Configuration Manual

Identifying At-Risk Students in Virtual Learning Environment using Clustering Techniques

MSc Research Project
MSc in Data Analytics

Kamalesh Palani
Student ID: x18180311

School of Computing
National College of Ireland

Supervisor:
Dr. Paul Stynes
Dr. Pramod Pathak

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Kamalesh Palani		
Student ID:	x18180311		
Programme:	MSc in Data Analytics	**Year:**	2019-2020
Module:	MSc in Research Project		
Lecturer:	Dr. Paul Stynes, Dr. Pramod Pathak		
Submission Due Date:	17th August 2020		
Project Title:	Identifying At-risk Students in Virtual Learning Environment using Clustering Techniques		
Word Count:	**930**	**Page Count: 10**	

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: Kamalesh Palani

Date: 17th August 2020

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

Kamalesh Palani
Student ID: x18180311

1 Introduction

This manual contains the step wise information of the research conducted in identifying the at-risk students using clustering technique. By following the steps and procedures in this document the research project can be completely reproduced. This report also contains information of environmental step-up and system requirements of the conducted research.

2 System Specification

2.1 Hardware Configuration

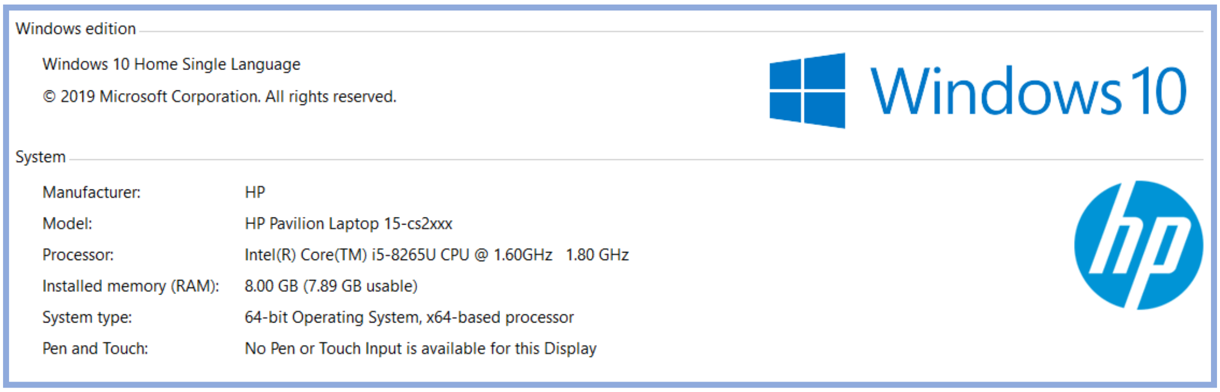

Figure 1. Hardware specification

Figure 1 shows the hardware specification used in this research work for the project implementation.

2.2 Software Configuration

In this research implementation part is conducted using python programming language of version 3.7.4. To use this programming language Anaconda for windows version has to be installed[1] . 64-bit graphical installer for windows version is used in this research work which is shown in figure 2.

