figure 3.1) – after shortening the race values to their respective continents – the data accounts for a large proportion of the world. However, I needed to examine the variations in race to get a better impression of the people involved in my dataset.

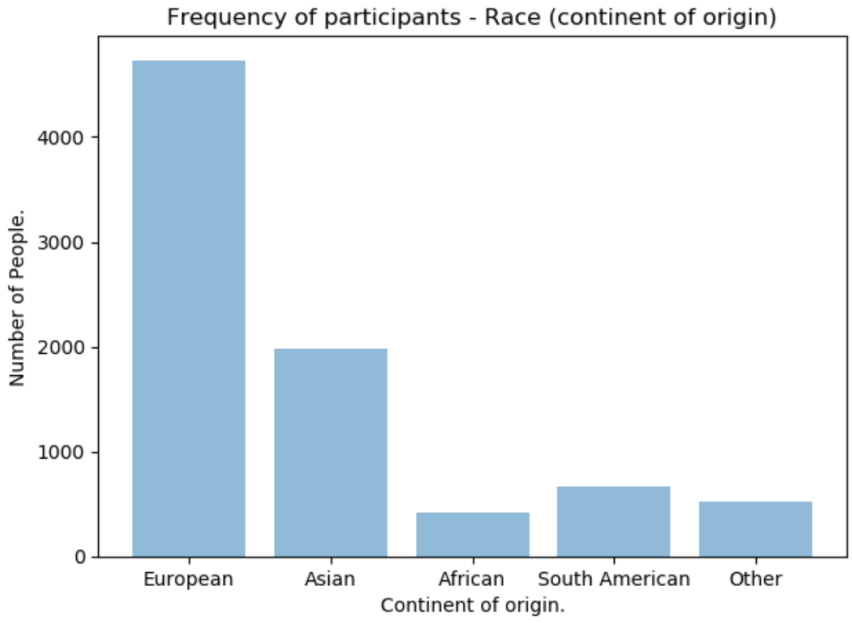

Figure 3.1 Bar chart displaying frequency of participants race.

This was the first bias I encountered, clearly demonstrating that, after European and Asian people, the rest of the world is hardly represented. Again, in the context of a dating app, it was important to understand this split in terms of gender to note where improvements could be made to the quality of the data in the future.

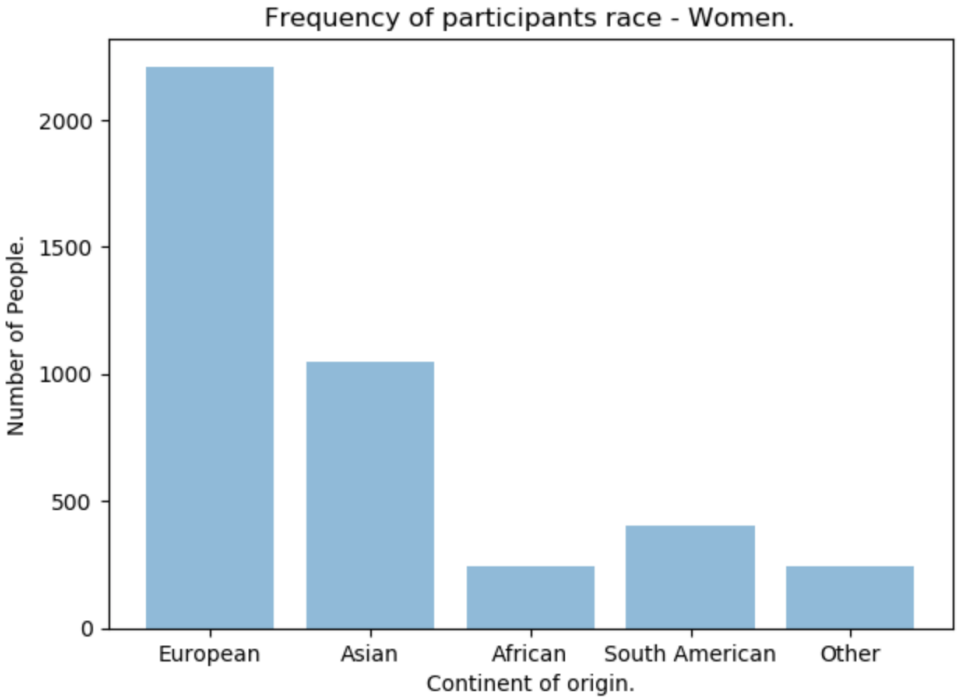

Figure 3.2 Bar Chart - frequency of female participants by race.

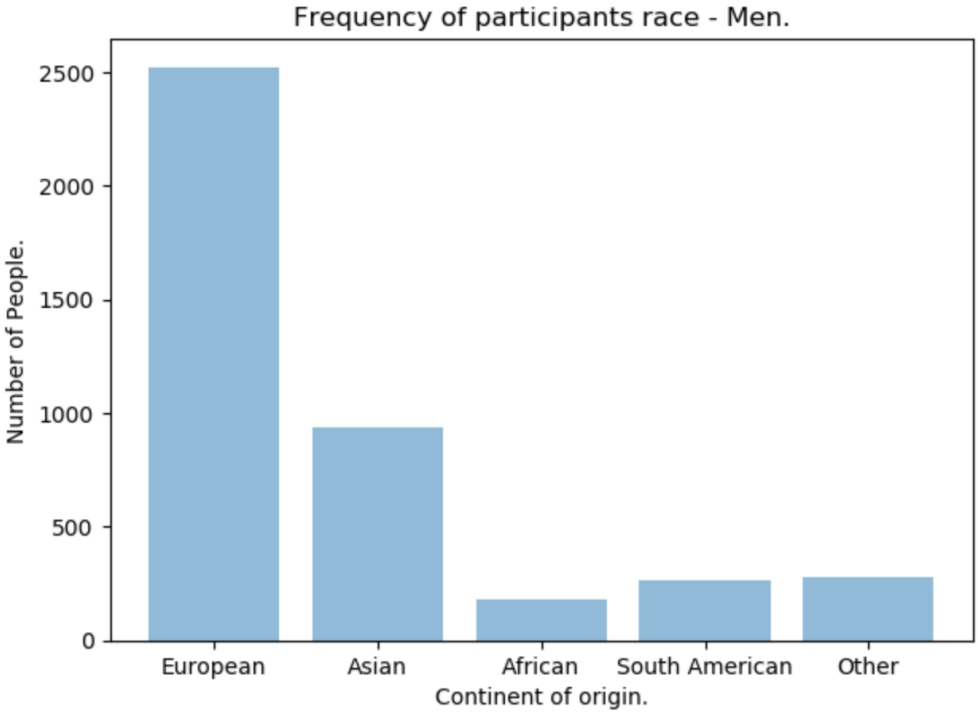

Figure 3.3 Bar Chart - frequency of male participants by race.

