DATA MINING SEMINAR

K-MEANS

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids.

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters

Step-4: Calculate the variance and place a new centroid of each cluster

Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

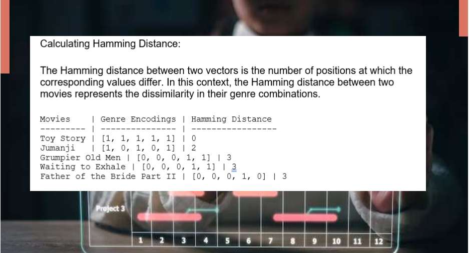
Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

Here's an example of performing k-means clustering on a 5-tuple movie dataset based on genres using the Hamming distance:

Dataset:

```
Movie Title | Genres
------- | --------
Toy Story (1995) | Adventure, Animation, Children, Comedy, Fantasy
Jumanji (1995) | Adventure, Children, Fantasy
Grumpier Old Men (1995) | Comedy, Romance
Waiting to Exhale (1995) | Comedy, Drama, Romance
Father of the Bride Part II (1995) | Comedy
```



Calculating Hamming Distance:

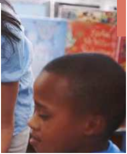
The Hamming distance between two vectors is the number of positions at which the corresponding values differ. In this context, the Hamming distance between two movies represents the dissimilarity in their genre combinations.

Movies	Genre Encodings	Hamming Distance
Toy Story	[1, 1, 1, 1, 1]	0
Jumanji	[1, 0, 1, 0, 1]	2
Grumpier Old Men	[0, 0, 0, 1, 1]	3
Waiting to Exhale	[0, 0, 0, 1, 1]	3
Father of the Bride Part II	[0, 0, 0, 1, 0]	3

Encoding Genres:

To apply k-means clustering, we need to encode the categorical genre values into numerical representations. We'll use binary encoding, where each genre is represented by a binary vector of length 5, with 1 indicating the presence of the genre and 0 indicating its absence.

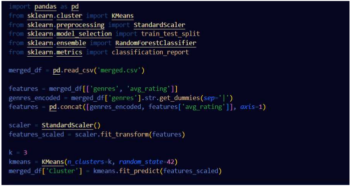
Genre	Binary Encoding
Adventure	[1, 0, 0, 0, 0]
Animation	[0, 1, 0, 0, 0]
Children	[0, 0, 1, 0, 0]
Comedy	[0, 0, 0, 1, 0]
Drama	[0, 0, 0, 0, 1]
Fantasy	[0, 0, 0, 0, 1]



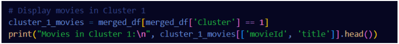
Clustering Results:

After applying the k-means algorithm, the movies are grouped into two clusters:

Cluster 1	Cluster 2
Toy Story	Jumanji
Grumpier Old Men	Waiting to Exhale
Father of the Bride Part II	

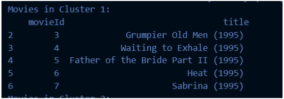


```python
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

merged_df = pd.read_csv('merged.csv')

features = merged_df[['genres', 'avg_rating']]
genres_encoded = merged_df['genres'].str.get_dummies(sep='|')
features = pd.concat([genres_encoded, features['avg_rating']], axis=1)

scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

k = 3
kmeans = KMeans(n_clusters=k, random_state=42)
merged_df['Cluster'] = kmeans.fit_predict(features_scaled)
```



```python
# Display movies in Cluster 1
cluster_1_movies = merged_df[merged_df['Cluster'] == 1]
print("Movies in Cluster 1:\n", cluster_1_movies[['movieId', 'title']].head())
```

OUTPUT:



```
Movies in Cluster 1:
   movieId                              title
2        3            Grumpier Old Men (1995)
3        4           Waiting to Exhale (1995)
4        5  Father of the Bride Part II (1995)
5        6                         Heat (1995)
6        7                      Sabrina (1995)
```

Random Forest Algorithm

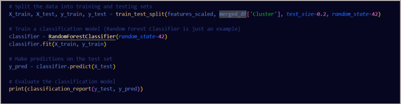
Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points.

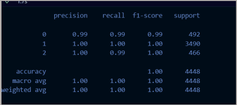
Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

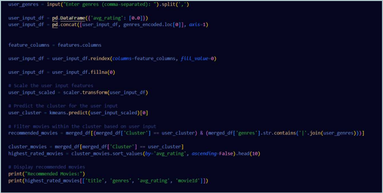
Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.



```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features_scaled, merged_df['Cluster'], test_size=0.2, random_state=42)

# Train a classification model (Random Forest Classifier is just an example)
classifier = RandomForestClassifier(random_state=42)
classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = classifier.predict(X_test)

# Evaluate the classification model
print(classification_report(y_test, y_pred))
```

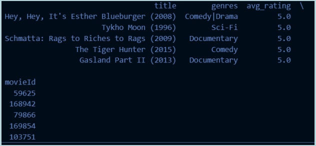


```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       492
           1       1.00      1.00      1.00      3490
           2       1.00      0.99      1.00       466

    accuracy                           1.00      4448
   macro avg       1.00      1.00      1.00      4448
weighted avg       1.00      1.00      1.00      4448
```



```python
user_genres = input("Enter genres (comma-separated): ").split(',')

user_input_df = pd.DataFrame({'avg_rating': [0.0]})
user_input_df = pd.concat([user_input_df, genres_encoded.loc[0]], axis=1)


feature_columns = features.columns

user_input_df = user_input_df.reindex(columns=feature_columns, fill_value=0)

user_input_df = user_input_df.fillna(0)

# Scale the user input features
user_input_scaled = scaler.transform(user_input_df)

# Predict the cluster for the user input
user_cluster = kmeans.predict(user_input_scaled)[0]

# Filter movies within the cluster based on user input
recommended_movies = merged_df[(merged_df['Cluster'] == user_cluster) & (merged_df['genres'].str.contains('|'.join(user_genres)))]

cluster_movies = merged_df[merged_df['Cluster'] == user_cluster]
highest_rated_movies = cluster_movies.sort_values(by='avg_rating', ascending=False).head(10)

# Display recommended movies
print("Recommended Movies:")
print(highest_rated_movies[['title', 'genres', 'avg_rating', 'movieId']])
```

OUTPUT:



```
                                     title       genres  avg_rating  \
Hey, Hey, It's Esther Blueburger (2008)  Comedy|Drama         5.0
                       Tykho Moon (1996)        Sci-Fi         5.0
Schmatta: Rags to Riches to Rags (2009)   Documentary         5.0
                 The Tiger Hunter (2015)        Comedy         5.0
                  Gasland Part II (2013)   Documentary         5.0

movieId
  59625
 168942
  79866
 169854
 103751
```

THANK YOU