



**DTSA 5510 - UNSUPERVISED ALGORITHMS IN MACHINE
LEARNING**

UNIVERSITY OF COLORADO, BOULDER

MASTER OF SCIENCE IN DATA SCIENCE

**Unsupervised Clustering | Kaggle | Final
Project**

1 Introduction

Description

In this challenge, we are given a dataset where each row belongs to a particular cluster. Our job is to predict the cluster each row belongs to. We are not given any training data, and we are not told how many clusters are found in the ground truth labels.

Evaluation

Submissions are evaluated on the Adjusted Rand Index between the ground truth cluster labels of the data and our predicted cluster labels. We are not given the number of ground truth clusters or any training labels. This is a completely unsupervised problem.

Data Structure

data.csv contains rows of data with columns id and features f_00, f_01, ..., to f_28 containing either integers or floats.

2 Exploratory Data Analysis(EDA)

2.1 Data Loading and Libraries Import

```
import numpy as np
import pandas as pd
import seaborn as sns
sns.set(style='darkgrid', font_scale=1.4)
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler,
    PowerTransformer
from sklearn.mixture import GaussianMixture,
    BayesianGaussianMixture

data=pd.read_csv('../input/tabular-playground-series-jul-2022/
    data.csv', index_col='id')
```

2.2 EDA and Preprocessing

We can see that there are not any missing values or any duplicates in the dataset. We have 98000 rows with 29 features (i.e. f_00 to f_28), 7 of which are discrete values and the rest are continuous(floating numbers). :

```
# Shape and preview
print('Dataframe shape:', data.shape)
data.head()

print('MISSING VALUES:')
print(data.isna().sum().sum())
print(f'Duplicates in dataset: {data.duplicated().sum()}')
data.dtypes
```

```
# Figure with subplots
fig=plt.figure(figsize=(15,14))

for i in range(7):
    # New subplot
    plt.subplot(4,2,i+1)
    feat_num=i+7
    sns.countplot(x=data.iloc[:,feat_num])

    # Aesthetics
    plt.title(f'Feature: 0{feat_num}')
    plt.xlim([-1,44])          # same scale for all plots
    plt.ylim([0,11000])       # same scale for all plots
    plt.xticks(np.arange(0,44,2))
    plt.xlabel('')

# Overall aesthetics
fig.suptitle('Discrete feature distributions', size=20)
fig.tight_layout() # Improves appearance a bit
plt.show()

# Continuous features
cont_feats=[f'f_0{i}' for i in range(7)]
cont_feats=cont_feats + [f'f_{i}' for i in range(14,29)]

# Figure with subplots
```

```

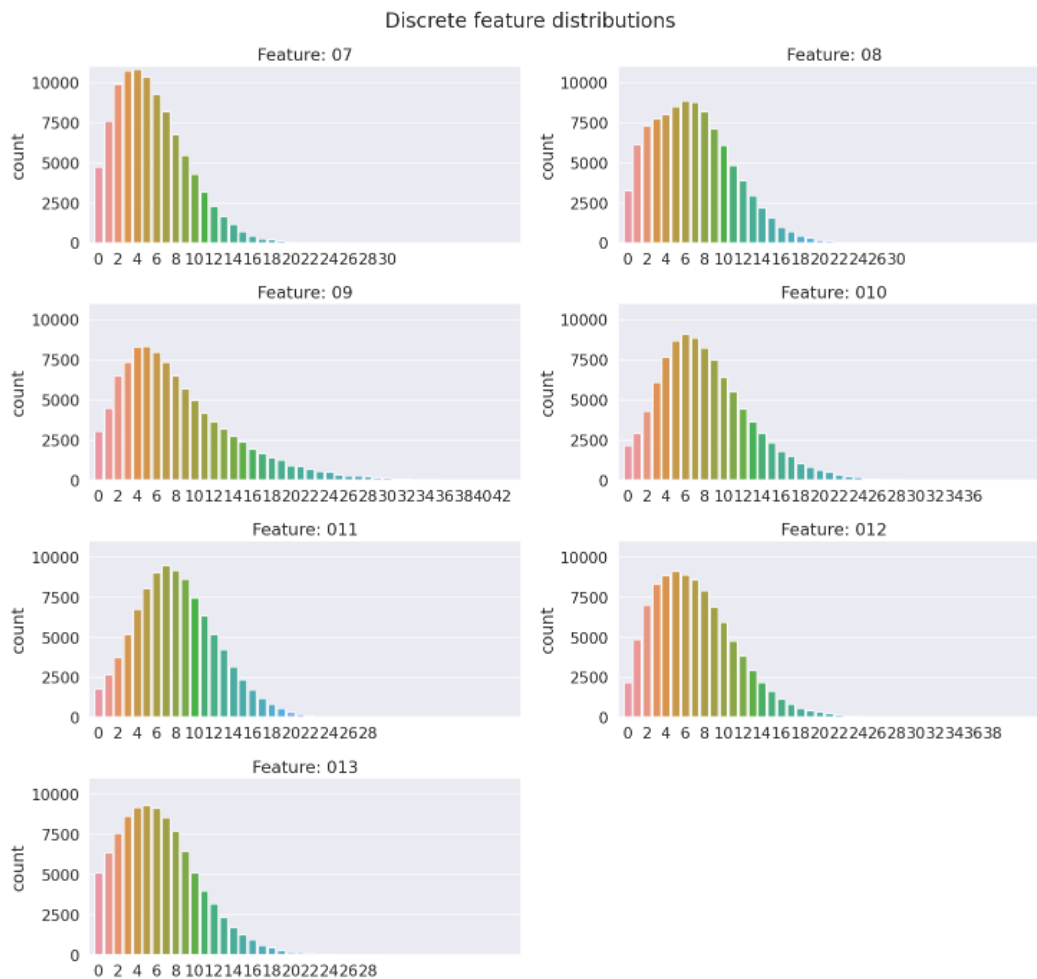
fig=plt.figure(figsize=(15,14))

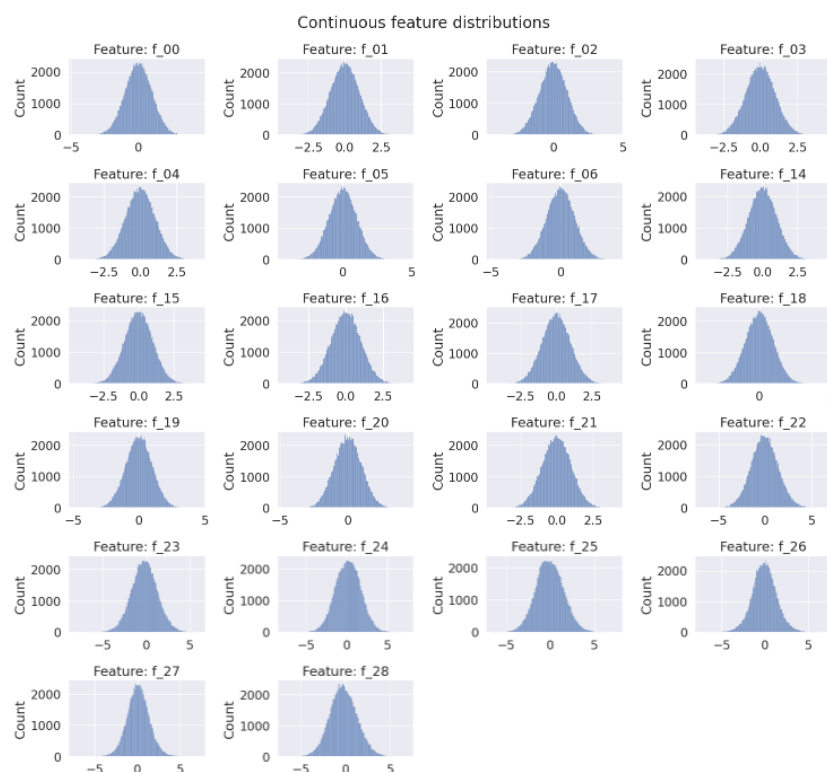
for i, f in enumerate(cont_feats):
    # New subplot
    plt.subplot(6,4,i+1)
    sns.histplot(x=data[f])

    # Aesthetics
    plt.title(f'Feature: {f}')
    plt.xlabel('')

# Overall aesthetics
fig.suptitle('Continuous feature distributions', size=20)
fig.tight_layout() # Improves appearance a bit
plt.show()