In terms of racial split over gender, there's an under-representation of African and South American men and, to a lesser extent, African and South American females. To build a better model I would endeavour to obtain a greater variance in the race of participants.

Another important aspect to examine before conducting analysis was the age of the participants. This was vital in terms of modern-day dating apps, as the largest proportion of users is between 18-34 (Clement, 2019), so I needed to know if my dataset fell within this normal distribution. Firstly, I inspected the description of the ages to make sure they were within a realistic range.

```
In [108]: ages.describe()

Out[108]: count    8283.000000
          mean       26.358928
          std         3.566763
          min        18.000000
          25%        24.000000
          50%        26.000000
          75%        28.000000
          max        55.000000
          Name: age, dtype: float64
```

Table 3.2 Description of age variable

After describing the age variable, I could clearly see that my values made sense. The ages began at 18 (as expected) and 75% are within a reasonable range of typical dating app users. In order to better visualise the ages, I constructed a boxplot.

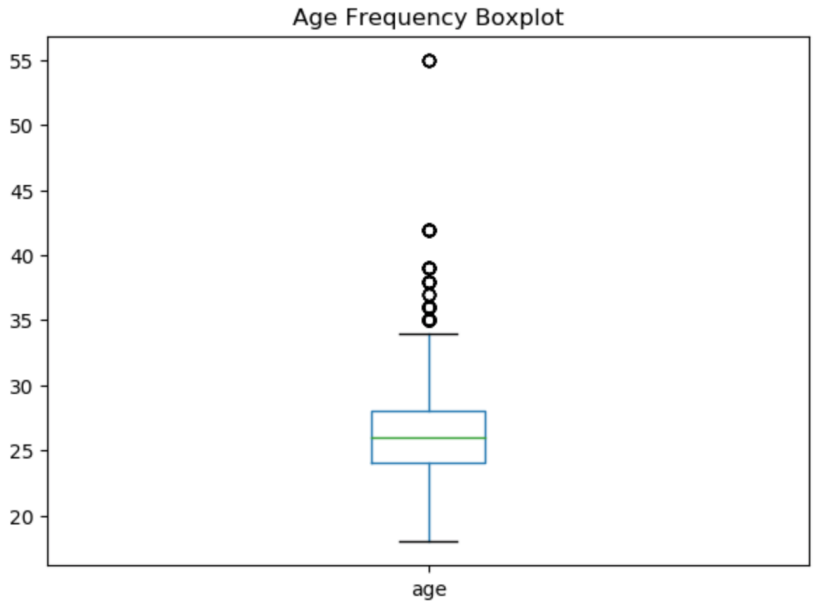

Figure 3.4 Boxplot of age variance in dataset.

Through boxplot visualisation it's easier to see that each of my ages fell within the normal distribution of dating-app users. Outliers from the age of 35 onwards is exactly what I would hope to see when examining the age ranges of participants. As a further examination of the variance in ages, a density plot provides a slightly more granular examination.

.

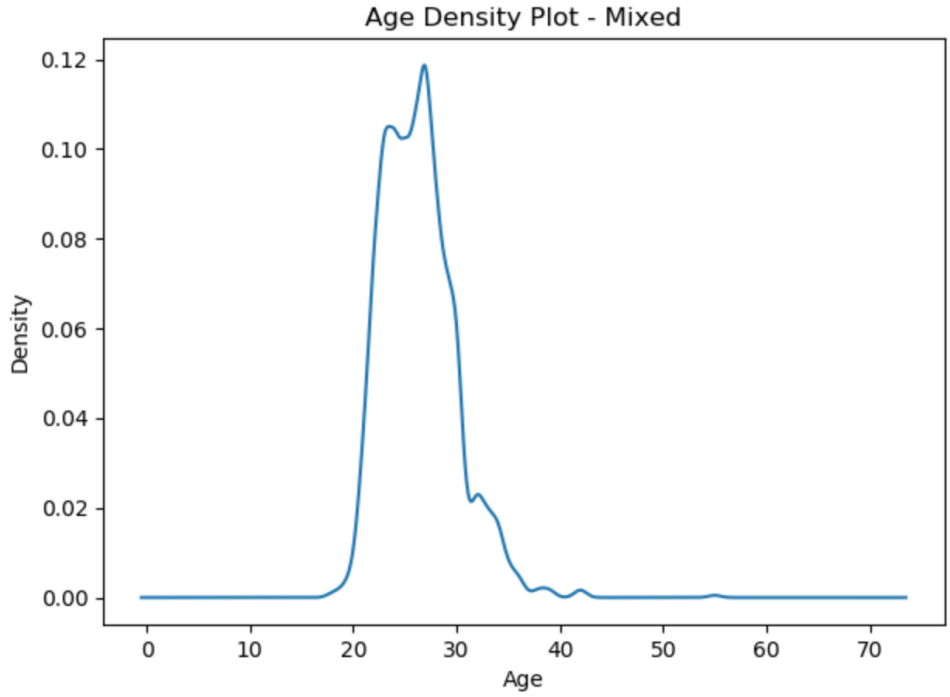

Figure 3.5 Density Plot of age variance within dataset.

With the above density plot I can clearly see the mini peaks within our outliers, giving me a better idea of the outlying age ranges within the dataset.

After examining the frequency of races within genders and checking the age ranges were within an acceptable distribution of ‘regular’ dating app users, the next aspect to examine were the matches themselves. It was important to see the extent to which people can successfully match without the aid of machine learning.

