Data Mining Tools and Techniques

Cloud Service Provider Recommendation

Under Professor:
Dr. Kewei Sha

Team Members:
Shraddha Varvadekar
Swapnil Patel
Fatemeh Farahani

Purpose of Project:

The aim of the project is to give recommendation of a cloud service to new user/company from the available dataset of cloud service providers. We want to achieve a maximum probability to recommend a correct cloud service to the user company.

Project Description

- **Preparing Datasets**

Our aim is to have one dataset. The dataset will contain all the details of the cloud services such as the revenue, number of employees, customer satisfaction, the services provided by the company, name of the company, etc. The dataset will be formed with respect to the cloud user company. The dataset will have information of the cloud service providers in numbers and nominal (binary) values would define if it uses the services or not provided by the company respectively. Main attributes of services considered are database, networking, and storage, infrastructure.

The basic idea of the project is to do a predictive analysis to make suggestions to the companies if they can or cannot opt for cloud services based on the attributes it satisfies from the dataset.

The dataset will be developed manually considering the values available from different resources for the selected attributes. The dataset will have the cloud provider companies such as Google, Salesforce, IBM, and Amazon Web Service and others. The cloud user companies would be Netflix, Instagram, Dell and others.

There will not be much data cleaning in terms of having incomplete, noisy, inconsistent data since the data will be manually developed and the number of records for each dataset wouldn't be huge.

- **Result of the Project - Predictive Analysis**

We will design an input interface where a new user will input its company profile. The purpose of this interface is to help the company get a recommendation if it can use the cloud services based on the metrics such as revenue and price. We will form clusters of the dataset depending on the cloud services it uses. These clusters will help us evaluate the profile of new company and recommend it to use and follow the cloud services. This will be done by predicting a decision tree through the input dataset given to the system. Also following a heuristic approach to approach the outcome of the result.

- **Evaluation Metric**

  a. **Reliability:** To find whether the analysis provides us with a reliable answer to evaluate the new company.
  b. **Accuracy:** Measuring the accuracy of the system and see how effective it is.

- **Plan of the Project**

1st Week: Getting a basic project idea and gathering resources

2nd Week: Defining the dataset required for our project

3rd Week: Overview of "R" language and study of various techniques

4th Week: Setting up the environment and designing project

5th Week: Implementation of project

6th Week: Implementation of project

7th Week: Implementation and testing of project along with documentation

- **Background research with bibliography of relevant searches**

The background research done in our project was to read about the various mining algorithms in clustering and classification with thorough discussion with the professor. We came up with an idea to follow and implement the classification approach which would best suit our project need.

We used R language for implementation and required to refer various tutorials to study R. Further research and reading on the project is in progress and we are updating ourselves with new ideas and methods to further enhance the project.[1]

We also developed the same project idea using Python. As R used decision tree to give the recommendation, in Python we followed another heuristic approach working parallel with R and giving the same cloud recommendation to the cloud user company by not typically forming a decision tree.

- **Detailed Research methodology or approach taken in the project**

For the dataset, thorough research was made and all the attributes were manually filled by visiting every cloud company website and other informative resources such as stock market data description for a company. The various annual reports were read and followed to fill in the values of the attributes.

- **Status of Implementation**

This project is developed with two different techniques but with the same ideology. We have created our own dataset which is named as "Final Dataset 3_latest.xlsx".

Now the data is integrated with R using the following inputs which is added in a variable 'mother'.

```
mother=read_excel("Final Dataset 3_ latest.xlsx",sheet=1)
```

```
mother_sub=subset(mother[1:157,])
```

We performed data reduction in this dataset by removing the first attribute which was 'Cloud User Company'.

```
mother=mother_sub[,-1]
```

Since we had both nominal and numeric data in our dataset we had to standardize both the data into one nominal form which could then be used for our analysis. For the nominal data we followed the chi-square approach. And for the nominal data we adopted the following two hypothesis:

*{**H0**=there is no correlation between two attributes}*

*{**H1**=there is significant correlation between two attributes}*

```
chisq.test(mother[,5],mother[,6])

chisq.test(mother[,5],mother[,7])

chisq.test(mother[,5],mother[,8])

chisq.test(mother[,5],mother[,9])
```

For the numeric data we used the min-max normalization to normalize the data. The numeric data was brought in the 2 digit range (10 to 99). Also, correlation coefficient was performed by observing the plots which would indicate a relation between two attributes.

```
cor(mother[,1],mother[,2])

plot(mother[,1],mother[,2],xlab="number of
employee",ylab="revenue",col=20)
```

A sample min-max normalization for the 'number of employee' attribute was done as follows.

```
employee=mother[,1]

mini1=min(employee)

maxi1=max(employee)

standard_employee=((employee-mini1)/(maxi1-
mini1))*89+10

range(standard_employee)

mother=cbind(standard_employee,mother)

mother=mother[,-2]
```

The Scatter plots and Q-Q plots were made to get a visualization information which helped us in deciding the ‘bins’ to be formed for the data. Binning technique was employed to distribute the data into 3 different bins such as ‘high’, ‘medium’ and ‘low’.

```
dm=dim( mother)

for (i in 1:dm[1]){

if (mother[i,1]>=69) {mother[i,1]="high"}

else if (mother[i,1]>=39) {mother[i,1]="medium"}

else if (mother[i,1]<39)  {mother[i,1]="low"}     }
```

- **Detection Technique**

After this categorization which was data reduction and cleaning we moved on to the predictive analysis technique. We used classification as the prediction technique for the analysis. We created a decision based on the data which was reduced in the above process.

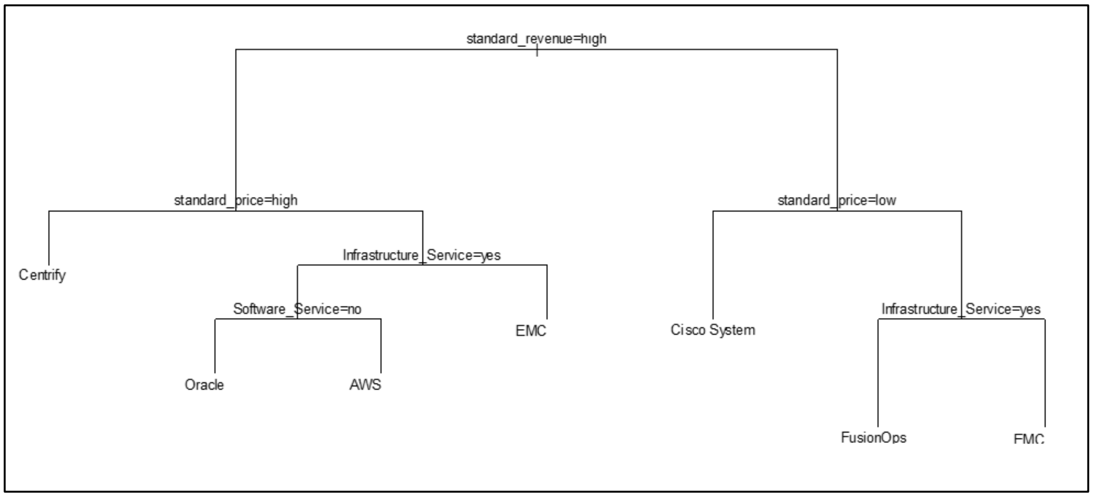
The data was divided into two parts testing and training dataset which was distributed equally (50%). A decision tree was developed firstly with the training data. This training data was then compared with testing data and a similar prediction model was developed.

