Automotive Products Clustering Experiment

Beijing Institute of Technology

谨此承诺：

本人所写的实验报告及代码由本人独立完成，保证不存在任何剽窃、抄袭他人作业的现象。凡涉及其他同学的观点和材料，均作了注释，如出现抄袭及侵犯他人知识产权的行为，由本人承担由此产生的一切后果。

# Automotive Products Clustering Experiment

## Experiment Introduction

The automotive industry is highly competitive, with many car models vying for market share. In order to gain a competitive edge, companies must conduct thorough analysis of their products and their competitors. One useful approach to analyzing the competitive landscape of car models is clustering analysis, which groups similar cars together based on key features and characteristics.

In this experiment, we will be using three different clustering algorithms – KNN, Kmeans, and Gaussian mixture clustering – to cluster data on car models. By doing so, we will be able to provide a clustering classification for cars and identify the competing models for a given car model.

Through this experiment, we will have the opportunity to conduct a model portrait analysis, comparing and contrasting different car models within and across clusters. They will also be able to provide data for product positioning and competitive analysis, using the insights generated from the clustering analysis to inform their decision-making.

Overall, this experiment aims to provide us with practical skills and knowledge in competitive analysis, using clustering algorithms to identify and analyze the competitive landscape of car models.

## Experiment Objectives

1. To master three clustering methods: KNN, Kmeans, and Gaussian mixture clustering
2. To complete the clustering analysis of the experimental data.

## Relevant Theories and Knowledge

**K-means:**

Choosing the number of clusters

The first step is to define the K number of clusters in which we will group the data.

Initializing centroids

Centroid is the center of a cluster but initially, the exact center of data points will be unknown so, we select random data points and define them as centroids for each cluster. We will initialize 6 centroids in the dataset.

Assign data points to the nearest cluster

Now that centroids are initialized, the next step is to assign data points X\_n to their closest cluster centroid C\_k. Then calculate the distance between data point and centroid:

Re-initialize centroids

Next, we will re-initialize the centroids by calculating the average of all data points of that cluster, by using:

Repeat

Steps above are repeated until convergence, which is defined as the point at which the centroids no longer change or a maximum number of iterations is reached.

**KNN:**

k-Nearest Neighbors, is a simple but effective algorithm for classification and regression tasks.

Initialization

The algorithm begins by storing the training data points and their corresponding labels.

Distance calculation

For a given test data point, the distances between the test data point and all training data points are calculated using a distance metric, usually Euclidean distance or Manhattan distance.

Nearest neighbors selection

The k training data points with the smallest distances to the test data point are selected as the "nearest neighbors."

Label assignment

For classification tasks, the label of the test data point is assigned based on the majority class among the k nearest neighbors. For regression tasks, the output value of the test data point is assigned based on the average of the output values of the k nearest neighbors.

**Gaussian mixture clustering:**

Gaussian Mixture Clustering is a probabilistic clustering algorithm that models data points as a mixture of Gaussian distributions. The probability density function would be given by:

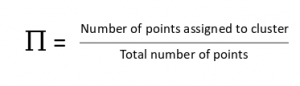
Expectation-Maximization (EM) is a statistical algorithm for finding the right model parameters. We typically use EM when the data has missing values, or in other words, when the data is incomplete.

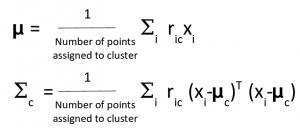
Expectation Step

For each data point, the probability of it belonging to each Gaussian distribution is calculated using Bayes' theorem. These probabilities are called the "responsibilities". For each point xi, calculate the probability that it belongs to cluster/distribution c1, c2, … ck.