[1] https://www.anaconda.com/

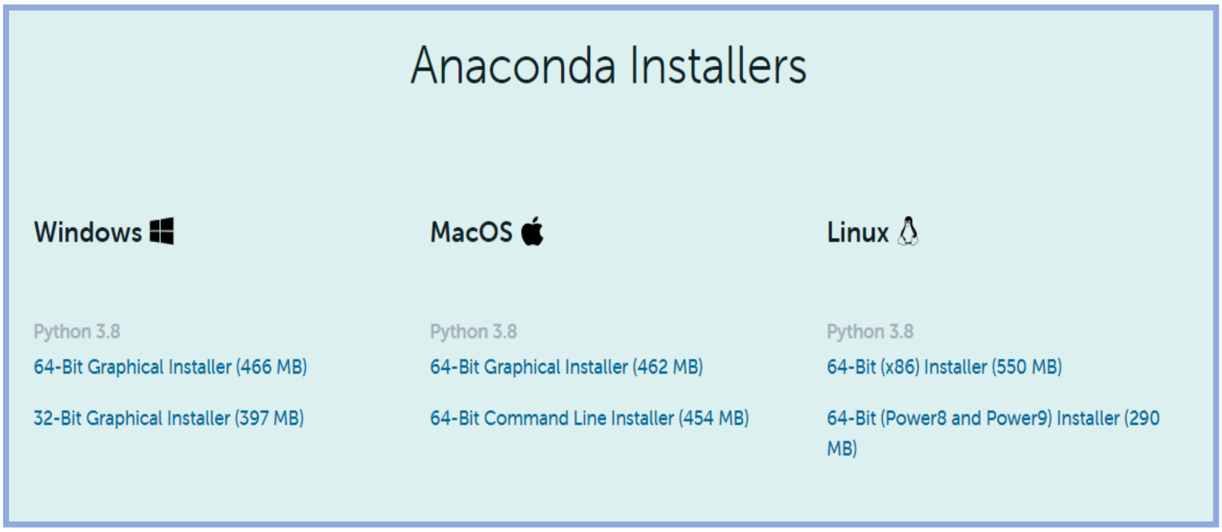

Figure 2. Anaconda Installer

After installation of the anaconda software anaconda navigator will display different Integrated Development Environment (IDE) in which Jupyter notebook of version 6.0.1 is used in this research which is shown in figure 3.

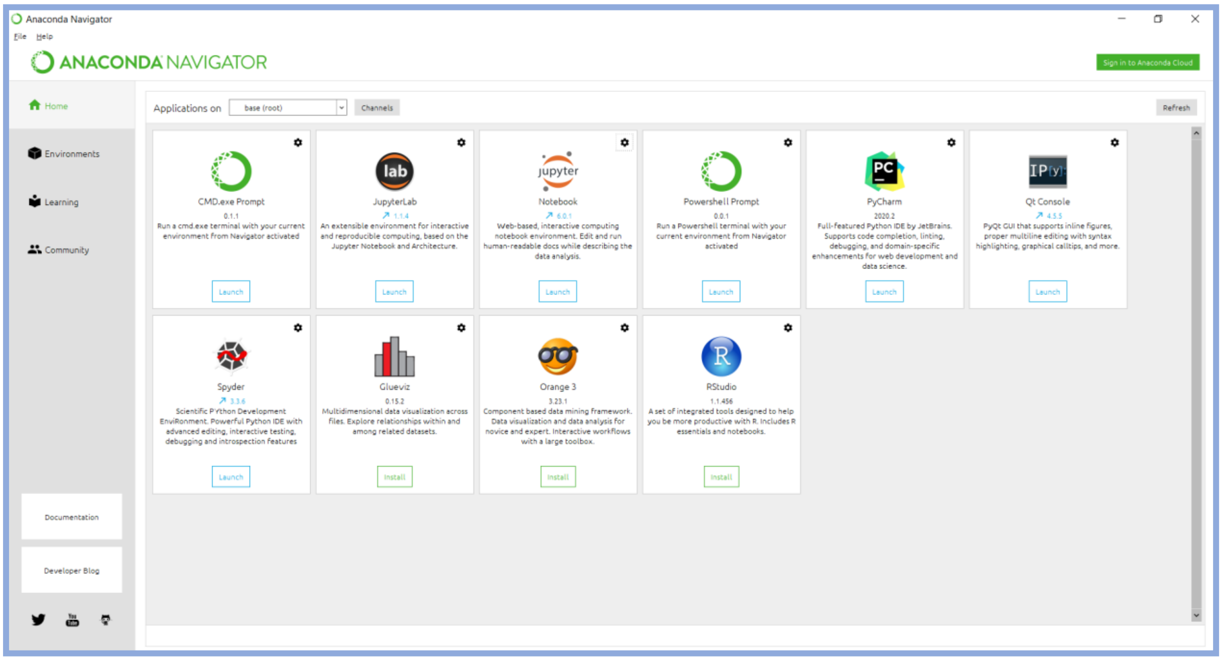

Figure 3. Anaconda Navigator

After downloading the anaconda, python libraries related to the projects has to be imported. To import the libraries into the Jupyter notebook IDE. Anaconda Powershell Prompt is opened by searching it in windows search bar. And pip install command and the name of the below mentioned libraries is used to import the python libraries package to the IDE.

- Matplotlib-version 3.1.1
- Seaborn-version 0.9.0

- Scikit-learn- version 0.21.3
- Pandas -version 0.23.4
- Numpy-version 1.16.5
- Plotly-version 4.2.1
- Scipy-version 1.4.1
-

3 Implementation of the Models

After the installation of the software to implement the project below steps can be performed to reproduce the clustering models and replicate the project result used in this research.

3.1 Data Source

For this research dataset is downloaded from the Open University [2].Which is a publicly available dataset and it is downloaded as a zip file. After unzipping the folder 7 different files related to student's interaction with virtual learning environment, student's academic performance and student information are present in the files.