- **Evaluation Results**

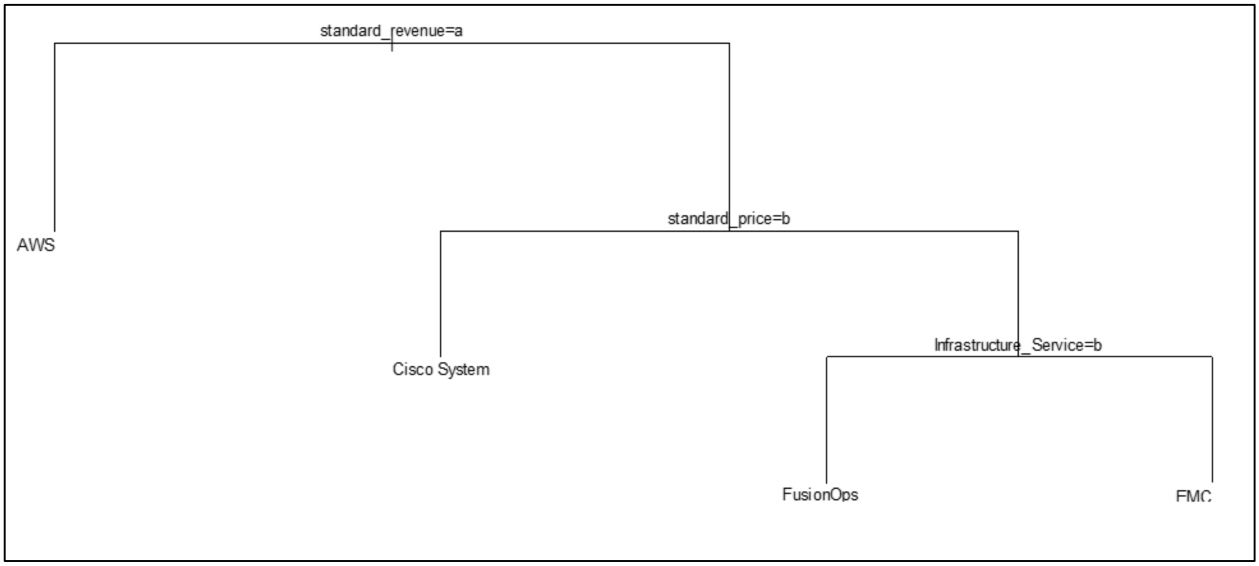
For the decision tree, after the tree was formed with the testing and training data we calculated the misclassification rate which helped us in our evaluation. This was calculated to better understand if we needed pruning of the tree which would help us give a better analysis. The misclassification rate calculated at first was 0.9746835 which could be reduced through pruning which came out to be 0.9367089. Similarly a confusion matrix was formed to give accuracy result.