Maximization Step

Post the E-step, we go back and update the Π, μ and Σ values. These are updated in the following manner:

1. The new density is defined by the ratio of the number of points in the cluster and the total number of points:
2. The mean and the covariance matrix are updated based on the values assigned to the distribution, in proportion with the probability values for the data point. Hence, a data point that has a higher probability of being a part of that distribution will contribute a larger portion:



## Experimental Tasks and Grading Criteria

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Task Name** | **Specific Requirements** | **Grading Criteria (100-point scale)** |  |
| **1** | KNN |  | Accuracy: 100 |  |
| **2** | K-means |  | Silhouette Coefficient: 1 |  |
| **3** | Gaussian mixture clustering |  | Accuracy: 100 |  |

## Experimental Conditions and Environment

|  |  |  |  |
| --- | --- | --- | --- |
| Requirements | Name | Version | Remarks |
| **Programming Language** | Python | 3.9.0 |  |
| **Development Environment** | Mindspore | 1.10.0 |  |
| **Third-party toolkits/libraries/plugins** | Pandas  Numpy  Matplotlib  Seaborn  Scipy | 1.4.4  1.20.3  3.4.3  0.11.2  1.6.3 |  |
| **Other Tools** | None |  |  |
| **Hardware Environment** | Intel Core i7-10750H |  |  |

## Experimental Data and Description

|  |  |
| --- | --- |
| **Attribute (Entry)** | Content |
| **Dataset Name** | **Automotive product clustering analysis dataset** |
| **Dataset Origin** | Automotive products |
| **Main Contents of the Dataset** | An automotive product clustering analysis dataset typically contains information about a range of automotive products that can be used in cluster analysis to determine similarities and differences between different automotive products.  The following are the main elements that may be included in an automotive product cluster analysis dataset:  Attributes of automotive products, size and weight of automotive products, fuel economy of automotive products, safety features of automotive products, comfort and convenience of automotive products.  The information can help perform cluster analysis to find the similar groups of automotive products and can be used for market analysis, product positioning, and marketing strategies. |
| **Dataset File Format** | car\_price.csv, the data includes 26 fields for 205 cars. |

## Experimental Steps and Corresponding Codes

|  |  |
| --- | --- |
| **Step number** | **1** |
| **Step Name** | Data process |
| **Step Description** | In this step, I’ll load the data from csv file and transform the word information into different number classes for further implementation. |
| **Code and Explanation** | Details are shown in data\_preprocess.ipynb  # load data  train=pd.read\_csv('car\_price.csv')  # data preprocessing  # The classification of cars can be basically determined by the 'carbody' attribute, and there are roughly five categories. However, the last two categories are very few and can basically be counted as "sports cars", so the current review of car classification is determined to be four categories.  train.loc[(train['carbody']=='hardtop'),'carbody']='sportcar'  train.loc[(train['carbody']=='convertible'),'carbody']='sportcar'  train['carbody'].value\_counts()  train['brand']=train['CarName'].str.split(" ",expand=True)[0]  train['brand']=train['brand'].replace({'toyouta':'toyota','vokswagen':'volkswagen','vw':'volkswagen','porcshce':'porsche','maxda':'mazda','Nissan':'nissan'})  # cylindernumber  train['cylindernumber']=train['cylindernumber'].map({'two':'2','three':'3','four':'4','five':'5','six':'5','eight':'8','twelve':'12'})  train['cylindernumber']=pd.to\_numeric(train['cylindernumber'],downcast='integer')  # car size: car height & car width  train['carsize']=train['carlength']\*train['carwidth']  sns.distplot(train['carsize'])  train['carsize\_band']=pd.cut(train['carsize'],4)  train.loc[(train['carsize']<=10111),'carsize']=0  train.loc[(train['carsize']>10111)&(train['carsize']<=11714),'carsize']=1  train.loc[(train['carsize']>11714)&(train['carsize']<=13317),'carsize']=2  train.loc[(train['carsize']>13317),'carsize']=3  train['carsize']=train['carsize'].astype(np.int64)  train.drop(['carsize\_band','carlength','carwidth'],axis=1,inplace=True)  #car weight  train['carheight\_band']=pd.cut(train['carheight'],5)  train.loc[(train['carheight']<=50.2),'carheight']=0  train.loc[(train['carheight']>50.2)&(train['carheight']<=52.6),'carheight']=1  train.loc[(train['carheight']>52.6)&(train['carheight']<=55),'carheight']=2  train.loc[(train['carheight']>55)&(train['carheight']<=57.4),'carheight']=3  train.loc[(train['carheight']>57.4),'carheight']=4  train['carheight']=train['carheight'].astype(np.int64)  train.drop(['carheight\_band'],axis=1,inplace=True) |
| **Output results and Interpretation** | By observing the distribution of different feature, I’ll separate them into different classes. |

