Sophia Genetics Data Engineering Task

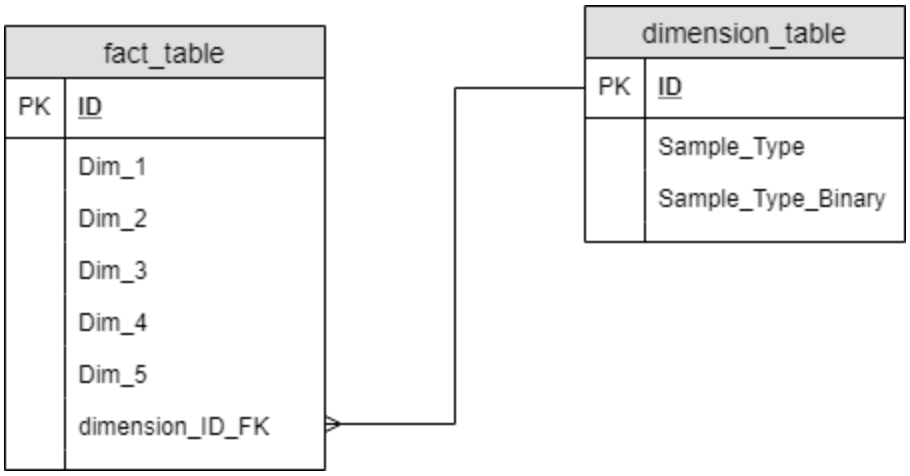
Kieron Ellis

Database design

The entity relationship model I used for this task is based on the star schema containing one fact table and one dimensions table as seen below. The star schema groups data into fact and dimension tables. Fact tables hold quantitative/measurable data and dimension tables store qualitative attributes related to the facts table.

This schema is widely used for dimensional modelling because they are denormalized which allows for simpler queries and business logic, as well as query performance gains, particularly for metrics analysis; hence my decision to use it for this task.

Alternatively, a normalized schema, such as the snowflake schema, would reduce the required storage space and data redundancy as well as increasing data integrity. However, this comes at the cost of performance, particularly for metrics analysis.



Dimension table

The dimension table contains 3 fields: sample type binary, sample type, and an auto incrementing integer primary key field called ID. This table contains all of the unique combinations of sample type and sample type binary values present in the raw data table.

Fact table

The fact table contains the 5 principle component fields (dim 1, dim 2, … , dim 5), an auto incrementing integer primary key field called ID as well as a foreign key pointing to the ID field of the dimensions table called dimension_ID_FK.

The sample field from the raw data was dropped in favour of the auto incrementing integer primary key field. The reason for this is that keeping the sample field as a string format (e.g. “Sample 1”) increases the file size of the table and will slow down both writing to and reading from the table.

There is a one-to-many relationship between dimension_table.ID and fact_table.dimension_ID_FK, i.e. one record in the dimension table can appear multiple times in the fact table. The dimension table includes only unique combinations of dimensions (i.e. sample type and sample type binary) found in the raw data table to reduce the storage space required for this table.

Indexing

I have made the assumption that objective of storing this data in a database is for fast select queries rather than write queries, and that these tables will need to be dropped and recreated as additional data is added. Because of this I have added indexes to each column in the database.

In general this indexing approach is not recommended because it increases time to write to the database and also the storage space required.

As the database is queried and more understanding of its use cases become known, more selective indexes can be used.

Deployment and ETL

Due to the limited direction given in the background and explanation of the dataset as well as the expectations, assumptions have to be made on setting up the database structure and designing an ETL pipeline to populate it.

I have made several assumptions:

1. One such assumption I have made is due to the lack of timestamp/batch number/etc fields in the dataset, as well as the expectation that the dataset will increase to 10 million rows and additional dimensions of label assignments, I have assumed that this data is a subset of a single analysis.

2. The full dataset when it is provided will replace the subset of data.
3. The names of all quantitative fields (e.g. the Dim 1 PCA field) will always be called Dim followed by a number.

With this in mind, I have created an automated ETL pipeline without hardcoded SQL which will sort all quantitative fields into the facts table and all qualitative fields in to the dimensions table without requiring further tuning.

