Uber Drive 2016 Analysis

First name	Last Name	IIT Email	Same project in ITMD 527 (Yes or No)
Arpit	**Khandekar**	akhandekar@hawk.iit.edu	No
Yashasvi	**Sharma**	ysharma4@hawk.iit.edu	No
Neeraj	**Iyer**	niyer4@hawk.iit.edu	No

Presentation Link: **https://prezi.com/view/n7Z3KD55HmUSZAHj7DJI/**

TABLE OF CONTENTS:

1. INTRODUCTION

This dataset provides details of 'my Uber Drives' in year 2016. It provides analysis of ordinary Uber customer's behavior and contains Start Date, End Date, Start Location, End Location, Miles driven, purpose of drive, gender, status and rider's age.

The purpose of this project is to provide comprehensive analysis of Uber customer's behavior based on survey data taken from Kaggle.com. These days growing number of people are now getting engaged on Uber technology that helps people in getting what they want and when they want. While Uber and other companies like Lyft, yellow cabs in the gig economy receive a lot of attention, a deep observation of Uber's own data about its customers reveals them to be less consequential than most people assume. We wish to present the results of extensive analysis on Uber experiences across United states and application of specifics unique to consumer behavior. At the same time, we also wish to investigate a research and identify several problems that Uber customers are facing.

The results of this application provide useful information not only for Uber owner who may be able to develop more marketing strategies using consumer behavior like if there are less number of cabs available in particular area during peak hours then more cabs will be provided in those areas, but also for consumers who will be able to make better choices while booking cabs with different companies like fare comparison which will let them know how much can they save from a ride.

2. DATA

This dataset provides details of 'my Uber Drives' in year 2016. It provides analysis of ordinary Uber customer's behavior and contains Start Date, End Date, Start Location, End Location, Miles driven, purpose of drive, gender, status and rider's age. This data set was collected from: https://www.kaggle.com/zusmani/uberdrives. Data set size is approximately equal to 45 Mb.

Column Name	Description
1. Date	1. It describes the date when the cab was booked.
2. Ride Duration	2. It describes the time duration of the ride

3. Days	3. It describes whether the day was week day or weekend
4.Category	4. It describes if the type for which cab was booked i.e. whether it was personal or business
5. Start	5. It describes the starting area of ride.
6. Stop	6. It describes the area where the ride ends.
7. Miles	7. It describes total number of miles travelled in a ride.
8. Purpose	8. It describes the purpose of travel whether it is for entertainment, supplies or meeting.
9. Gender	9. It describes the gender of the personbooking the cab.
10. Status	10. It describes the status of the cab whether it was cancelled, Delay or On-time.
11. Rider's Age	11. It describes the age of the rider.
12. Fare	12. It describes the price for each ride
13. Peak hours	13. It describes peak hours when cabs are booked the most.

3. Data Processing:

The dataset available on the kaggle website has around 15,000 records. We found that many records were containing null values. Hence, to discard those records, we used the following code in python for identifying and replacing missing values:

- ➔ mydata.replace('?',-99999,inplace=True) # finding the missing data and replacing it by a value

We eliminated unwanted columns using the following python code:

- ➔ mydata.drop([''],1,inplace=True) # Dropping unwanted column

We changed nominal values in Gender to binary using excel where 1 is for men and 0 denotes women.

- ➔ mydata.Gender.replace('Male',1,inplace=True) # finding the Male data and replacing it by 1.

Nominal value Rider's Age in Status were also changed to binary values from nominal using excel.

The feature miles had a minimum value of 4 and maximum of more than 100. It required some normalization and hence we applied normalization using Min-Max normalization in Python using the below code:

- ➔ →minmax=preprocessing.MinMaxScaler(feature_range=(1,10)) # Will normalize miles between the range of 1 to 10.
- ➔ →minmax.fit(miles). transform(miles)

Feature Days was transformed to Weekday and Weekend using the following code:

- ➔ Mydata.replace('sunday','Weekend',inplace=True) #Days consist of Weekday and Weekend.

We have performed analysis on different sub-samples of this dataset of 1200 records by varying number of records, depending upon research problem. Also we have created START and STOP, and sorted different areas into 4 categories –"NORTH","SOUTH","EAST", "WEST".