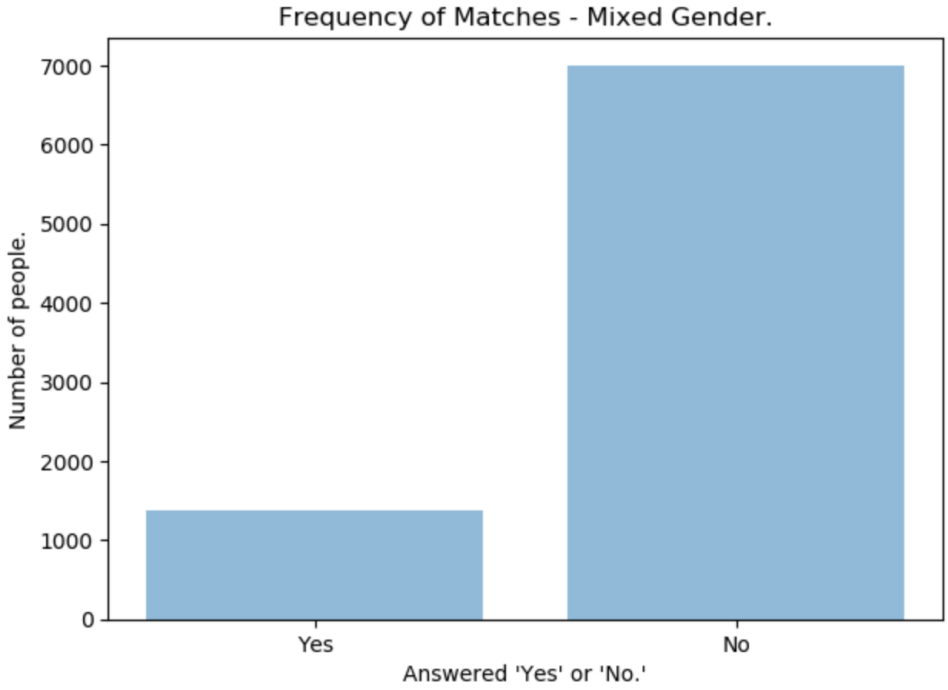

Figure 3.6 Bar chart of ‘Match’ frequency across genders

As can clearly be seen from figure 3.6, a huge majority of people decided not to match with their respective partners. I can only put this consistent lack of matching down to incorrect pairing and the completely random nature of speed-dating events. We could also possibly account for this in the amount of time that each person spends with their respective partner, four minutes seeming a short amount of time to make a decision.

Now that I had an excellent overview of my data, it was time to start examining specific variables, with a view to using this information to inform my Principle Component Analysis. I used a heatmap to conduct this examination by grouping all the variables I considered as possibly important.

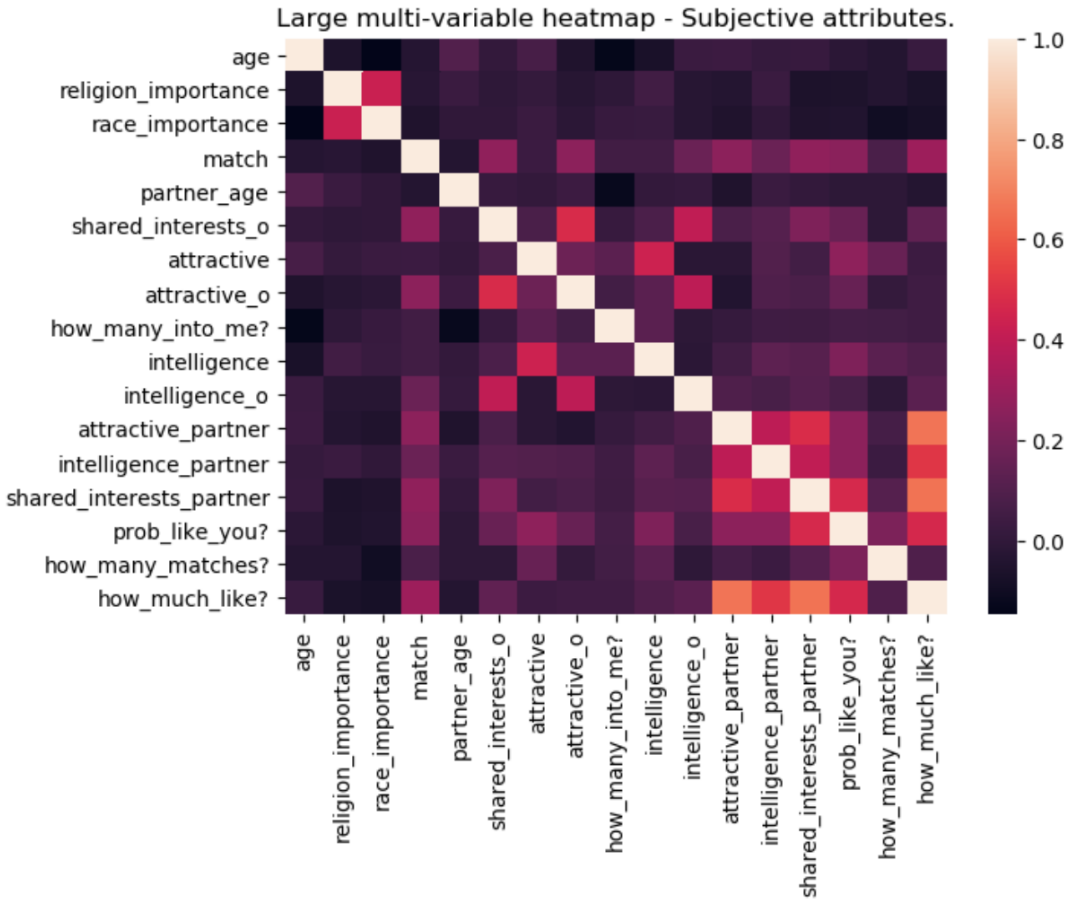

Figure 3.7 Heatmap of subjective attribute variables, age, religious and racial importance.

The heatmap confirmed my suspicions, that the largest correlations can be found within the subjective ratings that each participant conducts on themselves and one another. We have an expected correlation between *'How much you like the other person'* and how you've rated their various attributes. However, other interesting points to note are the correlation between racial and religious importance and the correlation between '*Intelligence'* and '*Attractiveness'*, both interesting aspects that could be explored out with this report.

The main points to note from the figure above though, are the large square of correlations encompassing a *'Match'* and the various ratings of participants. The analysis of these variables, moving forward, will inform my methods.