- **Output**

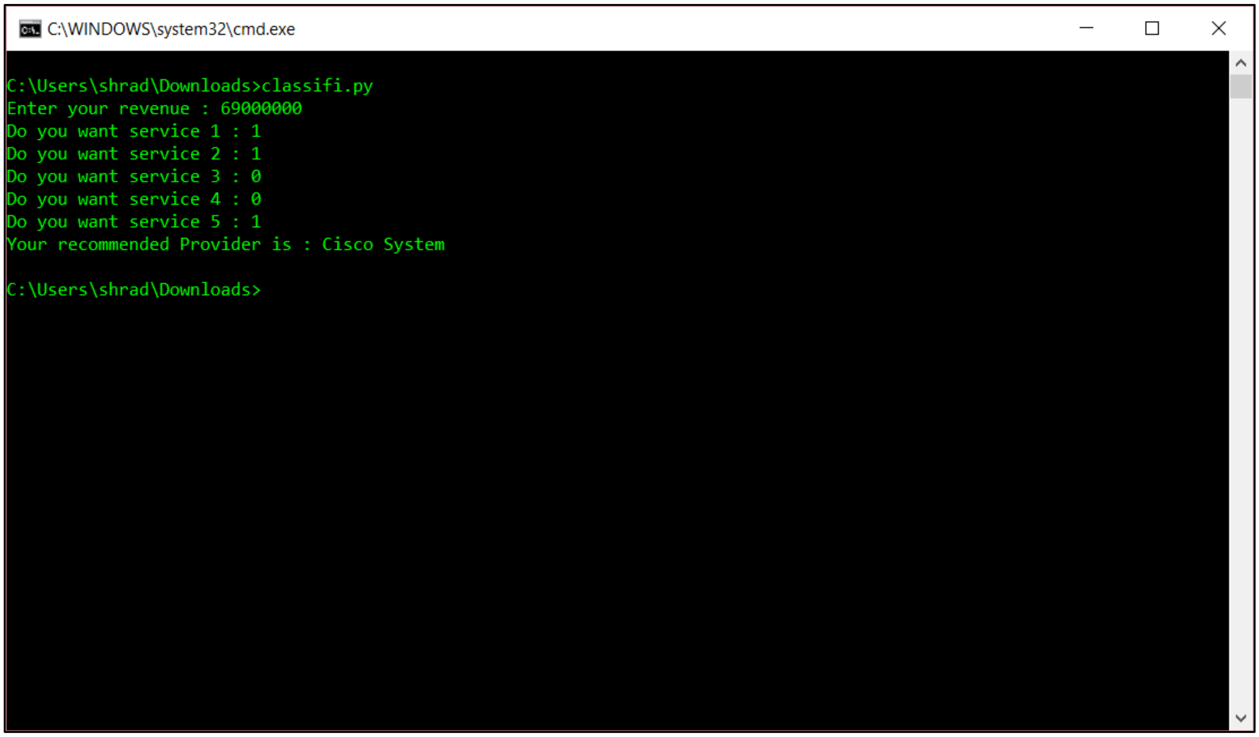
In R



The above figure is the tree which is formed with the input dataset and is without pruning. The tree below is one with pruning which was done using cross-validation.



In Python



```
C:\Users\shrad\Downloads>classifi.py
Enter your revenue : 69000000
Do you want service 1 : 1
Do you want service 2 : 1
Do you want service 3 : 0
Do you want service 4 : 0
Do you want service 5 : 1
Your recommended Provider is : Cisco System

C:\Users\shrad\Downloads>
```

- **Conclusion**

As we did not have our own dataset and a dummy dataset with fewer values was developed it held out to be the most important part of the project. The second important part of the project was to standardize both the numeric and nominal data to one form to form a decision tree. After using techniques such as R, Python we did come up to an analysis of recommending any new company a cloud service which it can use or go for. We can thus say that our results met the expectation.

- **Future Work**

Since the major focus and time was spent on the dataset which was small, we could work more on the dataset by adding more data to it and also different attributes if required. The reason could be the range or area of prediction would increase and perfect recommendation system could be developed.