3.2 Import of Libraries

After downloading the dataset in the local machine jupyter notebook is launched from the anaconda navigator prompt. And, New drag down button is clicked then python 3 is chosen to open a new notebook to implement the project which is shown is figure 4.

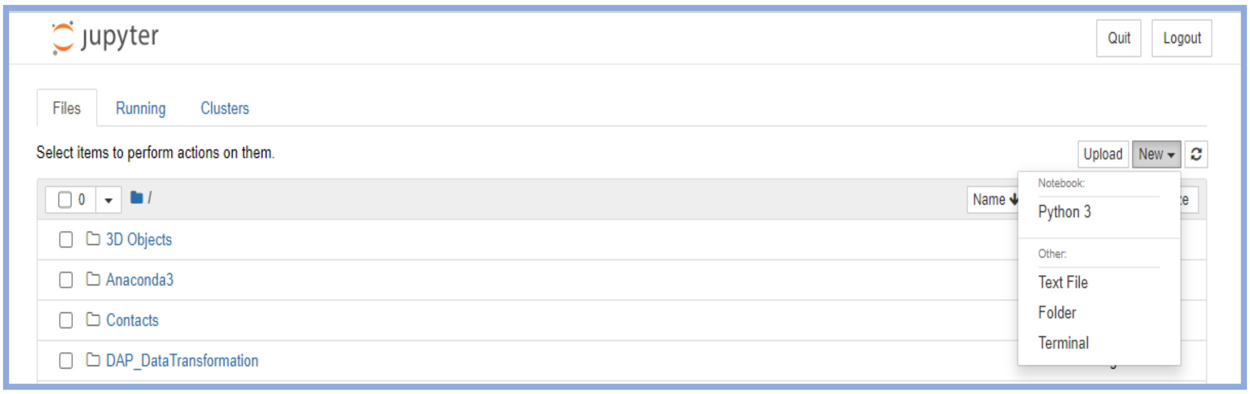

Figure 4. Homepage

Figure 5 shows the libraries that is used in this research project.

[2] https://analyse.kmi.open.ac.uk/

```
#importing Libraries
import numpy as np
import pandas as pd
import time
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import missingno as msno
import sklearn.metrics as metrics
import pyclustertend
import plotly
import plotly.graph_objects as go
import scipy.cluster.hierarchy as shc
import scipy.cluster.hierarchy as shc

# ML libraries
from sklearn.metrics import accuracy_score
from sklearn.mixture import GaussianMixture
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm.libsvm import predict_proba
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import davies_bouldin_score
from sklearn.metrics import roc_curve
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import normalize
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import normalize
from sklearn.decomposition import PCA
from sklearn.datasets import make_classification
from sklearn.cluster import AffinityPropagation
from kmodes.kmodes import KModes
from kmodes.kprototypes import KPrototypes
from fcmeans import FCM
from sklearn import datasets
from pyclustertend import hopkins
from sklearn.preprocessing import scale
from mpl_toolkits import mplot3d
from  plotly.offline import plot
from sklearn.decomposition import PCA
from scipy.cluster.hierarchy import linkage,dendrogram
from sklearn.cluster import KMeans
```

Figure 5. libraries

3.3 Data Pre-processing

After importing the libraries and dataset in to the Jupyter book, next thing is to process the data. In this research for processing the data three different attributes are mainly extracted from the 7 different files by merging the files using the primary key column which is students ID in all the files. The screenshot of three attributes creation code snippet is given below.

```
Student Learning Behaviour Attributes

# merging StudentVLE data on vle

vle_details = pd.merge(student_vle, vle, how = 'left',  left_on = ['code_module', 'code_presentation', 'id_site'],
                                        right_on = ['code_module', 'code_presentation', 'id_site'])


# Removing the negative date column from the VLE dataset

columns = ['date']
filter_ = (vle_details[columns] > 0).all(axis=1)
vle_filtered_data=vle_details[filter_]

# Concating the studentid,course and year as primary key column

vle_filtered_data['concats'] = vle_filtered_data.id_student.astype(str).str.cat
                        (vle_filtered_data[['code_presentation', 'code_module']])


# Removing the column week_from and week_to
vle_filtered_data.drop(vle_filtered_data.columns[[7,8]], axis=1, inplace=True)


# Aggregating the column for each site  and clicks

vle_agg_site_data  = vle_filtered_data.groupby(['id_site', 'activity_type']).agg({'sum_click': ['mean', 'sum']})
vle_agg_site_data.reset_index(level= [0,1], inplace=True)
vle_agg_site_data.columns = ['id_site', 'activity_type', 'mean_clicks', 'sum_clicks']


# Aggregating the column for each site based on student id and clicks

vle_agg_data  = vle_filtered_data[['id_student','activity_type','concats','sum_click']].groupby
                          (['concats', 'activity_type','id_student']).agg({'sum_click': ['mean', 'sum']})
vle_agg_data = vle_agg_data.reset_index()
vle_agg_data.columns = ['concat','activity_type', 'id_student','mean_clicks', 'sum_clicks']
```