The pipeline is built using Python but queries a PostgreSQL database for all data operations, i.e. the raw data is not stored by Python in Pandas/a Numpy array at any stage of the ETL process.

The raw dataset is first ingested into a table called raw_data_table, which could be used to store it in a data lake. This raw_data_table is then queried to create the fact_table and dimension_table.

In order to query this data to return all of the fact and dimension fields the fact_table and dimension_table must be queried with a left outer join SQL query where the dimension_ID_FK matches the dimension_table ID. An example of such a query is shown below

```
SELECT *
FROM fact_table
LEFT JOIN dimension_table
ON fact_table.dimension_ID_FK = dimension_table.id
```

Data Analysis

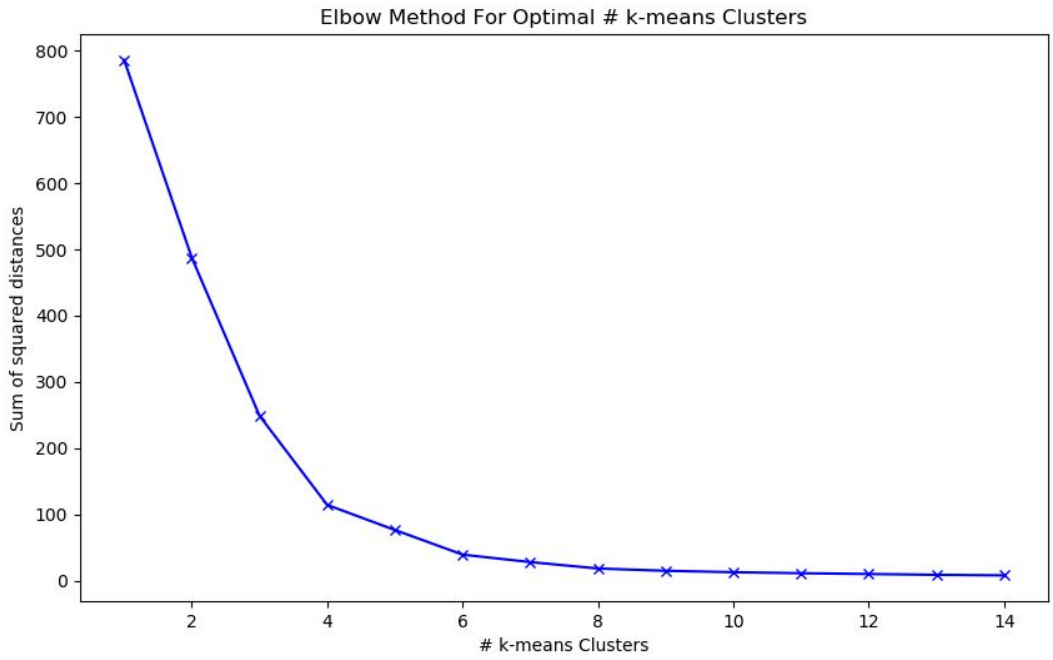
In order to estimate the number of clusters the quantitative data may contain I used the k-means algorithm and the elbow method to determine optimal number of clusters.

K-means is an unsupervised clustering algorithm which accepts a user defined number of clusters (k) and a numeric dataset as its inputs. The algorithm first defines k random centroids and then iteratively assigns each datum to the closest centroid before re-calculating each centroid based on the average location of the data assigned to it. This process repeats until the data assigned to clusters does not change, or another stopping condition is met such as the maximum number of iterations being reached or the variance not improving by a certain amount.

The elbow method involves looking at the sum of squared distances (SSD) from each datum to its assigned centroid for increasing numbers of k (the input parameter number of clusters). As k increases there are diminishing returns in the SSD. The optimal value of k is then the point at which this curve levels out.

In order to give equal importance to all variables, for this task they were scaled using sklearn's MinMaxScaler.

The graph below shows the elbow method applied to this data set. I selected the optimal number of clusters as 6.