Chapter Four: Principle Component Analyses

The first part of my analysis involves conducting some dimensionality reduction, primarily to improve visualisation, but also as a source of focused information to feed my learning techniques. Currently, using my heatmap as a source of variables, I am working within 16-dimensional space. It would be incredibly difficult (and not very meaningful) to visualise all these features at once. Instead we can condense each feature down to its first two principle components. For this I'll use PCA (Principle Component Analysis).

As mentioned, visualising our full 16-dimensional data is difficult, so using PCA I can derive the first two principle components (those that describe the greatest variance) and visualise in 2-dimensional space. By reducing the space to 2 dimensions I felt it beneficial for informing my supervised and unsupervised methods.

The first task was to collect all the variables of importance into a single data-frame and check null values.

```
Out[147]: religion_importance           79
          race_importance               79
          match                          0
          partner_age                  104
          shared_interests_o          1076
          attractive                   105
          attractive_o                 212
          how_many_into_me?              0
          intelligence                 105
          intelligence_o               306
          attractive_partner           202
          intelligence_partner         296
          shared_interests_partner    1067
          prob_like_you?               309
          how_many_matches?              0
          how_much_like?               240
          dtype: int64
```

Table 4.1 Variable grouping for PCA analysis with null value count.

In PCA, every output depends, to some degree, on every input, so without removing these values the entire vector would become null. Considering the comparatively low instances of NaN values I determined that their removal would have a negligible effect on my results.

Next, I used the *StandardScaler* object to scale the data contained within my heatmap. Once scaled and transformed the data is ready to be fit to the PCA model. The number of components was set to 2 (deriving the first 2 principle components) and the *transform* method called to apply the rotation and dimensionality reduction.

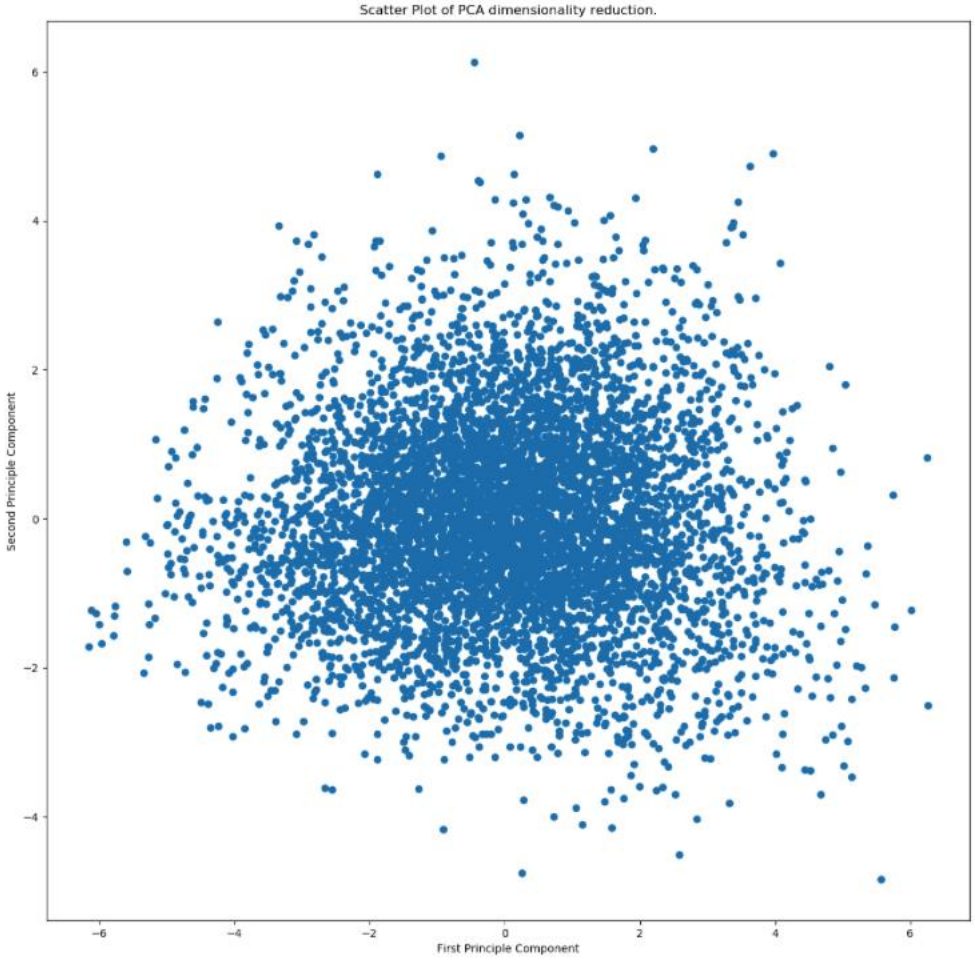

Figure 4.1 Scatter plot of PCA dimensionality reduction on 16 dimensions to two principle components.

