Configuration Manual

MSc Research Project
MSc in Data Analytics

Samradni Ranganath Bharadwaj
Student ID: X22214801

School of Computing
National College of Ireland

Supervisor: Dr. Catherine Mulwa

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Samradni Ranganath Bharadwaj
Student ID:	X22214801
Programme:	MSc in Data Analytics
Year:	2023-2024
Module:	MSc Research Project
Lecturer:	Dr. Catherine Mulwa
Submission Due Date:	12th August 2024
Project Title:	Detecting the Genes that have High Probability of Causing Kawasaki Disease
Word Count:	661
Page Count:	5

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Samradni Ranganath Bharadwaj

Date: 12th August 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Samradni Ranganath Bharadwaj
Student ID: X22214801

1 Introduction

This confirguration manual for ‘Detecting Genes that have High Probability of Causing Kawasaki Disease’. This is a guide on the system requirements, establishing libraries and running the code. The information is dedicated to the setting-up procedures and code steps required to develop a predictive model.

2 System Specification

Model Name	Asus Zenbook 14
Operating System	Microsoft Windows 11 Home Single Language
Version	10.0.22631 Build 22631
Processor	Intel® Core™ i5-1035G1 CPU @ 1.00GHz, 1190 Mhz, 4 Core(s), 8 Logical Processor
Storage	475 GB
RAM	8 GB

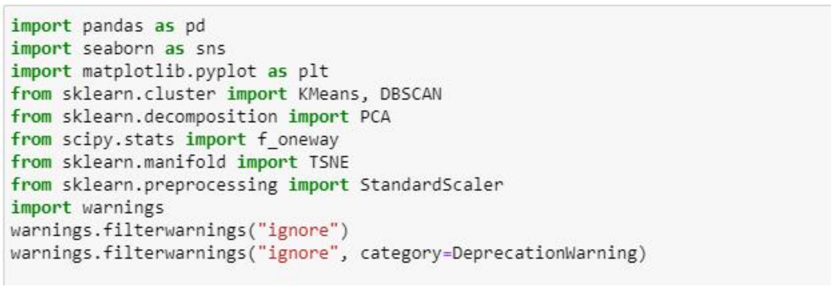
Table 1: Specifications of Systems

3 Software Used

In this project, python was the main the programming language. The python version that was used in building this project was 3.11.5 version. The python was accesed from the Anaconda Jupyter Environment. It is the updated version, therefore has new and updated libraries in it. The anaconda notebook’s version is 23.7.4 and the jupyter noterbook’s version is 6.5.4. This setup is updated and there are no errors while running the code.

4 Research Project Developement

4.1 Importing Libraries

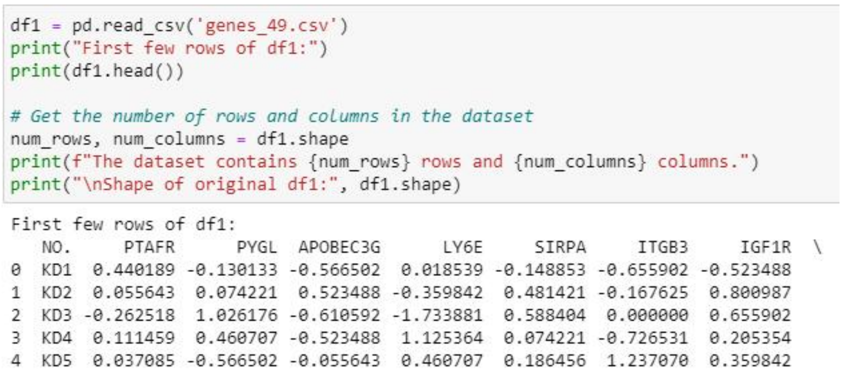


```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans, DBSCAN
from sklearn.decomposition import PCA
from scipy.stats import f_oneway
from sklearn.manifold import TSNE
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings("ignore")
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

Figure 1: Importing Libraries

Various python libraries shown in Figure 1 are necessary and were imported to support the steps that will be performed to achieve the solution for the research questions. As python is the principal language mainly because of the availability of numerous libraries to handle data like Pandas for efficient handling of complex data, Matplotlib and Seaborn for visualization ,sklearn.cluster for the clustering algorithms like K Means Clustering and DBSCAN that are implemented. sklearn.decomposition for implementing Principal Component Analysis, sklearn.manifold for t-SNE and sklearn.preprocessing for StandardScalar to standardize the features.

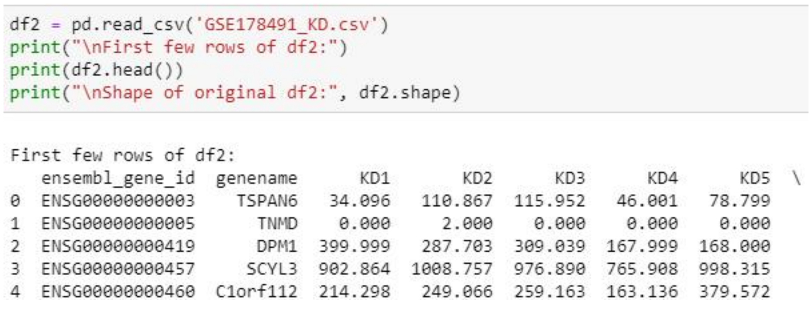
4.2 Uploading Files