Alternative methods to determine optimal number of clusters include silhouette analysis. This involves plotting silhouette coefficients on top of the data which measure how far a sample is away from the decision boundary of neighbouring clusters.

Data Visualisation

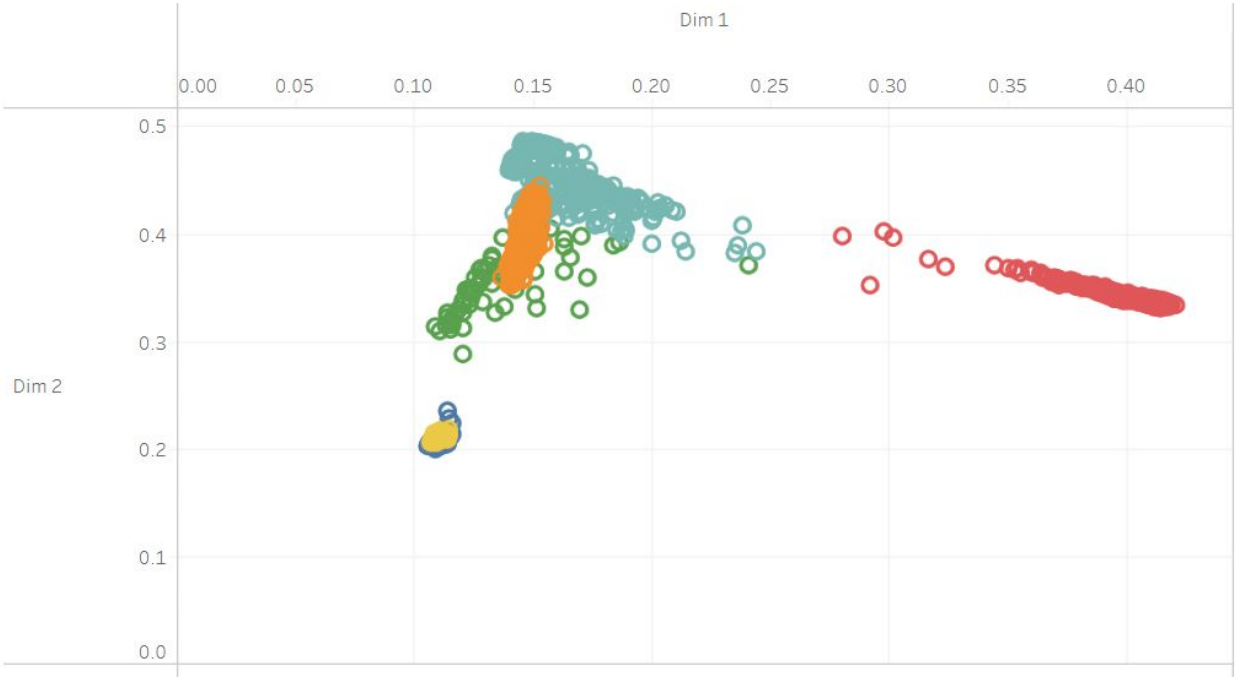
The interactive webpage can be seen at https://public.tableau.com/profile/kieron4186#!/vizhome/SophiaGeneticsInteractiveDataAnalysis/GeneticsData