|  |  |
| --- | --- |
| **Step number** | **2** |
| **Step Name** | KNN algorithm implementation |
| **Step Description** | In this step, I will implement KNN using mindspore and other tools. |
| **Code and Explanation** | def distance(d1,d2):  res = 0  for key in ("wheelbase", "carlength", "carwidth", "carheight", "horsepower", "price"):  res+=(float(d1[key])-float(d2[key]))\*\*2  return res\*\*0.5  def knn(data):  res = [  {"result":train['symboling'],"distance":distance(data,train)}  for train in train\_set  ]  res = sorted(res,key = lambda item:item['distance'])  res2 = res[0:K]  result = {'-2':0,'-1':0, '0':0, '1':0, '2':0, '3':0}  sum = 0  for r in res2:  sum+=r['distance']  for r in res2:  result[r['result']]+=1-r['distance']/sum  tmp = max(result.values())  for key,value in result.items():  if(value == tmp):  return key |
| **Output results and Interpretation** | This step will calculate the distance and return the result of classification. |

|  |  |
| --- | --- |
| **Step number** | **3** |
| **Step Name** | K-means algorithm implementation |
| **Step Description** | Some functions related to K-means  Initialization: The algorithm begins by randomly selecting K initial centroids, which represent the centers of each of the K clusters.  Assignment: Each data point is assigned to the nearest centroid based on the Euclidean distance between the data point and the centroid.  Update: The centroids of each cluster are updated based on the mean of all data points assigned to that cluster. |
| **Code and Explanation** | def createInitial(sample,D): # D is dataset and sample is a sample  Set=[]  for i in sample:  a=[]  a.append(D[i])  Set.append(a)  return Set    def InsertToSet(D,setA,b,Center): #b is row vector , setA is setCluster , CenterC is CenterSet  dismin=0  minN=0  for j in range(len(Center[0])):  dismin=dismin+(Center[0][j]-b[j])\*(Center[0][j]-b[j])    for u in range(len(Center)):  dis=0  uu=Center[u]  for j in range(len(uu)):  dis=dis+(uu[j]-b[j])\*(uu[j]-b[j])  if dis<dismin and dis!=0:  minN=u    setA[minN].append(b)  def Caculcenter(setA):  a=[]  for set in setA:  u=set[0]  for i in range(1,len(set)):  u =list(map(lambda x,y:x + y,u,set[i]))  x\_=[]  for i in range(len(u)):  x\_.append(u[i]/len(set))  u=x\_  a.append(u)  return a  def renewdata(D,Set): #D is dataset and S is already get cluster  Dnew=[]  for i in range(len(D)):  start=0  for j in range(len(Set)):  for jj in range(len(Set[j])):  if Set[j][jj]==D[i]:  start=start+1  if start==0:  Dnew.append(D[i])  return Dnew      def kmeans(D,K,N):  sample=Inport6  set = createInitial(sample,D)  D=renewdata(D,set)  u=Caculcenter(set)  p=[]  for i in sample:  p.append(D[i])    for j in range(N):  D=renewdata(D,set)  for i in range(len(D)):  InsertToSet(D,set,D[i],u)  u1=Caculcenter(set)  if u == u1:  break  u=u1    return set |
| **Output results and Interpretation** |  |