```python
df1 = pd.read_csv('genes_49.csv')
print("First few rows of df1:")
print(df1.head())

# Get the number of rows and columns in the dataset
num_rows, num_columns = df1.shape
print(f"The dataset contains {num_rows} rows and {num_columns} columns.")
print("\nShape of original df1:", df1.shape)
```

```
First few rows of df1:
   NO.     PTAFR      PYGL  APOBEC3G      LY6E     SIRPA     ITGB3     IGF1R  \
0  KD1  0.440189 -0.130133 -0.566502  0.018539 -0.148853 -0.655902 -0.523488
1  KD2  0.055643  0.074221  0.523488 -0.359842  0.481421 -0.167625  0.800987
2  KD3 -0.262518  1.026176 -0.610592 -1.733881  0.588404  0.000000  0.655902
3  KD4  0.111459  0.460707 -0.523488  1.125364  0.074221 -0.726531  0.205354
4  KD5  0.037085 -0.566502 -0.055643  0.460707  0.186456  1.237070  0.359842
```

Figure 2: Loading df1 file



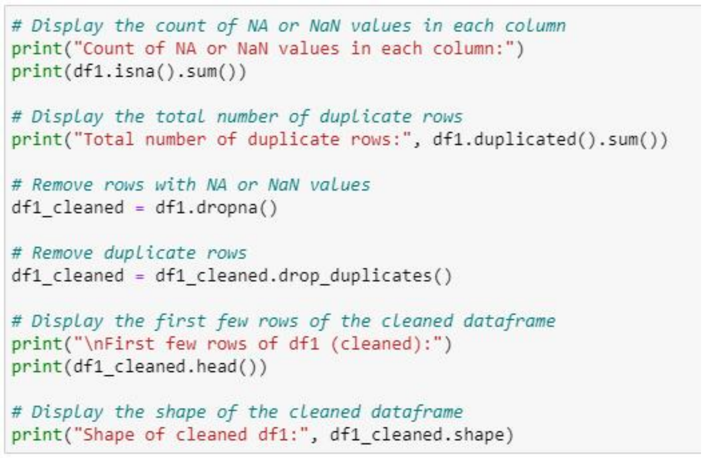
```python
df2 = pd.read_csv('GSE178491_KD.csv')
print("\nFirst few rows of df2:")
print(df2.head())
print("\nShape of original df2:", df2.shape)
```