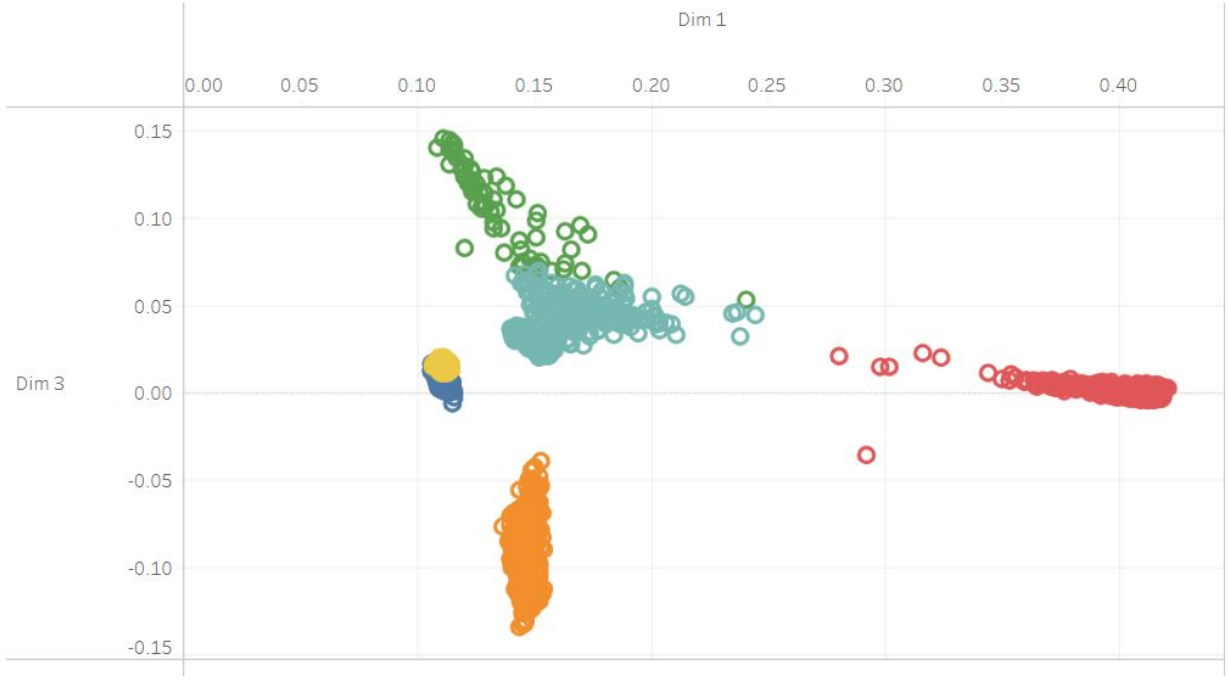
The dashboard includes 3 main sections:

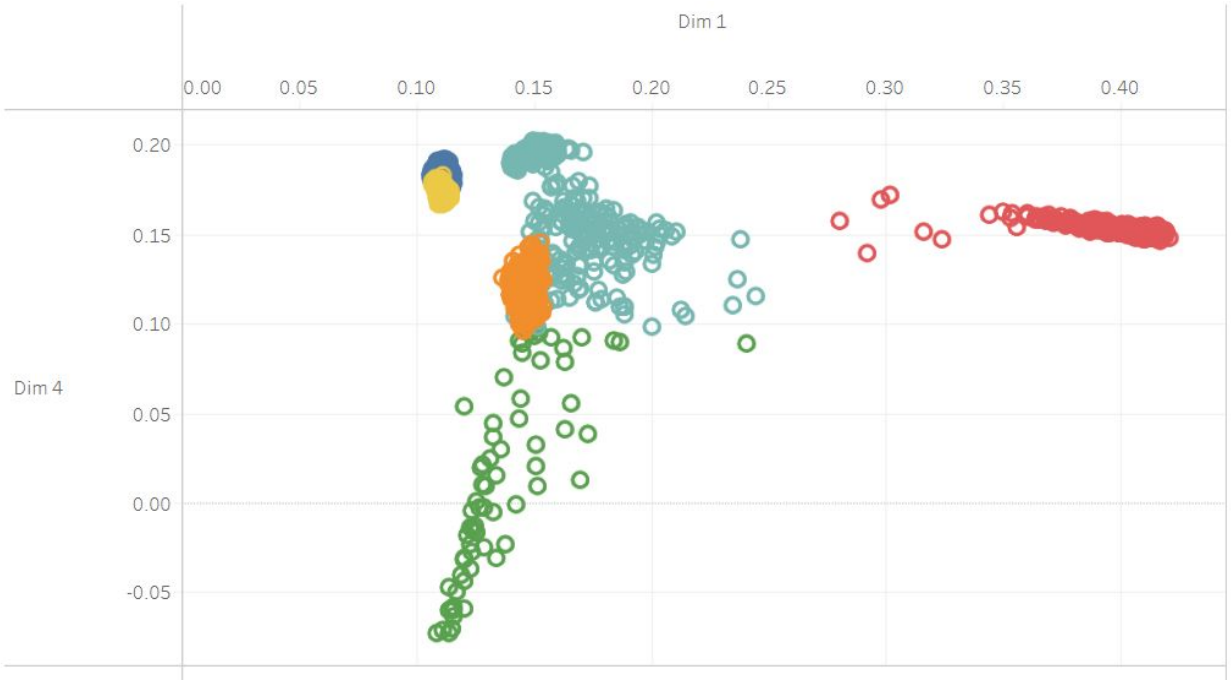
1. A count of the number of records in the dataset
2. A count of the number of good and bad samples per sample type
3. An interactive scatter chart where the x-axis, y-axis and colour can be selected by the user for self-service analysis.