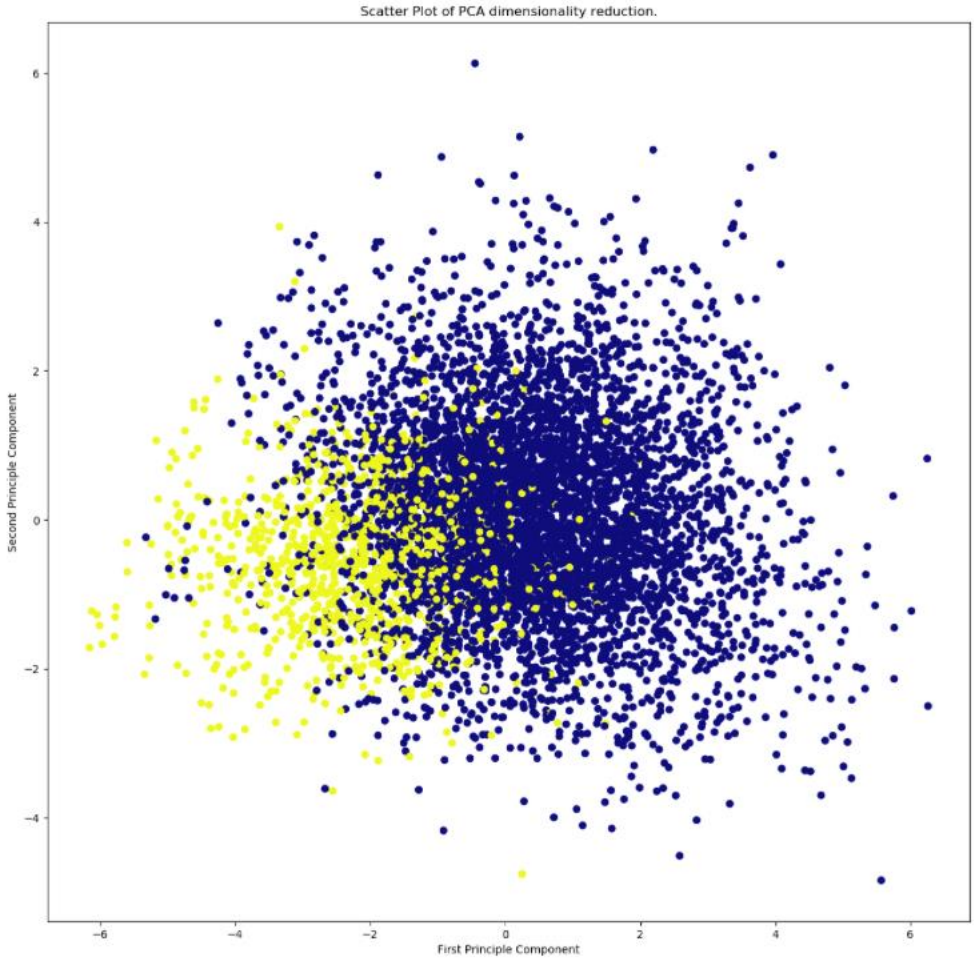

Figure 4.2 Scatter Plot of dimensionality reduction with points coloured based on the 'match' column.

The ability of PCA to reduce the dimensionality in aiding visulasation can be seen in figure 4.1, but figure 4.2 shows the real power of this technique. Based on the first two principle components we can see a clear separation on the '*Match'* attribute (yellow for '*Yes'* and blue for *'No'*). These matches are based off the first two principle components vs. the entire 16-dimensions. The components do not relate one-to-one with my subjective attributes, rather they correspond to combinations of the original features.

```
array([[ 0.05256943,  0.05833805, -0.28546693,  0.01122809, -0.23015576,
        -0.13301145, -0.18243384, -0.06953795, -0.15683718, -0.16620571,
        -0.35628325, -0.32936869, -0.41632196, -0.36488521, -0.1154474 ,
        -0.44271584],
       [-0.06157988, -0.07013104, -0.13185418, -0.03258706, -0.45305845,
        -0.24320512, -0.54621908, -0.07330513, -0.17457344, -0.38517605,
         0.3127133 ,  0.14411094,  0.18353206,  0.01839264,  0.00792772,
         0.27059806]])
```

Table 4.2 The Eigenvectors are stored as an attribute of the PCA object.

The individual Eigenvectors can be viewed by calling the *.components_* method on the PCA object. Each row represents a principle component and each column relates back to the original features. This can be better visualised using a heatmap.

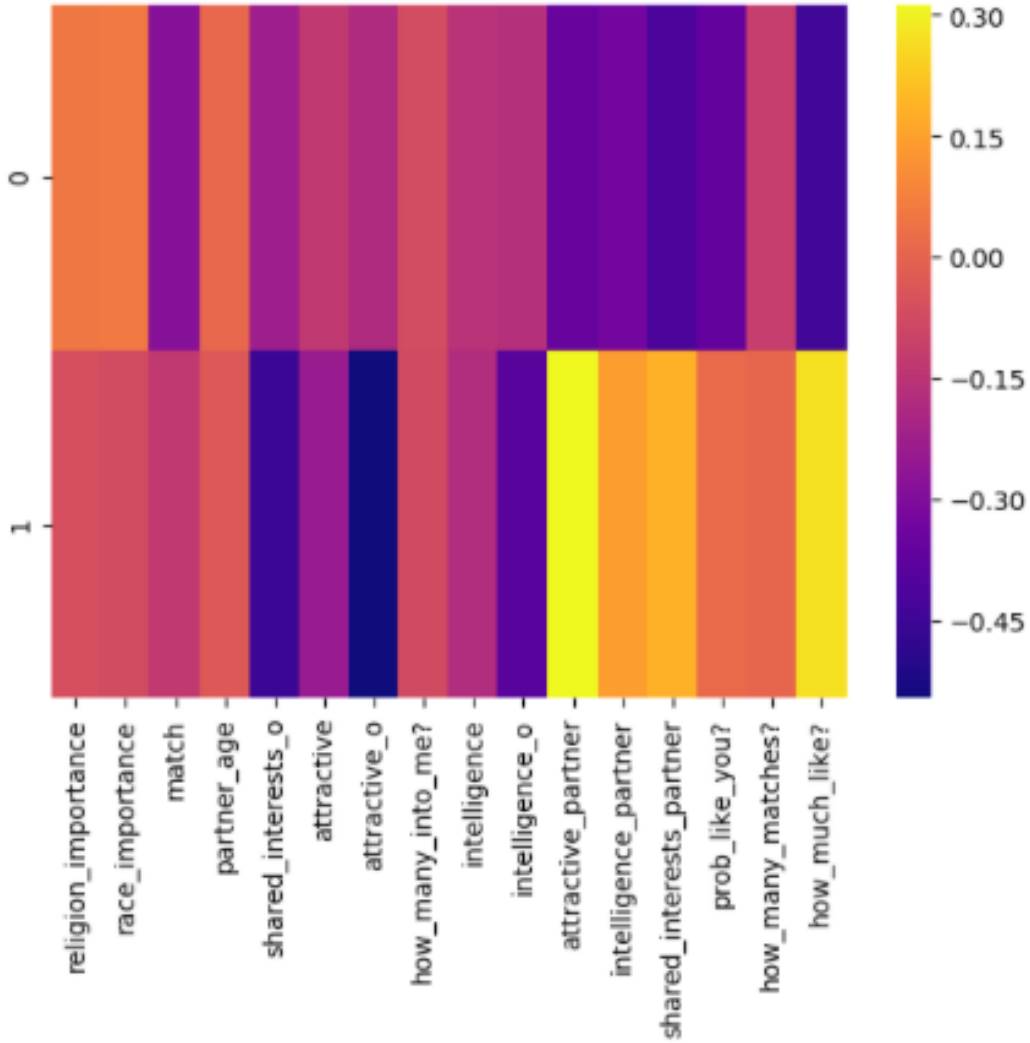

Figure 4.3 Heatmap showing correlation of Principle Components (0 & 1) to each of the 16 subjective attributes.

