# \

# Assignment Details

## Pre-processing done on data

The dataset file was read transaction by transaction and each transaction was saved as a list. A mapping was created from the unique items in the dataset to integers so that each item corresponded to a unique integer. The entire data was mapped to integers to reduce the storage and computational requirement. A reverse mapping was created from the integers to the items, so that the item names could be written in the final output file.

## Formulas Used

Confidence (X -> Y) = support (X U Y) / support (X)

Support (X, Y) = support-count (X, Y) / total dataset size

We have used support instead of support count because computations with integers are faster than that of floating point numbers.

*Support (X) = Support count (X) / Total number of transactions*

## Results for different for values of support and confidence

|  |  |  |
| --- | --- | --- |
| **Confidence/Support** | **No. of frequent itemsets** | **No of rules** |
| High confidence(MIN\_CONF=0.5)  High support count(MINSUP=60) | 725 | 60 |
| Low confidence(MIN\_CONF=0.1) High support count(MINSUP=60) | 725 | 1189 |
| High confidence(MIN\_CONF=0.5)  Low support count(MINSUP=10) | 11390 | 4187 |
| Low confidence(MIN\_CONF=0.1)  Low support count(MINSUP=10) | 11390 | 35196 |

frequent\_itemset.txt and association\_rules.txt for different MIN\_CONF and MINSUP values can be found in the RESULTS folder

## Observation

Most of the rules we generated have a common item (*whole milk* and *other vegetables*) on the consequent side. This happens when any item is very frequent in the transactions. This can be avoided by using *lift* instead of confidence.

Lift (X -> Y) = support (X U Y) / support (X) \* support (Y)

The purpose of this assignment is to cluster adults using K-means clustering and Hierarchical Agglomerative clustering models and to visualize clusters for predicted and actual cluster labels.

Your dataset is part of "Adult". You can find more information here:

[https://archive.ics.uci.edu/ml/datasets/adult (https://archive.ics.uci.edu/ml/datasets/adult).](https://archive.ics.uci.edu/ml/datasets/adult) The classification problem is whether they earn more than 50,000$ or not.

You need to submit this ipython file after renaming it.

Preprocessing will be needed for the data as most of the data is in string and needs to be quantified.

In

[ ]:

%%javascript

IPython

.

OutputArea

.

prototype

.

\_should\_scroll

**=**

**function**

(

lines

)

{

**return**

false

;

}

# Required Python Packages

In [ ]: *# Import required Python packages here*

*#Seaborn,numpy,pandas,sklearn,matplotlib only*

# Clustering

**Determine “k” value from the elbow method**

In this task, you will be using the elbow method to determine the optimal number of clusters for k-means clustering.

We need some way to determine whether we are using the right number of clusters when using k-means clustering. One method to validate the number of clusters is the elbow method.

The idea of the elbow method is to run k-means clustering on the dataset for a range of values of k (k will be from 1 to 10 in this task), and for each value of k calculate the sum of squared errors (SSE). Then, plot a line chart of the SSE for each value of k. If the line chart looks like an arm, then the "elbow" on the arm is the value of k that is the best. The idea is that we want a small SSE, but that the SSE tends to decrease toward 0 as we increase k (the SSE is 0 when k is equal to the number of data points in the dataset, because then each data point is a cluster, and there is no error between it and the center of its cluster). So our goal is to choose a small value of k that still has a low SSE, and the elbow usually represents where we start to have diminishing returns by increasing k.

For this task, you need to perform the elbow method for k from 1 to 10 and plot a line chart of the SSE for each value of k, and determine the best k (the number of clusters). Note that you need to use the whole dataset in this task and you need to print your decision for k.

In

[ ]:

*#########################begin code for Task 1-a*

*#########################begin code for Task 1-a*

**Visualization for K-Means Clustering**

In this task, you will be performing k-means clustering for k=2 and visualize the predicted training samples and actual training samples on scatter plots. Use 70% of the dataset for training and 30% of the dataset for testing. Perform kmeans for clustering samples in your training set.

Use two subplots for visualizing the predicted training samples and actual training samples on two scatter plots.