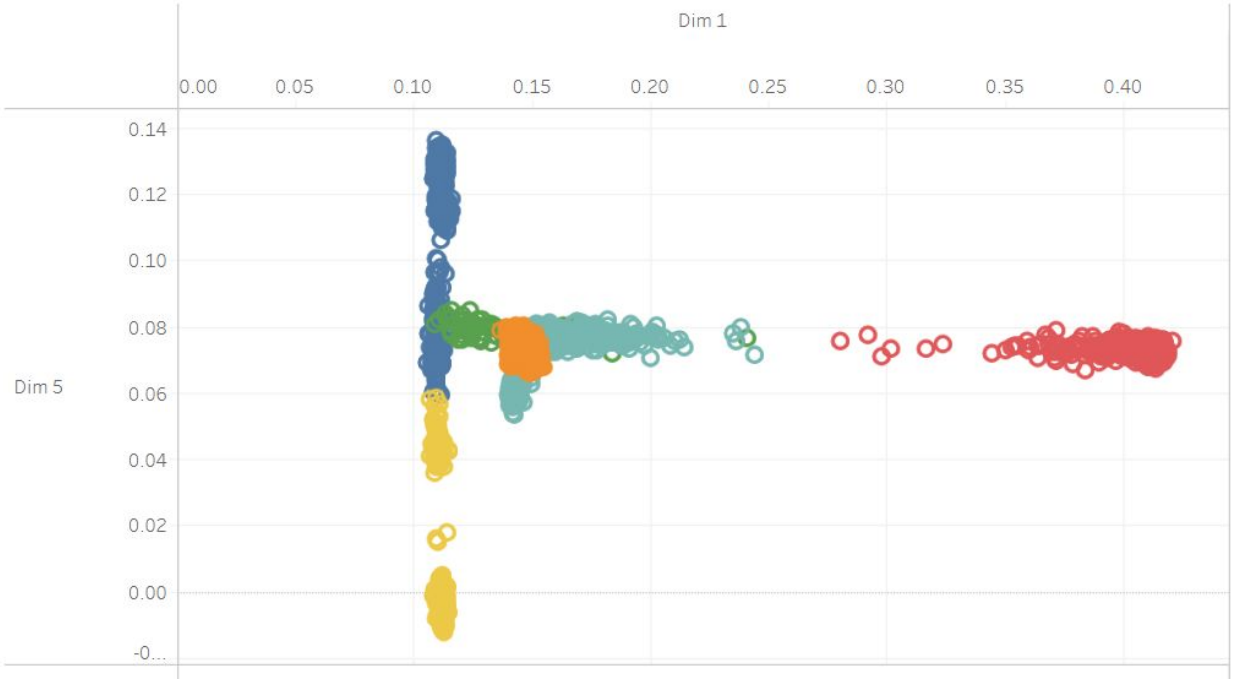
The scatter chart is initially coloured by the k-means clusters from the Data Analysis above. As can be seen below, this algorithm with 6 clusters as its input did a good job of segmenting the dataset into distinct clusters.

The plots below show Dim 2, Dim 3 Dim 4 and Dim 5 against Dim 1 respectively and are all coloured by the same 6 k-means cluster assignments.






Dim 1
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Dim 4
0.20
0.15
0.10
0.05
0.00
-0.05


Dim 1
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Dim 5
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
-0...