After creating a dataframe from my PCA object (PCA1, PCA2, and their relationship to each of the 16 original attributes) I created the heatmap above. This shows the correlation between various features and the principle components, with each PCA shown as a row. The “hotter” looking the colour, the more correlated to a specific feature.

By conducting the above dimensionality reduction, I have achieved the ability to visualise my data using the principle components that encompass the maximum variance. I can now use this data in reduced dimensions to inform my K-Means clustering and Support Vector Machines analyses.

Chapter Five: Unsupervised Method – K-Means

I used dimensional reduction principally as a means to create a better visual overview of the data, but also as an enhanced input for both my supervised and unsupervised methods. I felt that with my data represented by two principle components encompassing the maximum variance, I would have a greater chance of successfully clustering into appropriate segments (Harrison, 2018).

My aim in using K-Means is to cluster clouds of similar users based on the various subjective ratings. By continually rating other users' profiles by the above subjective criteria I would hope to optimise their number of successful matches. With my data nicely visualised in two dimensions I can use the PCA object components as my model parameters. Initially I set the number of clusters to an arbitrary 8, after which I trained and tested the model to examine the scores.

```
After implementing the arbitary value of 8 clusters.

Completeness Score is 0.08387168945458685.
Homogeneity Score is 0.36351643756649077.
```

Table 5.1 Completeness and Homogeneity score of K-Means model after initially setting to 8 clusters.

After fitting my model, I examined each of the scores, using the 'match' column as a target variable and the K-means object labels. As figure 5.1 shows, my homogeneity score (the measure by which each cluster contains members of a single class) is very low, but my completeness score (the measure by which my clustering has successfully assigned samples to the correct label) is relatively high.

To demonstrate our scoring results, I've plotted the data in figure 5.2 below, which shows a clear grouping of the data.

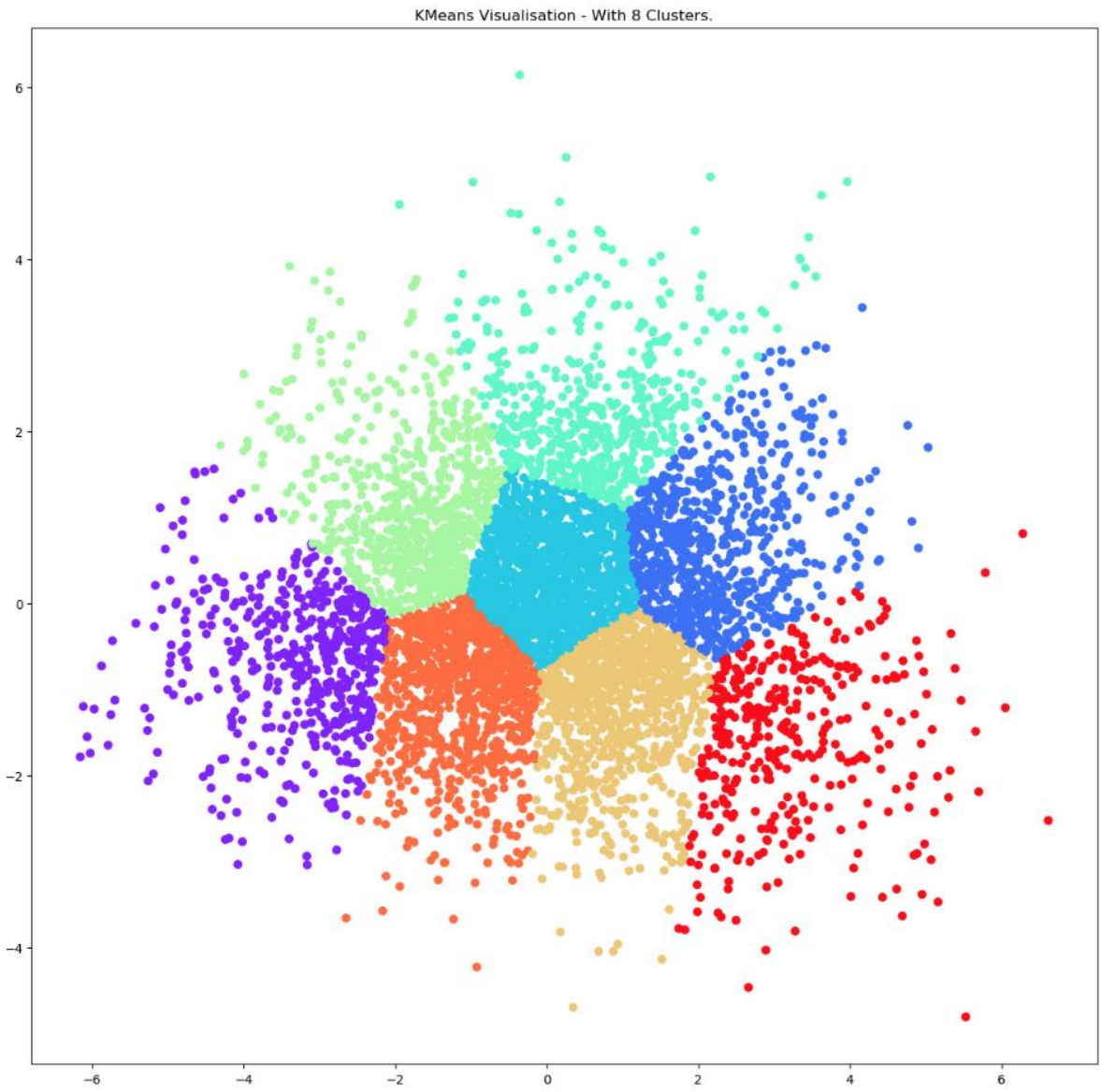

Figure 5.2 Scatter plot of K-Means clustering with K set to 8.

In order to improve our model scores we'll conduct the elbow method to try and uncover an optimum number of K clusters. Using the principle components, for each K value, we initialise K-means and use the inertia attribute to identify the sum of squares. So, as K increases, the sum of squared distance tends toward zero, meaning the distortion between clusters reduces. We are looking for the point at which this distortion is at its highest, signifying the last notable shift in cluster centroids and hence the optimum value for K.