|  |  |
| --- | --- |
| **Step number** | **3** |
| **Step Name** | Silhouette Coefficient |
| **Step Description** | Calculate the silhouette coefficient to demonstrate the effectiveness of K-means. |
| **Code and Explanation** | def SilCoef(dataA):  Mcoef=0  l=0  for data in range(len(dataA)):  tcoef=0  tl=len(dataA[data])  l=l+tl  for j in range(len(dataA[data])):  a=0  b=0  max=0  for jj in range(len(dataA[data])):  for k in range(len(dataA[data][j])):  a=a+(dataA[data][j][k]-dataA[data][jj][k])\*(dataA[data][j][k]-dataA[data][jj][k])  a=a/len(dataA[data])  for k in range(len(dataA)):  if k == j:  continue  else:  start=0  temp=0  for kk in range(len(dataA[k])):  for kkk in range(len(dataA[k][kk])):  temp=temp+(dataA[k][kk][kkk]-dataA[data][j][kkk])\*(dataA[k][kk][kkk]-dataA[data][j][kkk])  if start==0:  b=temp/len(dataA[k])  start=start+1  else:  if (temp/len(dataA[k]))>b:  b=(temp/len(dataA[k]))  if a > b :  max=a  else:  max=b    tcoef=(b-a)/max  Mcoef=Mcoef+tcoef    return Mcoef/l |
| **Output results and Interpretation** |  |

|  |  |
| --- | --- |
| **Step number** | **3** |
| **Step Name** | Gaussian mixture clustering |
| **Step Description** |  |
| **Code and Explanation** | # train GMM  car\_base\_info  gaussian=GMM(n\_components=5,random\_state=42).fit(train)  gaussian\_pred=gaussian.predict(train)  preds = pd.DataFrame({'class':gaussian\_pred})  result=pd.concat((car\_base\_info,preds),axis=1)  # evaluate  correct = 0  result['class']  train\_pre['symboling']  for i in range (len(result['class'])):  if result['class'][i] == train\_pre['symboling'][i]:  correct+=1    res = correct/len(result['class']) |
| **Output results and Interpretation** |  |

## Experiment Difficulties and Precautions

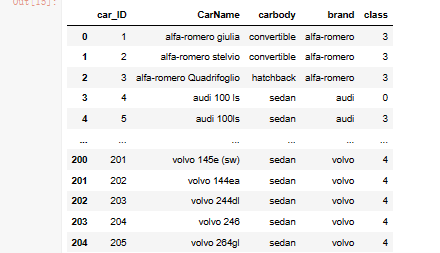
## Experiment Results and Interpretation

1. Result of data preprocess

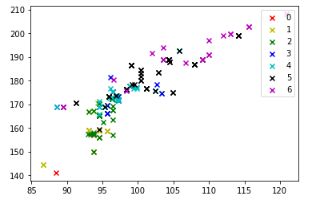
The result is shown in data\_preprocess.csv

1. Result of KNN

The accuracy of KNN is 41.58%



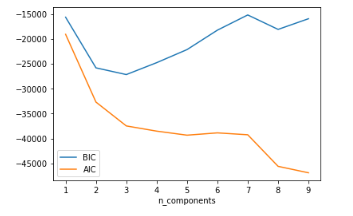
1. Result of K-means

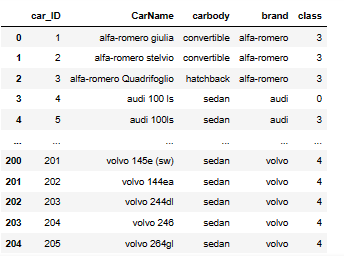


The Silhouette Coefficient of K-means clusters is 0.84486. The more the coefficient close to 1, the better the separation of clusters.

1. Result of GMM

Select the most appropriate number of clusters





## References

[1] Li K, Machine Learning and Knowledge Discovery, 2023 Spring

## Experiment-related Metadata

|  |  |
| --- | --- |
| Metadata Item | Content |
| Case name | Automotive product clustering analysis |
| Applicable course name | Machine learning |
| Keyword/Search Term |  |
| AliTianchi URI | Publish the experiment on AliCloud and give the URI of the experiment on AliTianchi |

## Remarks and Others