Figure 6. Student learning behaviour attributes pre-processing part-1

Figure 6 shows the dropping of columns, aggregating of clicks for each site and for each student.

```
# Agrregating the column for each course and year


vle_agg_course_data  = vle_filtered_data[['concats','sum_click']].groupby(['concats']).agg({'sum_click': ['mean', 'sum']})
vle_agg_course_data = vle_agg_course_data.reset_index()
vle_agg_course_data.columns  = ['concat','mean_clicks','sum_clicks']

### Aggregating the column for each Students weekly from the VLE

# Aggregating the clicks for each student for a single day
avg_clicks = vle_filtered_data.groupby(['concats','date']).agg({"sum_click":"mean"}).rename
                                        (columns={'sum_click':'average_clicks'}).reset_index()
avg_clicks.head(10)


# changing the date type to string
weekly_clicks = avg_clicks.astype({'date':'str'})
weekly_clicks.info()

# Replacing the date with week
weekly_clicks.replace(to_replace=["1","2","3","4","5","6","7","8","9","10","11","12","13","14","15","16","17","18","19","20"]
           value= ["W1","W1","W1","W1","W1","W1","W1","W2","W2","W2","W2","W2","W2","W2","W3","W3","W3","W3","W3","W3","W3"]
           inplace=True)

weekly_clicks = weekly_clicks.groupby(['concats','date']).agg({"average_clicks":"mean"}).
                          rename(columns={'average_clicks':'average_clicks_weekly'}).reset_index()

# Melting the week columns to rows

df_unmelted = weekly_clicks.pivot(index='concats', columns='date')
df_unmelted = df_unmelted['average_clicks_weekly'].reset_index()
df_unmelted.columns.name = None
df_unmelted = df_unmelted.replace(np.nan, 0)
```

Figure 7. Student learning behaviour attributes pre-processing part-2

Figure 7 shows the week wise clickstream aggregation for each student and unmelt function which is used to get the original data frame and pivot function is used to convert the week wise columns to rows.

Student Performance Attributes

```
## weight for each course is maintained same

assessments.loc[(assessments.code_module=='CCC') &(assessments.assessment_type=='Exam'),'weight'] = \
assessments.loc[(assessments.code_module=='CCC') &(assessments.assessment_type=='Exam'),'weight']/2
assessments.loc[(assessments.code_module=='GGG') & (assessments.assessment_type=='TMA'),'weight']=(100/3)
assessments.groupby(['code_module','code_presentation']).agg({'weight': ['sum']})
```

```
# Calculation of the marks by merging 2 tables to have assignment scores and weights together in one table.

# Join Assessment and StudentAssessment tables
joined=pd.merge(student_assessment,assessments,on='id_assessment',how='left')

# Calculate weighted scores for all assessments of all students
joined['score*weight']=(joined['score']*joined['weight'])

# Sum up score*weights and divide by total weights (There are some students has total weight higher or much lower than %100)
# for all students of all modules to calculate final mark.
marks=joined.groupby(['id_student','code_module','code_presentation'],as_index=False)['score*weight','weight'].sum()

marks['total_mark'] = marks['score*weight']/marks['weight']
marks["mark"]  = marks['score*weight']/200
marks.rename(columns = {'score*weight': 'total_weight', 'weight': 'attempted_weight'}, inplace=True)
marks = marks.round(1)


#merging the data to the student info table
joined = pd.merge(marks,student_infocopy,on=['id_student','code_module','code_presentation'],how='left')

# stduent withdrawn form the courses are marked as NAN in the mark
joined.loc[joined.final_result=='Withdrawn','mark']= np.nan
joined.loc[joined.final_result=='Withdrawn','total_mark']= np.nan
joined.head(10)
```

Figure 8. Student performance attributes pre-processing

Figure 8 shows normalization of weights for all the courses and new column creation namely total mark, mark, attempted weights for each student are created.