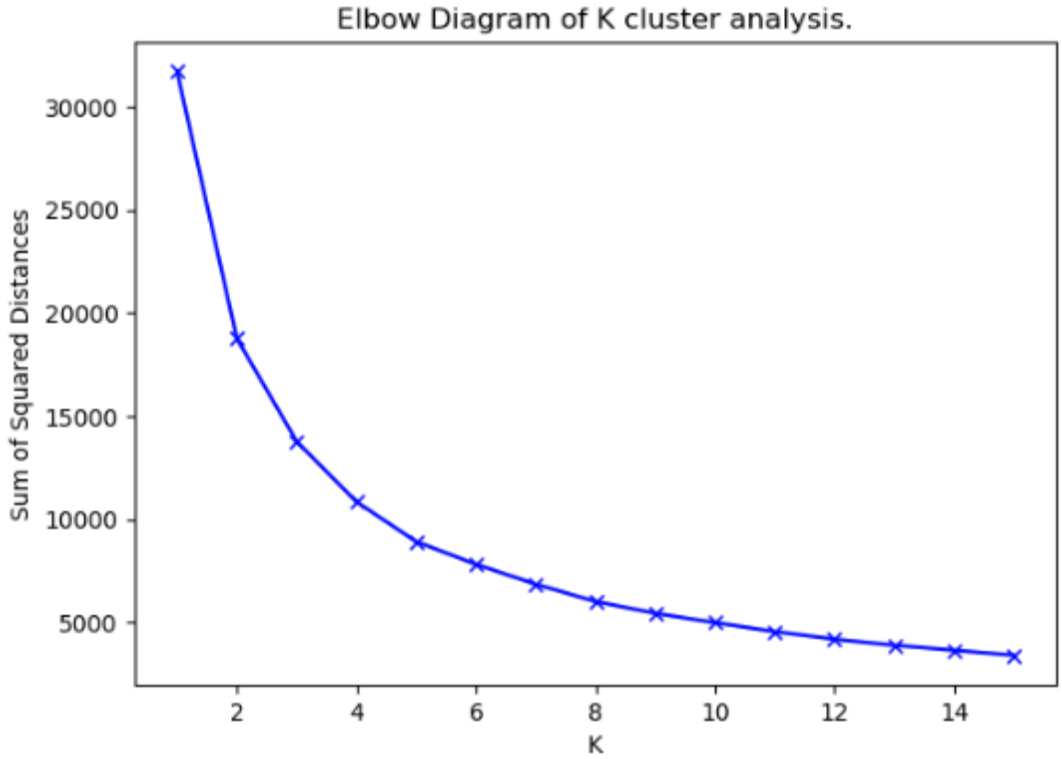

Figure 5.2 Elbow diagram of K cluster / Sum of squared distance ratio.

The elbow diagram is one way of demonstrating the optimum number of clusters by which to set our algorithm. From the graph above I'd surmise that the bend appears its greatest when K is equal to 5. With this in mind I ran another K-means model with 5 clusters which produced the following scores.

```
K-means scoring after implemeting with n_clusters = 5.

Completeness Score is 0.09964872748981951.
Homogeneity Score is 0.3386565963476357.
```

Table 5.2 K-means scoring after setting cluster to an optimum value of 5.

As table 5.2 demonstrates, we have significantly increased our completeness score while mildly decreasing homogeneity. After multiple experimentations with a number of different clusters and maximum iterations, the analysis of the elbow diagram was confirmed as correct. Figure 5.3 below shows the clustering of data when set to this optimum level. We can see clearly defined boundaries between five central clusters.