- **Program**

Using R:

```
install package (xlsx,readxl,rpart)

setwd("C:\\Users\\FATEMEH\\Desktop")

getwd()

require(readxl)

mother=read_excel("Final Dataset 3_ latest.xlsx",sheet=1)

mother_sub=subset(mother[1:157,])


#data reduction:remove first column#

mother=mother_sub[,-1]

dim(mother)

#compute the correlation using chi.square for nominal attributes
,significant level alpha=0.05 #

chisq.test(mother[,5],mother[,6])

chisq.test(mother[,5],mother[,7])

chisq.test(mother[,5],mother[,8])

chisq.test(mother[,5],mother[,9])

chisq.test(mother[,6],mother[,7])

chisq.test(mother[,6],mother[,8])

chisq.test(mother[,6],mother[,9])
chisq.test(mother[,7],mother[,8])

chisq.test(mother[,7],mother[,9])

chisq.test(mother[,8],mother[,9])

#column 6(Software_Service) and column
9(Cloud_Platform_and_Development_Service ) are significantly
correlated #

#column 7(Cloud_Security_Service) and column
8(Cloud_Storage_Service) are significantly correlated#

 #remove column 8(Cloud_Storage_Service) #

mother=mother[,-8]

#remove column (Cloud_Platform_and_Development_Service )#

mother=mother[,-8]

#compute the correlation using correlation coefficient for numeric
attributes #

cor(mother[,1],mother[,2])

plot(mother[,1],mother[,2],xlab="number of
employee",ylab="revenue",col=20)

cor(mother[,1],mother[,4])

plot(mother[,1],mother[,4],xlab="number of
employee",ylab="customer satisfaction")

cor(mother[,1],mother[,8])

plot(mother[,1],mother[,8],xlab="number of
employee",ylab="price")

cor(mother[,2],mother[,4])

plot(mother[,2],mother[,4],xlab="revenue",ylab="customer
satisfaction",col=90)

cor(mother[,2],mother[,8])

plot(mother[,2],mother[,8],xlab="revenue",ylab="price",col=20)

cor(mother[,4],mother[,8])

plot(mother[,4],mother[,8],xlab="customer
satisfaction",ylab="price",col=90)
```

```
#==============================================================#

min & max standardizing (number of employee)attribute in to 2 digit range#

#so it better fits in to the category #

employee=mother[,1]

mini1=min(employee)

maxi1=max(employee)

standard_employee=((employee-mini1)/(maxi1-mini1))*89+10

range(standard_employee)

mother=cbind(standard_employee,mother)

mother=mother[,-2]

mother

#creating vector  #

p=rep(1:157,1)

q=(p-.5)/157

w=rep(1,157)

#scatter plot and qqplot of (number of employee) attribute#

plot(mother[,1],main="scatter plot of standard_employee " ,ylab="standard_employee" )

qqplot(q,mother[,1],ylab="standard_employee",main="prcentile plot of standard_employee")

qqplot(mother[,1],w,xlab="standard_employee")

par(mfrow=c(2,2))

qqnorm(mother[,1],ylab="standard_employee")

range(mother[,1])

#by visualizing this plot bining in to three category is appropriate #

**CATHEGORIZE EMPLOYEE NUMBERS**

dm=dim( mother)
```

```
for (i in 1:dm[1]){

if (mother[i,1]>=69) {mother[i,1]="high"}

else if (mother[i,1]>=39) {mother[i,1]="medium"}

else if (mother[i,1]<39)  {mother[i,1]="low"}

}

mother

#2=============================================================#
min & max standardizing (revenue)attribute in to 2 digit range#

#so it better fits in to the category #

revenue=mother[,2]

maxi2=max(revenue)

standard_revenue=((revenue-mini2)/(maxi2-mini2))*89+10

range(standard_revenue)

mother=cbind(standard_revenue,mother)

mother=mother[,-3]

mother

#scatter plot and qqplot of (revenue) attribute#

plot(mother[,1],main="scatter   plot   of   standard_revenue   "
,ylab="standard_revenue" )

qqplot(q,mother[,1],ylab="standard_revenue",main="prcentile  plot
of standard_revenue")

qqplot(mother[,1],w,xlab="standard_revenue")

par(mfrow=c(2,2))

qqnorm(mother[,1],ylab="standard_revenue")

range(mother[,1])

#by  visualizing  this  plot  bining  in  to  three  category  is
appropriate #

**CATHEGORIZE REVENUE**

for (J in 1:dm[1]){
```

```
if (mother[J,1]>=69) {mother[J,1]="high"}