```





3 More Preprocessing, Model Selection and Training

First, we start by plotting the inertia in reference to the number of clusters, a process known as "Elbow method". It's hard at first sight to decide where the elbow is happening but with a bit trial and error seems like $k=7$ can be a good starting point. There are lots of efforts regarding some sort of standardization for all the columns to help with the performance of clustering. We can check whether these columns are coming from a normal, Poisson, or other distributions but for the sake of simplicity, here we are only applying power transform to those columns.[2-3] For models, I decided to try K-means, Gaussian mixture model and Bayesian Gaussian mixture models and compare the results. We can simply use the `fit_predict` on the scaled data to perform clustering. Now, one point to consider before moving on is to plot the position of the center of each cluster to visualise how well each feature is able to separate the different clusters[4]. This shows that our features f_00 to f_06 and f_14 to f_21 don't separate the clusters so we can try to drop these features and attempt to train our models again.

```
inertias = []
for k in range(1,30):
    km = KMeans(n_clusters=k)
    km.fit(data.iloc[:10000,:])
```

```
inertias.append(km.inertia_)

# Plot inertias
plt.figure(figsize=(16,6))
plt.plot(range(1,30), inertias, 'bx-')
plt.xlabel('Number of clusters, k')
plt.ylabel('Inertia')
plt.title('Elbow method')
plt.show()

scaled_data = pd.DataFrame(PowerTransformer().fit_transform(
    data))

scaled_data.columns = data.columns

# Models

# K-Means

model_km = KMeans(n_clusters=7, random_state=0)
preds_km = model_km.fit_predict(scaled_data)

# Gaussian Mixture Model

model_gmm = GaussianMixture(n_components=7, random_state=0)
preds_gmm = model_gmm.fit_predict(scaled_data)

# Code from AmbrosM: https://www.kaggle.com/competitions/
    tabular-playground-series-jul-2022/discussion/334808
plt.figure(figsize=(20,4))
for i in range(model_gmm.means_.shape[0]):
    plt.scatter(np.arange(scaled_data.shape[1]), model_gmm.
        means_[i])
plt.xticks(ticks=np.arange(scaled_data.shape[1]), labels=
    scaled_data.columns)
plt.title('Cluster means')
plt.show()

# Drop useless features
drop_feats = [f'f_0{i}' for i in range(7)]
drop_feats = drop_feats + [f'f_{i}' for i in range(14,22)]
```