4. PROBLEMS TO BE SOLVED

1) Average miles travelled is 10.4. A random sample of 40 customers is collected by using 95% confidence level, validate the hypothesis through at least 3 approaches.

2) For Business user's compare whether average fare is greater than average fare for Personal user's.

3) Using classification find which age group prefers more cabs (categories, purpose, gender and riders age)

4) Select the features which are used to estimate cab fare?

5) Perform a cross tabulation of two gender variables vs three purpose variables?

6) Use smoothing by bin means on Fare attribute.

7) Analyze the average ride duration using Linear Regression.

8) Using clustering analyze which age group uses more cabs based on the category.

9) To recommend the cabs based on drivers rating.

10) Analyze most number of rides taken in a month using Tableau/Power BI.

5. METHODS AND PROCESS

Solution 1) For a random sample of 200 miles records collected, we need to validate whether the average miles travelled is 10.4 miles or not using 95% confidence interval.

Below is the R code showing the summary of sample.

```
> PB1=read.csv("Q1.csv",header=TRUE,sep=",")
> summary(PB1)
        Date      Ride.Duration.mins.      Days            CATEGORY.     START.
 6/12/2016:  4   Min.   :10.00       Min.   :0.0000   Business:100   East :69
 10/6/2016:  3   1st Qu.:21.50       1st Qu.:0.0000   Personal: 99   North:50
 4/2/2016 :  3   Median :40.00       Median :0.0000                  South:35
 5/4/2016 :  3   Mean   :38.46       Mean   :0.4824                  West :44
 6/1/2016 :  3   3rd Qu.:47.50       3rd Qu.:1.0000                  West : 1
 6/5/2016 :  3   Max.   :75.00       Max.   :1.0000
 (Other)  :180
   STOP.         MILES.                    PURPOSE.      Gender
 East :43   Min.   : 0.500   Between Offices:51   Min.   :0.0000
 North:61   1st Qu.: 2.700   Errand/Supplies:60   1st Qu.:0.0000
 South:63   Median : 6.000   Meal/Entertain :88   Median :1.0000
 West :32   Mean   : 7.966                        Mean   :0.7286
            3rd Qu.:10.400                        3rd Qu.:1.0000
            Max.   :63.700                        Max.   :1.0000

 Status.Cancelled  Status.Delay    Status.On.time   Rider.Age.Young
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
 Median :0.0000   Median :0.0000   Median :0.0000   Median :0.0000
 Mean   :0.3769   Mean   :0.1859   Mean   :0.4372   Mean   :0.2111
 3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:0.0000
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000

 Rider.Age.Middle Rider.Age.Old         Fare                  Peak.hours
 Min.   :0.0000   Min.   :0.0000   Min.   : 4.00              8:00PM : 16
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:12.00   10:00 AM          : 16
 Median :1.0000   Median :0.0000   Median :20.00   10:30 AM          : 16
 Mean   :0.5377   Mean   :0.2513   Mean   :21.91   7:00 PM           : 16
 3rd Qu.:1.0000   3rd Qu.:0.5000   3rd Qu.:32.00                9:00AM: 15
 Max.   :1.0000   Max.   :1.0000   Max.   :40.00   6:00 PM           : 15
                                                   (Other)           :105

 Drivers.Rating
 Min.   :1.000
 1st Qu.:2.000
 Median :3.000
 Mean   :3.005
 3rd Qu.:4.000
 Max.   :5.000
```

Therefore from the calculation from R,

Mean of 200 records is: 7.966

So, our assumption is

Null Hypothesis: Average miles traveled by an Uber customer is 10.4 miles.

Alternate Hypothesis: Average miles traveled by an Uber customer is not 10.4miles.