Student Demographic Attributes

```
### extracting the studnet_info table and creating the unique key column using stduents ID,Module,year

student_infocopy = student_info.copy()
student_infocopy['concat'] = student_infocopy.id_student.astype(str).str.cat(student_infocopy[['code_presentation',
                                                                                 'code_module']])

Xfactors = student_infocopy[["gender", "region", "highest_education", "imd_band", "age_band",
                                                  "num_of_prev_attempts", "studied_credits", "disability"]]
X_noncat = pd.get_dummies(Xfactors)

X_noncat.head(5)
```

Figure 9. Student demographic attributes pre-processing

Figure 9 show the one hot encoding is done using the dummies function and primary column is created using the group by and cat function.

Merging the student course information,clicks and scores to single table

```
maintable = pd.merge(aggragated_score,vle_agg_course_data,on=['concat'],how='left')
maintable_copy = maintable.copy()
maintable_copy.head(10)
```

Figure 10. Merging of columns

In the above block three different attributes are merged using left out join function in pandas data frame and single aggregated dataset has been used to build the clustering model.

3.4 Data Modelling

In this section steps taken to implement the multiple clustering models and methods used to find the number of clusters in this research is discussed below. Implementation screenshot of the process followed is given below.

Gap Statistics

```
# Normalizing the data
scaler = MinMaxScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(model), columns=model.columns)

df_scaled_Hierarchical = df_scaled.copy()
df_scaled_gaussian = df_scaled.copy()
df_scaled_k_prototype = df_scaled.copy()
df_scaled_fuzzyc_means = df_scaled.copy()

# Gap Statistics method to determine the optimal clusters
Error =[]
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i).fit(df_scaled)
    kmeans.fit(df_scaled)
    Error.append(kmeans.inertia_)
import matplotlib.pyplot as plt
plt.figure(figsize=(5, 5))
plt.plot(range(1, 11), Error)
plt.title('Gap Statistics')
plt.xlabel('Number of Clusters')
plt.ylabel('Error')
plt.show()
```

Figure 11. Gap Statistics

Figure 11 shows the minmax function which is used to normalize the data before giving as an input to the gap statistics method. And to determine the number of clusters for the data gap statistics approach is used.

```
### Gaussian mixture clustering model

def doGMM(X, nclust=3):
    model = GaussianMixture(n_components=nclust,init_params='kmeans')
    model.fit(X)
    clust_labels3 = model.predict(X)
    return (clust_labels3)

clust_labels3 = doGMM(df_scaled_gaussian,3)
gaus_cluster = pd.DataFrame(clust_labels3)
df_scaled_gaussian.insert((df_scaled_gaussian.shape[1]),'gaus_cluster',gaus_cluster)



#plot data with seaborn
facet = sns.lmplot(data=df_scaled_gaussian, x='mark', y='sum_clicks', hue='gaus_cluster',
                   fit_reg=False, legend=True, legend_out=True)
```

Figure 12. Gaussian Mixture model

Figure 12 shows the implementation of the gaussian model and the parameters used to run the model. Also, after running the model the dispersion of the data points formed as clusters is visualized using seaborn libraries in python.

```
## k-prototype Clustering algorithm

# define the model
model_array = df_scaled.values
kproto = KPrototypes(n_clusters= 3 , verbose =2,max_iter=5)
# fit the model
clusters=kproto.fit_predict(model_array,categorical=[4])

cluster_dict=[]
for c in clusters:
    cluster_dict.append(c)
df_scaled_k_prototype['clusters']=cluster_dict

# visualization of data points
fig = plt.figure(figsize=(5, 4))
ax = fig.add_subplot(111)

scatter_Kprototype=ax.scatter(df_scaled.iloc[:,3],df_scaled.iloc[:,5],c=clust,cmap='rainbow',s=50)
ax.set_title('k-prototype Clustering')
ax.set_xlabel('Marks')
ax.set_ylabel('Clicks')
plt.colorbar(scatter_Kprototype)
```