else if (mother[J,1]>=39) {mother[J,1]="medium"}
else if (mother[J,1]<39)  {mother[J,1]="low"}
}
#3=============================================================#
min & max standardizing (price)attribute in to 2 digit range#
#so it better fits in to the category #
price=mother[,8]
mini3=min(price)
maxi3=max(price)
standard_price=((price-mini3)/(maxi3-mini3))*89+10
range(standard_price)
mother=cbind(standard_price,mother)
mother=mother[,-9]
mother
#scatter plot and qqplot of (price) attribute#
plot(mother[,1],main="scatter   plot   of   standard_price   "
,ylab="standard_price" )
qqplot(q,mother[,1],ylab="standard_price",main="prcentile plot of
standard_price ")
qqplot(mother[,1],w,xlab="standard_price")
par(mfrow=c(2,2))
qqnorm(mother[,1],ylab="standard_price")
range(mother[,1])
#by  visualizing  this  plot  bining  in  to  three  category  is
appropriate #
**CATHEGORIZE PRICE**
for (j in 1:dm[1]){
if (mother[j,1]>=69) {mother[j,1]="high"}
```

```
else if (mother[j,1]>=39) {mother[j,1]="medium"}

else if (mother[j,1]<39)  {mother[j,1]="low"}

}

4#=============================================================#

plot(mother[,5],main="scatter  plot  of  customer_satisfaction  "
,ylab="customer_satisfaction")

qqplot(q,mother[,5],ylab="customer_satisfaction",main="prcentile
plot of customer_satisfaction ")

qqplot(mother[,5],w,xlab="customer_satisfaction")

qqnorm(mother[,5],ylab="customer_satisfaction")

range(mother[,5])

**CATHEGORIZE SATISFACTION**

for (I in 1:dm[1]){

if (mother[I,5]>=74)  {mother[I,5]="satisfy"}

else if (mother[I,5]>=51)  {mother[I,5]="mutual"}

else if (mother[I,5] <51)   {mother[I,5]="unhappy"}

}

================================================================

mother1=mother

dim(mother1)

# we can repeat this section for different seed=3 #

set.seed(4321)

train=sample(1:nrow(mother1),nrow(mother1)*(1/2))

test= -train

trainning_data=mother1[train,]

testing_data=mother1[test,]

dim(testing_data)

dim(trainning_data)
tree_model=rpart(Cloud_Provider_Company~Infrastructure_Service+S
oftware_Service+Cloud_Security_Service+
standard_price+standard_revenue+standard_employee+Customer_Satis
faction, method= "class" , data=trainning_data)

plot(tree_model)

text(tree_model,pretty=0)

#compare testing dataset with  Cloud_Provider_Company#

Cloud_Provider_Company=mother1[,4]

testing_provider=Cloud_Provider_Company[test]

#check how the model is doing using test data & how testing _data
is classified by tree_model#

tree_pred=predict(  tree_model,testing_data ,type="class")

# it gave us misclassification ERROR to see if  prunning is needed#

mean(tree_pred != testing_provider)

#confusion matrix#

t=table(tree_pred,testing_data$Cloud_Provider_Company)

summary(t)

plot(t)

# to prun the tree cross validation shows where to stop pruning#

printcp(tree_model) # display the results

plotcp(tree_model) # visualize cross-validation results

summary(tree_model) # detailed summary of splits

prune_model=prune(tree_model,cp=tree_model$cptable[which.min(tre
e_model$cptable[,"xerror"]),"CP"])

plot(prune_model)

text( prune_model)

prun_tree_pred=predict(  prune_model,testing_data ,type="class")

mean(prun_tree_pred != testing_provider)

#we want to predict for a single data point#

single_datapoint49=testing_data[49,]
```

```
single_datapoint49=single_datapoint49[,-4]

single_datatee49_pred=predict(tree_model,single_datapoint49,type
="class")

single_dataprun49_pred=predict(    prune_model,single_datapoint49
,type="class")

################################################################

single_datapoint1=trainning_data[1,]

single_datapoint1=single_datapoint1[,-4]

single_datatree1_pred=predict(     tree_model,   single_datapoint1
,type="class")

single_dataprun1_pred=predict(    prune_model,   single_datapoint1
,type="class")
```