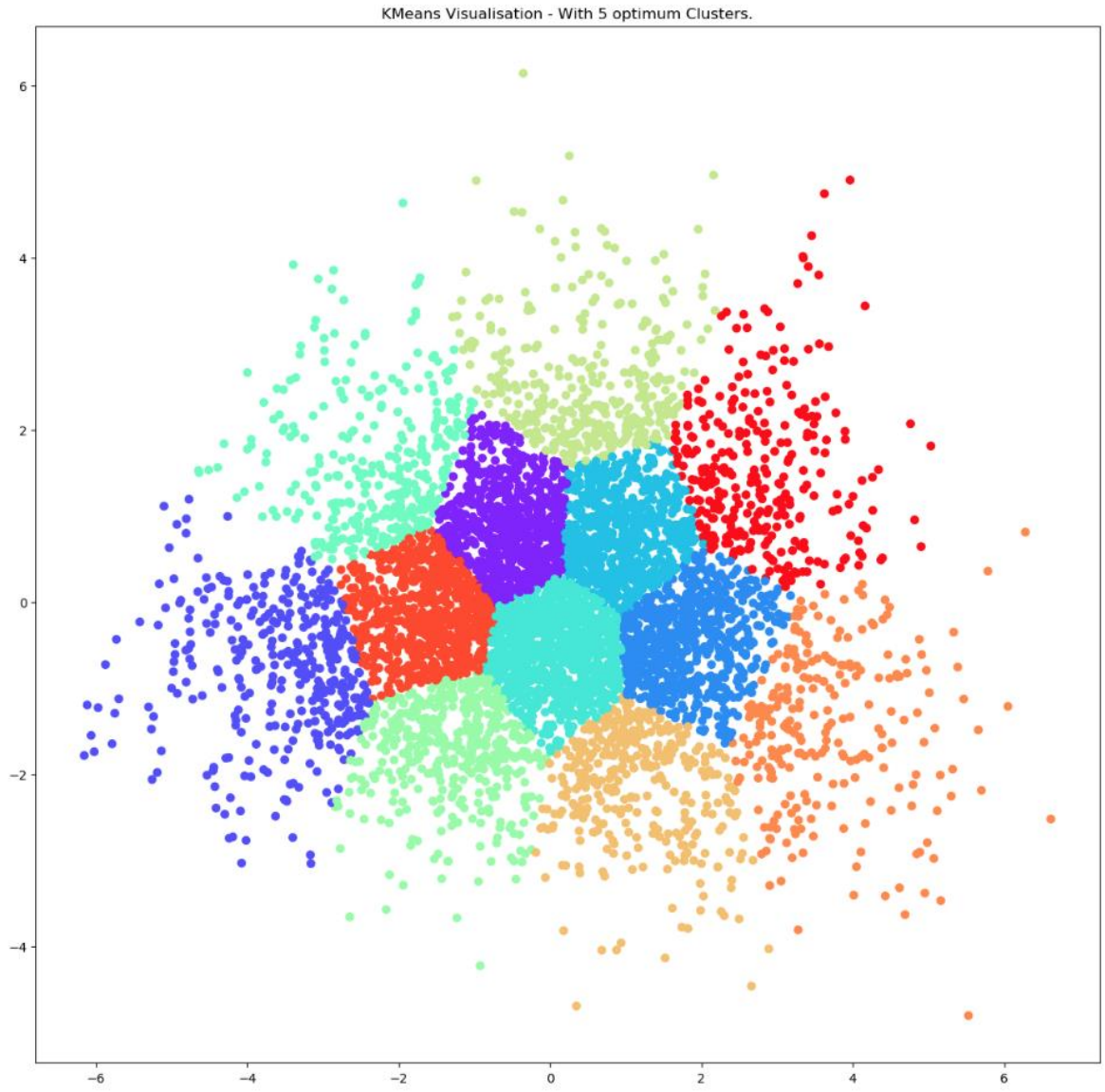

Figure 5.3 Cluster visualisation after setting an optimum number of K clusters.

By setting our number of clusters to an optimum value of 5, when visualised we can see clearer groupings at the centre. This, together with our somewhat improved scores, indicates that decreasing the value of K resulted in more accurate clustering of our data within the reduced 16-dimensional space.

The resulting analysis could be used to automatically shift users of a dating app into similar segments; however, its effectiveness could be questioned by the low homogeneity score.

Chapter Six: Supervised method – Support Vector Machines (SVM)

My supervised method aims to answer a specific problem,

“Can we make suggestive matches by classifying new users as matching with candidates in our current user base?”

To help answer this challenge I will be employing SVM (Support Vector Machines) - a binary linear classifier - making it ideal to classify users into ‘Match’ or ‘No Match’.

SVM is given training examples, each marked as belonging to one of two categories, and then builds a model such that new examples are predicted as belonging to one of the categories. The aim of SVM is to maximise the gap that separates the classes, so when new examples are mapped onto the same space they are predicated as belonging to one of the two classes depending on which side of the gap they fall. In SVM, using the right hyperplanes to maximise this margin between classes will optimise the model, and the vector points that touch the margin are known as ‘support vectors’ (hence the name Support Vector Machines).

Having already scaled my data into two principle components, I can use these as my target variables. Likewise, as in figure 4.2, we can use our *heatmap_value* column ‘match’ as our label. The final task before fitting the model is to split the data into training and testing. For this initial model I am implementing a 70 / 30 train / test split, with a view to experimentation with this ratio to optimise the results.

Confusion Matrix for default parameters:

```
[[1483   64]
 [ 138  169]]
```

Classification Report:

	precision	recall	f1-score	support
0	0.91	0.96	0.94	1547
1	0.73	0.55	0.63	307
micro avg	0.89	0.89	0.89	1854
macro avg	0.82	0.75	0.78	1854
weighted avg	0.88	0.89	0.88	1854

Table 6.1 Confusion matrix and classification report using default parameters.

After fitting our SVM and training against 70% of our data, I ran the test portion against the model. Table 6.1 shows our classification report and confusion matrix, demonstrating that the model performed relatively well. The figures above indicate a correctly classified 'No Match' 91% of the time with 1483 true negative results, and a correct classification of 'Match' 73% of the time with 169 true positive results.

SVM benefits from parameter experimentation to achieve the optimum settings for the model. This can be achieved using a gridsearch (a dictionary with the key's set to the name of the parameter and the corresponding values as input). Gridsearch then runs the same loop with cross-validation to find the best parameter combination. It will build a single model using the best parameters, however, after conducting a gridsearch the change to my model was nominal, so I opted not to include it in this report.

SVM was a successful method to make suggestive matches to users.

Chapter Seven: Reflection of analysis

In terms of the goals of my analysis, using PCA in conjunction with supervised and unsupervised methods demonstrated some potential to enhance dating applications.

The summary statistics allowed us to examine the slight bias in the dataset, as well as confirm that the age range of participants was acceptable. The heatmap was a key component to indicate potential variables for further examination.

PCA (Principle Component Analysis) allowed me to reduce the dimensions of data and get a better visual overview, as well as providing focused, optimised variables with which to build models.

The K-means approach was questionable, as although it performed well when grouping data (having a high completeness score), the low homogeneity would offset these gains. As an algorithm for enhancing dating applications I don't feel this would be suitable given our data.

The supervised learning method performed well overall, with a high degree of both true positives and true negative results. This appears a suitable solution to create a suggestive matching system based on this relatively small dataset.

The data itself could be improved, with a larger spread of races across participants and more accurate ratings in the "you prefer" columns.

In conclusion, I felt the techniques outlined above had some potential to enhance user experience. SVM was a suitable solution to make suggestive matches, however K-means struggled at grouping users together, so I would urge further research and experimentation to achieve a more precise approximation of user clouds.

A: Software Version, data and included packages.

Software versions

Python version: 3.7.3

Anaconda version: 4.7.12

Ipython version: 7.4.0

Ipython genutils version: 0.2.0

Data

Dataset was downloaded from openml.org, csv file included with report.

https://www.openml.org/d/40536

Included packages

```
import numpy as np
import pandas as pd
import re, random
import matplotlib.pyplot as plt
from pylab import rcParams
import sklearn
import seaborn as sns
from sklearn import cluster
from sklearn import metrics
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale, MinMaxScaler, StandardScaler
from sklearn import preprocessing
from sklearn.cluster import KMeans
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import GridSearchCV
from sklearn import neighbors
pd.set_option('display.max_columns', 500)
```

Works Cited

Harrison, O., 2018. *towarddatascience.com.* [Online]
Available at: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761
[Accessed 30 Septmber 2019].
J.Clement, 2019. *www.statista.com.* [Online]
Available at: https://www.statista.com/statistics/814698/share-of-us-internet-users-who-use-tinder-by-age/
[Accessed 31 October 2019].
VanderPlas, J., 2016. *jakevdp.github.io.* [Online]
Available at: https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html
[Accessed 16 October 2019].
VanderPlas, J., 2016. *jakevdp.github.io.* [Online]
Available at: https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html
[Accessed 16 October 2019].