Figure 13.K-Prototpye model

Figure 13 shows the code snippet of k-prototype model and the parameters used to improve the accuracy of the model.

```
### Hierarchical Clustering

# dendogram
plt.figure(figsize=(10, 7))
plt.title("Dendrograms")
dend = shc.dendrogram(shc.linkage(df_scaled_Hierarchical, method='ward'))


def doAgglomerative(X, nclust=3):
    model = AgglomerativeClustering(n_clusters=nclust, affinity = 'euclidean', linkage = 'ward')
    clust_labels1 = model.fit_predict(X)
    return (clust_labels1)


clust_labels1 = doAgglomerative(df_scaled_Hierarchical, 3)
agglomerative = pd.DataFrame(clust_labels1)
df_scaled_Hierarchical.insert((df_scaled_Hierarchical.shape[1]),'agglomerative',agglomerative)

#plot data with seaborn
facet = sns.lmplot(data=df_scaled_Hierarchical, x='mark', y='sum_clicks', hue='agglomerative',
                   fit_reg=False, legend=True, legend_out=True)
```

Figure 14. Hierarchical Clustering model

Figure 14 shows the code snippet of hierarchical clustering and visualization used in the implementation of the models.

3.5 Evaluation of Clustering Models

In this research multiple models performance is compared to find the best performing model using clustering evaluation metric. And evaluation metric is used to find the better separation of clusters between the data points and also to check the better-defined clusters. Shown below are the code snippet of evaluation metric.

```
## Gaussian mixture Metric
import sklearn.metrics as metrics

print("### Gaussian mixture Metric ###\n")
#ground truth label are not known
Gaussian_mixture_sil=metrics.silhouette_score(df_scaled, gaus_cluster[0], metric='euclidean')
print("silhouette_score: ",Gaussian_mixture_sil)

#ground truth label are not known
Gaussian_mixture_cal=metrics.calinski_harabasz_score(df_scaled,gaus_cluster[0])
print("calinski_harabasz_score: ",Gaussian_mixture_cal)

#ground truth label are not known
Gaussian_mixture_dav = davies_bouldin_score(df_scaled, gaus_cluster[0])
print("davies_bouldin_score: ",Gaussian_mixture_dav)
```

Figure 15. Gaussian Mixture evaluation metric

```
## Hierarchical Evaluation metric

print("### Hierarchical Evaluation Metric ###\n")
#ground truth label are not known
hier_sil=metrics.silhouette_score(df_scaled, agglomerative[0], metric='euclidean')
print("silhouette_score: ",hier_sil)


#ground truth label are not known
hier_cal=metrics.calinski_harabasz_score(df_scaled, agglomerative[0])
print("calinski_harabasz_score: ",hier_cal)

#ground truth label are not known
hier_dav = davies_bouldin_score(df_scaled, agglomerative[0])
print("davies_bouldin_score: ",hier_dav)
```

Figure 16. Hierarchical evaluation metric

```
## K-prototype Evaluation Metric

print("### K-prototype Evaluation Metric ###\n")
#ground truth label are not known
kpro_sil=metrics.silhouette_score(df_scaled_k_prototype, cluster_dict, metric='euclidean')
print("silhouette_score: ",kpro_sil)

#ground truth label are not known
kpro_cal=metrics.calinski_harabasz_score(df_scaled_k_prototype, cluster_dict)
print("calinski_harabasz_score: ",kpro_cal)

#ground truth label are not known
kpro_dav = davies_bouldin_score(df_scaled_k_prototype, cluster_dict)
print("davies_bouldin_score: ",kpro_dav)
```

Figure 17. K-Prototype evaluation metric

3.6 Visualization

To interpret the data points between the clusters PyLab library is used from python which bulk imports both the Matplotlib and NumPy libraries. Multiple markers have been used for the visualization to find the dispersion of the clusters. Figure 18 shows the code snippet of the visualization used in this research.

Data Visualization

```
import pylab as pl

fig = pl.figure(figsize=(5, 4))
pl = fig.add_subplot(111)


for i in range(0, df_scaled_k_prototype.shape[0]):
    if df_scaled_k_prototype.clusters[i] == 0:
        c1 = pl.scatter(df_scaled_k_prototype.iloc[i,2],df_scaled_k_prototype.iloc[i,5],c='r',
        marker='+')
    elif df_scaled_k_prototype.clusters[i] == 1:
        c2 = pl.scatter(df_scaled_k_prototype.iloc[i,2],df_scaled_k_prototype.iloc[i,5],c='g',
        marker='o')
    elif df_scaled_k_prototype.clusters[i] == 2:
        c3 = pl.scatter(df_scaled_k_prototype.iloc[i,2],df_scaled_k_prototype.iloc[i,5],c='b',
        marker='*')

ax=pl.legend([c1, c2, c3], ['cluster 0', 'cluster 1',
    'cluster 2'])

pl.set_xlabel('Marks')
pl.set_ylabel('Clicks')
```

Figure 18. Visualization