```
First few rows of df2:
   ensembl_gene_id  genename      KD1       KD2      KD3      KD4      KD5  \
0  ENSG00000000003    TSPAN6   34.096   110.867  115.952   46.001   78.799
1  ENSG00000000005      TNMD    0.000     2.000    0.000    0.000    0.000
2  ENSG00000000419      DPM1  399.999   287.703  309.039  167.999  168.000
3  ENSG00000000457     SCYL3  902.864  1008.757  976.890  765.908  998.315
4  ENSG00000000460  C1orf112  214.298   249.066  259.163  163.136  379.572
```

Figure 3: Loading df2 file

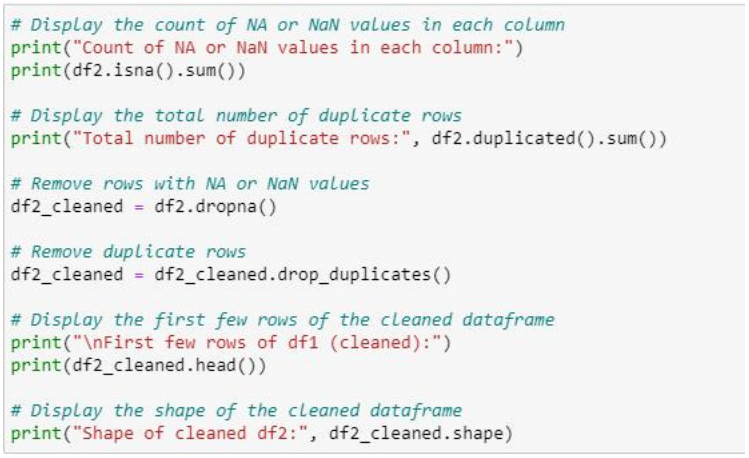
As shown in Figure 2 and Figure 3, the files were imported in jupyter to perform future steps. After importing these files, the shape for the dataset was checked by checking the first 5 rows and columns.

4.3 Data Cleaning



```python
# Display the count of NA or NaN values in each column
print("Count of NA or NaN values in each column:")
print(df1.isna().sum())

# Display the total number of duplicate rows
print("Total number of duplicate rows:", df1.duplicated().sum())

# Remove rows with NA or NaN values
df1_cleaned = df1.dropna()

# Remove duplicate rows
df1_cleaned = df1_cleaned.drop_duplicates()

# Display the first few rows of the cleaned dataframe
print("\nFirst few rows of df1 (cleaned):")
print(df1_cleaned.head())

# Display the shape of the cleaned dataframe
print("Shape of cleaned df1:", df1_cleaned.shape)
```

Figure 4: Data Cleaning of DF1

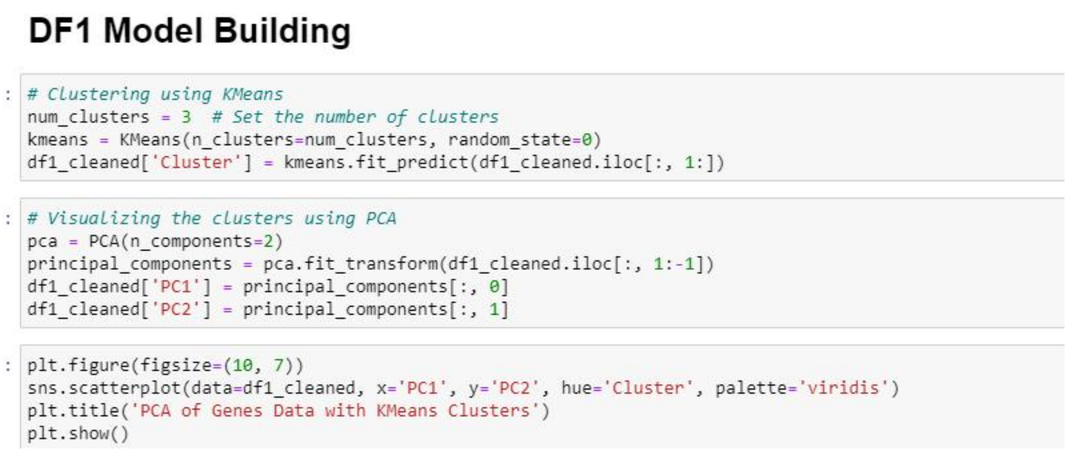


```python
# Display the count of NA or NaN values in each column
print("Count of NA or NaN values in each column:")
print(df2.isna().sum())

# Display the total number of duplicate rows
print("Total number of duplicate rows:", df2.duplicated().sum())

# Remove rows with NA or NaN values
df2_cleaned = df2.dropna()

# Remove duplicate rows
df2_cleaned = df2_cleaned.drop_duplicates()

# Display the first few rows of the cleaned dataframe
print("\nFirst few rows of df1 (cleaned):")
print(df2_cleaned.head())