3 MORE PREPROCESSING, MODEL SELECTION AND TRAINING

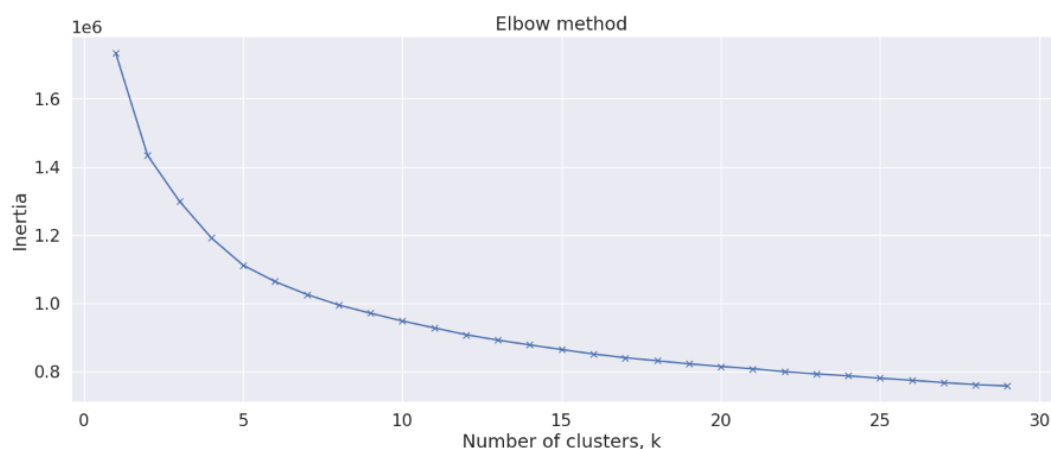
```
scaled_data_crop = scaled_data.drop(drop_feats, axis=1)

# Remake predictions for GMM
model_gmm_crop = GaussianMixture(n_components = 7, random_state
    =0)
preds_gmm_crop = model_gmm_crop.fit_predict(scaled_data_crop)

# Bayesian Gaussian Mixture Model
model_bgmm = BayesianGaussianMixture(n_components=7,
    covariance_type='full', max_iter=100, n_init=5, init_params=
    'random', random_state=0)
preds_bgmm = model_bgmm.fit_predict(scaled_data_crop)




sub = pd.read_csv('../input/tabular-playground-series-jul-2022/
    sample_submission.csv')

sub['Predicted'] = preds_bgmm
sub.to_csv('/kaggle/working/bgmm.csv', index=False)
```



4 Conclusion and Future Work

Our models performed comparatively different than each other. The competition metric is Rand Index, which is a measure of similarity between the predicted clusters and the ground truth clusters. Our k_means, GMM and BGMM performed with scores around 0.26, 0.49 and 0.60 respectively. The Bayesian Gaussian Mixture Model had the best performance between our chosen models. There are some future works to be done. It can be shown that the best score achieved out there is around 0.8 which was performed with a custom EM(Expectation_Maximization) algorithm with specific transformation to the variables[2]. One interesting point that was discussed in the public forum was about the data consisting of 7 group clusters of 6 clusters each. One may wish to explore further, by also examining the nature of our dataset with statistic tools and tests for understanding the distribution of these variables and the true effect of each on the resulting clustering performance.

Submission and Description	Private Score ⓘ	Public Score ⓘ
 km.csv Complete (after deadline) · 1d ago	0.27081	0.26642
 gmm.csv Complete (after deadline) · 1d ago	0.49882	0.49946
 bgmm.csv Complete (after deadline) · 1d ago	0.60092	0.60117

5 References

- [1] Walter Reade, Ashley Chow. (2022). Tabular Playground Series - Jul 2022. Kaggle. <https://kaggle.com/competitions/tabular-playground-series-jul-2022>
- [2] <https://www.kaggle.com/competitions/tabular-playground-series-jul-2022/discussion/341023>
- [3] <https://www.kaggle.com/code/samuelcortinhas/tps-july-22-unsupervised-clustering>
- [4] <https://www.kaggle.com/competitions/tabular-playground-series-jul-2022/discussion/3348>