Using Python

```
import csv

import math

def mining_model():

     """Array for storing data for different services"""

     a1 = [ 4*[0] for i in range(200) ]

     a2 = [ 4*[0] for i in range(200) ]

     a3 = [ 4*[0] for i in range(200) ]

     a4 = [ 4*[0] for i in range(200) ]

     a5 = [ 4*[0] for i in range(200) ]

"""Array for storing price"""

     p1 = []

     p2 = []

     p3 = []

     p4 = []

     p5 = []
```

```
"""counter for each category"""
     s1 = 0
     s2 = 0
     s3 = 0
     s4 = 0
     s5 = 0

     def roundup(x):
         return int(math.ceil(x / 10.0)) * 10

     with open('csvfile.csv', 'rb') as f:
         reader = csv.reader(f)
         for row in reader:
               if(row[2] == "TRUE"):
                    a1[s1][0] = row[0]
                    a1[s1][1] = row[1]
                    a1[s1][2] = row[7]
                    a1[s1][3] = row[2]
                    p1.append(int(row[0]))
                    s1 =  s1 + 1

     with open('csvfile.csv', 'rb') as f:
         reader = csv.reader(f)
         for row in reader:
               if(row[3] == "TRUE"):
                    a2[s2][0] = row[0]

                    a2[s2][1] = row[1]

                    a2[s2][2] = row[7]

                    a2[s2][3] = row[2]

                    p2.append(int(row[0]))

                    s2 = s2 + 1

     with open('csvfile.csv', 'rb') as f:

         reader = csv.reader(f)

         for row in reader:

               if(row[4] == "TRUE"):

                    a3[s3][0] = row[0]

                    a3[s3][1] = row[1]

                    a3[s3][2] = row[7]

                    a3[s3][3] = row[2]

                    p3.append(int(row[0]))

                    s3 = s3 + 1


     with open('csvfile.csv', 'rb') as f:

         reader = csv.reader(f)

         for row in reader:

               if(row[5] == "TRUE"):

                    a4[s4][0] = row[0]

                    a4[s4][1] = row[1]

                    a4[s4][2] = row[7]

                    a4[s4][3] = row[2]
                    p4.append(int(row[0]))
                    s4 = s4 + 1

     with open('csvfile.csv', 'rb') as f:
         reader = csv.reader(f)
         for row in reader:
               if(row[6] == "TRUE"):
                    a5[s5][0] = row[0]
                    a5[s5][1] = row[1]
                    a5[s5][2] = row[7]
                    a5[s5][3] = row[2]
                    p5.append(int(row[0]))
                    s5 = s5 + 1

# to delete extra index for each service
     del a1[s1:200]
     del a2[s2:200]
     del a3[s3:200]
     del a4[s4:200]
     del a5[s5:200]
"""price min and max array for each category"""
     pr1 = {}
     pr2 = {}
     pr3 = {}
     pr4 = {}
```

```
    pr5 = {}

    pr1["min"] = min(p1)
    pr1["max"] = max(p1)

    pr2["min"] = min(p2)
    pr2["max"] = max(p2)

    pr3["min"] = min(p3)
    pr3["max"] = max(p3)

    pr4["min"] = min(p4)
    pr4["max"] = max(p4)

    pr5["min"] = min(p5)
    pr5["max"] = max(p5)
"""Array for storing range"""
    p1 = {}
    p2 = {}
    p3 = {}
    p4 = {}
    p5 = {}

    int1 = (pr1['max'] - pr1['min']) / 5
    int2 = (pr2['max'] - pr2['min']) / 5
    int3 = (pr3['max'] - pr3['min']) / 5

    int4 = (pr4['max'] - pr4['min']) / 5

    int5 = (pr5['max'] - pr5['min']) / 5


    model = []


    """Service 1"""

    provider = []

    mindata = pr1['min']

    for i in range(5):

        p1[i] = {}

        data = {}

        data['start'] = mindata

        data['end'] = mindata + int1

        pricelist = []

        mindata = mindata + int1

        for j in range(len(a1)):

            if(int(a1[j][0])>=      int(data['start'])      and
int(a1[j][0])<= int(data['end'])):

                pricelist.append(a1[j][2])

                provider.append(a1[j][1])

        data['provider']                                   =
provider[pricelist.index(min(pricelist))]

        data['price'] = min(pricelist)

        p1[i] = data

    """Service 2"""
    provider = []
    mindata = pr2['min']
    for i in range(5):
        p2[i] = {}
        data = {}
        data['start'] = mindata
        data['end'] = mindata + int2
        pricelist = []
        mindata = mindata + int2
        for j in range(len(a2)):
            if(int(a2[j][0])>=      int(data['start'])      and