# Display the shape of the cleaned dataframe
print("Shape of cleaned df2:", df2_cleaned.shape)
```

Figure 5: Data Cleaning of DF2

The Figure 4 and Figure 5 depict the data cleaning steps. Intitally they were checked for missing and duplicate values and then the missing and duplicate values were removed to ensure that the data is clean and accurate to make predictions on. After cleaning the dataset, they were again check and for this the first 5 rows were chosen to see if the data is clean and then the shape of the dataset was checked.
After performing these steps various visualizations were performed to understand the data.

4.4 Modelling



DF1 Model Building

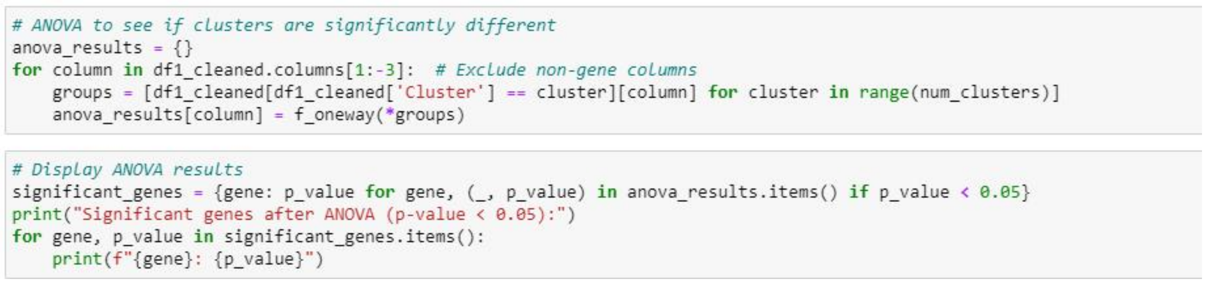
```
# Clustering using KMeans
num_clusters = 3  # Set the number of clusters
kmeans = KMeans(n_clusters=num_clusters, random_state=0)
df1_cleaned['Cluster'] = kmeans.fit_predict(df1_cleaned.iloc[:, 1:])
```

```
# Visualizing the clusters using PCA
pca = PCA(n_components=2)
principal_components = pca.fit_transform(df1_cleaned.iloc[:, 1:-1])
df1_cleaned['PC1'] = principal_components[:, 0]
df1_cleaned['PC2'] = principal_components[:, 1]
```

```
plt.figure(figsize=(10, 7))
sns.scatterplot(data=df1_cleaned, x='PC1', y='PC2', hue='Cluster', palette='viridis')
plt.title('PCA of Genes Data with KMeans Clusters')
plt.show()
```

Figure 6: Model Implementation on df1

The Figure 6 shows that KMeans Clustering was implemented on df1 where the data was divided in 3 clusters. Later PCA was implemented to reduce the dimensions.

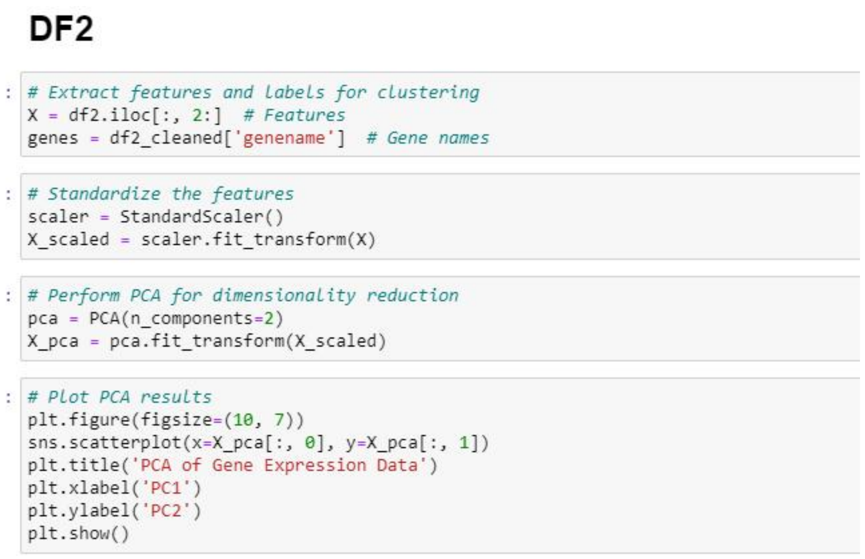


```
# ANOVA to see if clusters are significantly different
anova_results = {}
for column in df1_cleaned.columns[1:-3]:  # Exclude non-gene columns
    groups = [df1_cleaned[df1_cleaned['Cluster'] == cluster][column] for cluster in range(num_clusters)]
    anova_results[column] = f_oneway(*groups)
```