Since your dataset has multiple features(dimensions), you won't be able to plot your data on a scatter plot. Thus, you’re going to visualize your data with the help of one of the Dimensionality Reduction techniques, namely Principal Component Analysis (PCA). The idea in PCA is to find a linear combination of the two variables that contains most of the information. This new variable or “principal component” can replace the two original variables. You can easily apply PCA to your data with the help of scikit-learn.

In [ ]:

In

[ ]:

*###################begin code for Task 1-b-1: Split the dataset 70% for*

*### Important!!!*

*###################end code for Task 1-b-1*

*###################begin code for Task 1-b-2: Visualize the predicted tr*

*# Import PCA*

**from**

sklearn

.

decomposition

**import**

PCA

*# Create the KMeans model*

*# Compute cluster centers and predict cluster index for each sample*

*# Model and fit the data to the PCA model*

X\_train\_pca

**=**

**None**

*# Visualize the predicted training labels vs actual training labels.*

*### scatter(x, y, your\_data)*

x

**=**

X\_train\_pca

[:

,

0

]

y

**=**

X\_train\_pca

[:

,

1

]

*###################end code for Task 1-b-2*

Now, you need to visualize the predicted testing labels versus actual testing labels. Use the trained model in previous step.

*###################begin code for Task 1-b-3: Visualize the predicted te*

*# predict cluster index for each sample*

*# Model and fit the data to the PCA model*

X\_test\_pca

**=**

**None**

*# Visualize the predicted testing labels vs actual testing labels.*

*### scatter(x, y, your\_data)*

x

**=**

X\_test\_pca

[:

,

0

]

y

**=**

X\_test\_pca

[:

,

1

]

*###################end code for Task 1-b-3*

In this step, you need to provide the evaluation of your clustering model. Print out a confusion matrix.

In

[ ]:

*###################begin code for Task 1-b-4: Print out a confusion matr*

*###################end code for Task 1-b-4*

# Find the best Hierarchical Agglomerative Clustering Model

In this task, you will be performing Hierarchical Agglomerative clustering with different linkage methods (complete and average) and different similarity measures (cosine, euclidean, and manhattan) in order to find the best pair of linkage method and similarity measure. Use F1 score for evaluation and take n\_clusters = 2.

*###################begin code for Task 2-a: Print out a confusion matrix*

*# Import AgglomerativeClustering*

**from** sklearn.cluster **import** AgglomerativeClustering

*# Import pairwise\_distances for calculating pairwise distance matrix* **from** sklearn.metrics.pairwise **import** pairwise\_distances

*# Import f1\_score*

**from** sklearn.metrics **import** f1\_score

*## Calculate pairwise distance matrix for X\_train* pdm\_train **=** **None**

*## Model and fit the training data to the AgglomerativeClustering model*

*## complete linkage + cosine*

*## Model and fit the training data to the AgglomerativeClustering model*

*## complete linkage + euclidean*

*## Model and fit the training data to the AgglomerativeClustering model*

*## complete linkage + manhattan*

*## Model and fit the training data to the AgglomerativeClustering model*

*## average linkage + cosine*

*## Model and fit the training data to the AgglomerativeClustering model*

*## average linkage + euclidean*

*## Model and fit the training data to the AgglomerativeClustering model*

*## average linkage + manhattan*

print("F1-score for complete linkage + cosine", **None**) print("F1-score for complete linkage + euclidean", **None**) print("F1-score for complete linkage + manhattan", **None**) print("F1-score for average linkage + cosine", **None**) print("F1-score for average linkage + euclidean", **None**) print("F1-score for average linkage + manhattan", **None**)

*###################end code for Task 2-a*

REFERENCES

1. [http://www.recommenderbook.net/teachingmaterial/slides](http://www.recommenderbook.net/teaching-material/slides)
2. Recommender Systems Handbook, Ricci, F.; Rokach, L.; Shapira, B.; Kantor, P.B. (Eds.), 2011, Springer.
3. <http://ijana.in/papers/6.11.pdf>
4. [http://www.win.tue.nl/~laroyo/2L340/resources/rec ommender-systems-e-commerce.pdf](http://www.win.tue.nl/~laroyo/2L340/resources/recommender-systems-e-commerce.pdf)
5. Data Mining Concepts and Techniques 2nd Ed By Kamber pages 234-242.