int(a2[j][0])<= int(data['end'])):
                pricelist.append(a2[j][2])
                provider.append(a2[j][1])
        data['provider']                                   =
provider[pricelist.index(min(pricelist))]
        data['price'] = min(pricelist)
        p2[i] = data
    """Service 3"""
    provider = []
    mindata = pr3['min']
    for i in range(5):
        p3[i] = {}
        data = {}
        data['start'] = mindata
        data['end'] = mindata + int3
        pricelist = []
        mindata = mindata + int3
        for j in range(len(a3)):
            if(int(a3[j][0])>= int(data['start']) and
int(a3[j][0])<= int(data['end'])):
                pricelist.append(a3[j][2])
                provider.append(a3[j][1])
        data['provider'] =
provider[pricelist.index(min(pricelist))]
        data['price'] = min(pricelist)
        p3[i] = data
    """Service 4"""
    provider = []
    mindata = pr4['min']
    for i in range(5):
        p4[i] = {}
        data = {}
        data['start'] = mindata
        data['end'] = mindata + int4
        pricelist = []
        mindata = mindata + int4
        for j in range(len(a4)):
            if(int(a4[j][0])>= int(data['start']) and
int(a4[j][0])<= int(data['end'])):
                pricelist.append(a4[j][2])
                provider.append(a4[j][1])
        data['provider']                                      =
provider[pricelist.index(min(pricelist))]
        data['price'] = min(pricelist)
        p4[i] = data
    """Service 5"""
    provider = []
    mindata = pr5['min']
    for i in range(5):
        p5[i] = {}
        data = {}
        data['start'] = mindata
        data['end'] = mindata + int5
        pricelist = []
        mindata = mindata + int5
        for j in range(len(a5)):
            if(int(a5[j][0])>=     int(data['start'])     and
int(a5[j][0])<= int(data['end'])):
                pricelist.append(a5[j][2])
                provider.append(a5[j][1])
        data['provider']                                      =
provider[pricelist.index(min(pricelist))]
        data['price'] = min(pricelist)
        p5[i] = data


# conbine rabge for each service and store range
    model.append(p1)
    model.append(p2)
    model.append(p3)
    model.append(p4)
    model.append(p5)
    return model

def user_screen():
    model = mining_model()
    revenue = input("Enter your revenue : ")
    s = []
    s1 = input("Do you want service 1 : ")
    s.append(s1)
    s2 = input("Do you want service 2 : ")
    s.append(s2)
    s3 = input("Do you want service 3 : ")
    s.append(s3)
    s4 = input("Do you want service 4 : ")
    s.append(s4)
    s5 = input("Do you want service 5 : ")
    s.append(s5)
    result = mining_process(model,revenue,s)
    print "Your recommended Provider is : " + result

def mining_process(model,revenue,s):
    predict_price = []
```

```
     predict_provider = []

     for i in range(len(model)):

          if(s[i]==1):

               for  j in range(len(model[i])):

                    if(j==0 and revenue<=model[i][j]['end']):


     predict_price.append(int(model[i][j]['price']))


     predict_provider.append(model[i][j]['provider'])

                    elif(j==len(model[i])-1                   and
revenue>=model[i][j]['start']):


     predict_price.append(int(model[i][j]['price']))


     predict_provider.append(model[i][j]['provider'])

                    elif(revenue>model[i][j]['start']         and
revenue<model[i][j]['end']):


     predict_price.append(int(model[i][j]['price']))


     predict_provider.append(model[i][j]['provider'])

     # print predict_price

     # print predict_provider

     return
predict_provider[predict_price.index(min(predict_price))]

user_screen()
```

- **Responsibility of Team Member**

Since we were all new to data mining concepts, it was more of a team work and each put their maximum efforts to understand and build the project. We learned new techniques and languages such as R and Python and learned new data mining techniques and approaches.

- **References**

[1] Karim Chine (2010). Open science in the cloud: towards a universal platform for scientific and statistical computing. In: Furht B, Escalante A (eds) Handbook of cloud computing, Springer, USA, pp 453–474. ISBN 978-1-4419-6524-0.