```
# Display ANOVA results
significant_genes = {gene: p_value for gene, (_, p_value) in anova_results.items() if p_value < 0.05}
print("Significant genes after ANOVA (p-value < 0.05):")
for gene, p_value in significant_genes.items():
    print(f"{gene}: {p_value}")
```

Figure 7: Anova on df1

The Figure 7 depicts the implementation of Anova on df1 to check if there is statistical significant difference between the data.



DF2

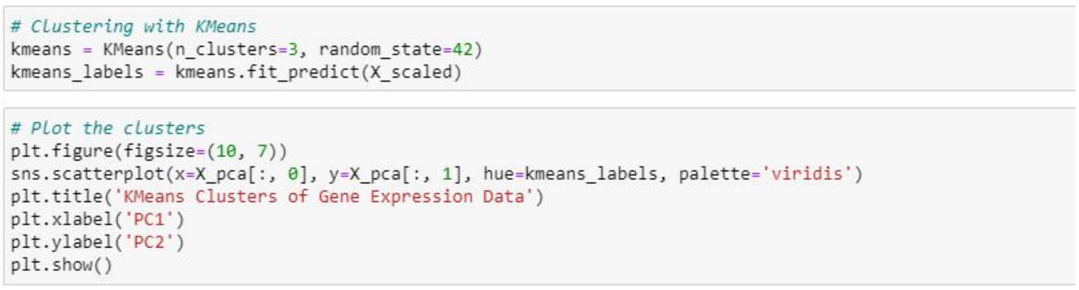
```
# Extract features and labels for clustering
X = df2.iloc[:, 2:]  # Features
genes = df2_cleaned['genename']  # Gene names
```

```
# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
# Perform PCA for dimensionality reduction
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
```

```
# Plot PCA results
plt.figure(figsize=(10, 7))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1])
plt.title('PCA of Gene Expression Data')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.show()
```

Figure 8: Feature extract for clustering on df2


```
# Clustering with KMeans
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans_labels = kmeans.fit_predict(X_scaled)
```

```
# Plot the clusters
plt.figure(figsize=(10, 7))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=kmeans_labels, palette='viridis')
plt.title('KMeans Clusters of Gene Expression Data')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.show()
```

Figure 9: KMeans implemented on df2

KMeans was implemented on df2 to make the clusters of the data and Anova was implemented to check the statistical difference in the data.

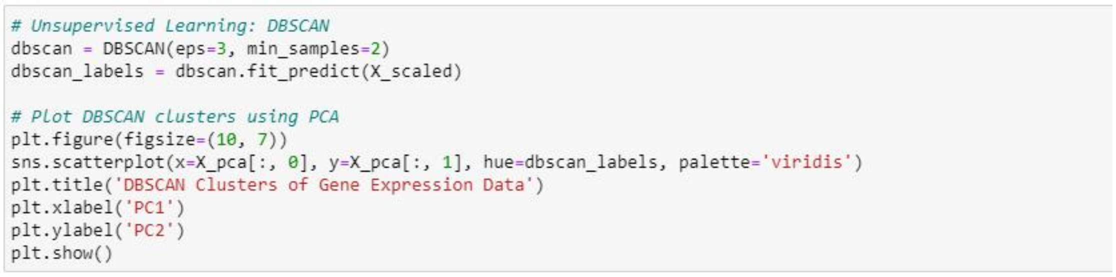

```
# Unsupervised Learning: DBSCAN
dbscan = DBSCAN(eps=3, min_samples=2)
dbscan_labels = dbscan.fit_predict(X_scaled)

# Plot DBSCAN clusters using PCA
plt.figure(figsize=(10, 7))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=dbscan_labels, palette='viridis')
plt.title('DBSCAN Clusters of Gene Expression Data')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.show()
```

Figure 10: Implementation of DBSCAN

On df2, multiple models and techniques were implemented and DBSCAN is one of them. DBSCAN seperates the low density data and that is why it is an ideal fit for the dataset as it has dense gene data and to trace the noise data is difficult. DBSCAN does that work perfectly. t-SNE is implemented to visually depict the noise data as compare to good data.

References

Gupta, P. and Bagchi, A., 2024. Introduction to Pandas. In Essentials of Python for Artificial Intelligence and Machine Learning (pp. 161-196). Cham: Springer Nature Switzerland.