We also think that average mile travelled is not 10.4

H0 = μ=10.4

HA = μ ≠ 10.4

Since the sample size is large so we should use z-distribution.

```
> utils:::menuInstallPkgs()
--- Please select a CRAN mirror for use in this session ---
trying URL 'https://cran.revolutionanalytics.com/bin/windows/contrib/3.4/BSDA_1.2.0.zip'
Content type 'application/zip' length 847525 bytes (827 KB)
downloaded 827 KB

package 'BSDA' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\user\AppData\Local\Temp\RtmpEDYwcX\downloaded_packages
> library(BSDA)
Loading required package: lattice

Attaching package: 'BSDA'

The following object is masked from 'package:datasets':

    Orange

>
```

```
> MILES=PB1$MILES
> z.test(MILES,NULL,alternative="two.sided",mu=10.4,sigma.x=sd(MILES),conf.level=0.95)

        One-sample z-Test

data:  MILES
z = -4.5153, p-value = 6.323e-06
alternative hypothesis: true mean is not equal to 10.4
95 percent confidence interval:
 6.909941 9.022723
sample estimates:
mean of x
 7.966332
```

From above we see that p-value is 6.323e-06 which is less than 0.05 which means it falls in rejection region and we need to reject null hypothesis and accept alternate hypothesis. Therefore, Average miles travelled by Uber customer is not 10.4 miles.

Hence, we conclude that at 95% confidence level that there is sufficient evidence that average miles travelled is not equal to 10.4 miles.

Solution 2) It is two sample independent hypothesis testing. Also, it is a two-tailed test. We will mention null and alternate hypothesis. Since sample size is greater than 30 so we will use z-distribution hypothesis testing to find whether mean fare of Business users is greater than mean fare of personal users. It is assumed that mean fare for both users is same, but we do not think so.

- Fare of Customers using Uber for Business use
  Sample size:80

- Fare of Customers using Uber for Personal use
  Sample size:90

Loading business user data and finding Standard deviation:

```
>
>
> Business=read.csv("Q2_business_vs_fare.csv",header=TRUE,sep=",")
> summary(Business)
    CATEGORY.        Fare
 Business:89   Min.   : 4.0
               1st Qu.:12.0
               Median :20.0
               Mean   :21.8
               3rd Qu.:32.0
               Max.   :40.0
> B_Fare=Business$Fare
> summary(B_Fare)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    4.0    12.0    20.0    21.8    32.0    40.0
> B_Fare_sd=sd(B_Fare)
> summary(B_Fare_sd)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  11.46   11.46   11.46   11.46   11.46   11.46
> B_Fare_sd
[1] 11.45761
>
```

Here Standard Deviation (sd) is **11.45761**

Now we will load personal user data and find sd:

```
> B_Fare=Q2_business_vs_fare$Fare
Error: object 'Q2_business_vs_fare' not found
> Personal=read.csv("Q2_personal_vs_fare1.csv",header=TRUE,sep=",")
> summary(Personal)
    CATEGORY.       Fare
 Personal:89   Min.   : 4.0
               1st Qu.:12.0
               Median :20.0
               Mean   :21.8
               3rd Qu.:32.0
               Max.   :40.0
> P_Fare=Personal$Fare
> summary(P_Fare)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    4.0    12.0    20.0    21.8    32.0    40.0
> P_Fare_sd=sd(P_Fare)
> P_Fare
 [1]  4 12  8 20 16 28 24 36 32 40  4 12  8 20 16 28 24 36 32 40  4 12  8 20 16 28 24 36 32 40  4 12  8 20 16 28 24 36 32 40  4 12  8 20 16 28 24 36 32 40  4 12  8 20 16 28 24 36 32 40  4
[62] 12  8 20 16 28 24 36 32 40  4 12  8 20 16 28 24 36 32 40  4 12  8 20 16 28 24 36 32
> P_Fare_sd
[1] 11.45761
> library(psych)
> describe(P_Fare)
   vars  n mean    sd median trimmed   mad min max range skew kurtosis   se
X1    1 89 21.8 11.46     20   21.75 11.86   4  40    36 0.01    -1.25 1.21
>
```

Here sd is **11.45761**

Null Hypothesis: Average fare for business users is same as the average fare for personal users.

Alternate Hypothesis: Average fare for business users is not same as the average fare for personal users.

$\mu 1$ = Average fare for business users.

$\mu 2$= Average fare for personal users.

