# Report on Item Recommendation System Using KMeans Clustering

## 1. Introduction

This report outlines the development of an item recommendation system utilizing KMeans clustering. The system is designed to provide item suggestions based on user-defined parameters related to time (daypart, daytype, and season). By clustering similar items, the model enhances user experience through personalized recommendations.

## 2. Data Loading and Preprocessing

### 2.1 Loading Data

data = pd.read\_csv('app/dataset.csv')

The dataset contains information on items along with their attributes, such as daypart, daytype, and season.

### 2.2 Mapping Categorical Variables

The categorical variables are converted to numerical codes using predefined mappings:

- Daypart Mapping:  
 - Night: 3  
 - Morning: 2  
 - Afternoon: 1  
 - Evening: 0  
- DayType Mapping:  
 - Weekend: 0  
 - Weekday: 1  
- Season Mapping:  
 - Spring: 0  
 - Summer: 1  
 - Autumn: 2  
 - Winter: 3

### 2.3 Handling Missing Values

Missing values in the categorical columns are filled with the mode (most frequently occurring value):

data['Daypart'] = data['Daypart'].fillna(data['Daypart'].mode()[0])  
data['DayType'] = data['DayType'].fillna(data['DayType'].mode()[0])  
data['Season'] = data['Season'].fillna(data['Season'].mode()[0])

### 2.4 Feature Selection

The features for clustering are selected from the dataset:

X = data[['Daypart', 'DayType', 'Season']]  
y = data['Items']

### 2.5 Encoding Target Variable

The Items variable is encoded using Label Encoding:

label\_encoder = LabelEncoder()  
y\_encoded = label\_encoder.fit\_transform(y)

### 2.6 Oversampling to Handle Class Imbalance

The data is resampled using Random Over Sampling to address potential class imbalances:

ros = RandomOverSampler(random\_state=42)  
X\_resampled, y\_resampled = ros.fit\_resample(X, y\_encoded)

### 2.7 Feature Scaling

Features are standardized using StandardScaler:

scaler = StandardScaler()  
X\_scaled = scaler.fit\_transform(X\_resampled)

## 3. Clustering with KMeans

### 3.1 Fitting the KMeans Model

kmeans = KMeans(n\_clusters=5, random\_state=42)  
kmeans.fit(X\_scaled)

### 3.2 Assigning Clusters

Clusters are assigned to the original data:  
data['Cluster'] = kmeans.predict(scaler.transform(X))

### 3.3 Model Evaluation (Optional)

The silhouette score can be calculated to evaluate the quality of the clusters:  
silhouette\_avg = silhouette\_score(X\_scaled, kmeans.labels\_)  
print(f'Silhouette Score: {silhouette\_avg}')

## 4. Model Persistence

The KMeans model and scaler are saved using the pickle library for future reuse:

with open('kmeans\_model.pk1', 'wb') as model\_file:  
 pickle.dump(kmeans, model\_file)  
with open('scaler.pk1', 'wb') as scaler\_file:  
 pickle.dump(scaler, scaler\_file)

## 5. Reusing the Model

The saved model and scaler can be loaded for making predictions:

with open('kmeans\_model.pk1', 'rb') as model\_file:  
 kmeans\_loaded = pickle.load(model\_file)  
with open('scaler.pk1', 'rb') as scaler\_file:  
 scaler\_loaded = pickle.load(scaler\_file)

## 6. Recommendation Function

### 6.1 Function Implementation

def recommend\_items(daypart, daytype, season, top\_n=5):  
 # Convert inputs to codes  
 daypart\_code = daypart\_map[daypart]  
 daytype\_code = daytype\_map[daytype]  
 season\_code = season\_map[season]  
  
 input\_data = pd.DataFrame([[daypart\_code, daytype\_code, season\_code]], columns=['Daypart', 'DayType', 'Season'])  
 input\_scaled = scaler\_loaded.transform(input\_data)  
  
 cluster\_label = kmeans\_loaded.predict(input\_scaled)[0]  
  
 # Recommend items from the same cluster  
 recommended\_items = data[data['Cluster'] == cluster\_label]['Items'].value\_counts().head(top\_n).index.tolist()  
  
 return recommended\_items

### 6.2 Example Usage

To retrieve recommendations:  
recommended\_items = recommend\_items('Afternoon', 'Weekend', 'Spring', top\_n=5)  
print(f'Recommended items: {recommended\_items}')

## 7. Conclusion

The item recommendation system successfully employs KMeans clustering to provide personalized item suggestions based on user inputs related to time and season. This system enhances user experience and can be further improved by optimizing the clustering parameters and incorporating additional features.