H0 = $\mu 1 = \mu 2$

HA = $\mu 1 \neq \mu 2$

i.e

H0 = $\mu 1 - \mu 2 = 0$ i.e $\mu d = 0$

HA = $\mu 1 - \mu 2 \neq 0$ i.e $\mu d \neq 0$

```
> z.test(B_Fare,P_Fare,alternative="two.sided",mu=0,sigma.x=sd(B_Fare),sigma.y=sd(P_Fare),conf.level=0.95)

        Two-sample z-Test

data:  B_Fare and P_Fare
z = 0, p-value = 1
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.366372  3.366372
sample estimates:
mean of x mean of y
 21.79775  21.79775
```

Here p-value is 1 which is greater than 0.05. So we will accept Null Hypothesis. It states that average fare is same for both business and personal category.

Hence, we conclude that at 95% confidence level that there is sufficient evidence that average fare for both business users and personal users is same.

Solution 3) Classification:

1) KNN:

This problem we solved using Python. Below is the code, we used to perform KNN classification.

#import libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

 # importing our csv dataset

mydata=pd.read_csv('classification_problem_Full.csv')

X=mydata.iloc[:,[0,1,2]].values # Category, Gender, Purpose Features

```
y=mydata.iloc[:,[3]].values        # Label -> Age {Y,M,O}
```

 # splitting data into training and testing data (75:25)

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.25, random_state=0)
```

 # Feature scaling

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

#Classification algorithm area

```
from sklearn import neighbors
classifier=neighbors.KNeighborsClassifier(n_neighbors=3,weights='distance')   #K value=3
classifier.fit(X_train,y_train)
```

 # get output of test results

```
y_pred=classifier.predict(X_test)
#y_pred=classifier.predict([[1,1,1]])
print(y_pred)
```

#create confusion matrix

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
```

Output:

Prediction for input [0,1,1]:

Category=Personal, Purpose = Between offices, Gender = Male

```
...: # Feature Scaling
...: from sklearn.preprocessing import StandardScaler
...: sc=StandardScaler()
...: X_train=sc.fit_transform(X_train)
...: X_test=sc.transform(X_test)
...:
...: #Classification algorithm area
...:
...: from sklearn import neighbors
...: classifier=neighbors.KNeighborsClassifier()
...: classifier.fit(X_train,y_train)
...:
...: # get output of test results
...:
...: #y_pred=classifier.predict(X_test)
...: y_pred=classifier.predict([[0,1,1]])
...: print(y_pred)
['M']
```

OUTPUT:

Prediction for input [0,1,1] → 'M' {Middle Age}

2) Naïve Bayes Classification:

Below shown is the code:

```
# -*- coding: utf-8 -*-

"""

Created on Thu Apr 19 00:31:27 2018
```

```
@author: neera

#import libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd


# importing our csv dataset

mydata=pd.read_csv('classification_problem_Full.csv')

X=mydata.iloc[:,[0,1,2]].values   # Category, Gender, Purpose Features

y=mydata.iloc[:,[3]].values        # Label -> Age {Y,M,O}


# splitting data into training and testing data [75:25]

from sklearn.cross_validation import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.25, random_state=0)


# Feature scaling

from sklearn.preprocessing import StandardScaler

sc=StandardScaler()

X_train=sc.fit_transform(X_train)

X_test=sc.transform(X_test)


#Classification algorithm area

from sklearn.naive_bayes import GaussianNB

classifier=GaussianNB()

classifier.fit(X_train,y_train)
```

```
# get output of test results

y_pred=classifier.predict([[1,1,1]])   # predicting output for [1,1,1,]

print(y_pred)
```

OUTPUT:

Prediction for input [1,1,1]:

Category=Business, Purpose = Between offices, Gender = Male

```
...: # Feature scaling
...: from sklearn.preprocessing import StandardScaler
...: sc=StandardScaler()
...: X_train=sc.fit_transform(X_train)
...: X_test=sc.transform(X_test)
...:
...: #Classification algorithm area
...:
...: from sklearn.naive_bayes import GaussianNB
...: classifier=GaussianNB()
...: classifier.fit(X_train,y_train)
...:
...: # get output of test results
...:
...: #y_pred=classifier.predict(X_test)
...: y_pred=classifier.predict([[1,1,1]])
...: print(y_pred)
['M']
```

OUTPUT:-

Prediction for input [1,1,1]: → 'M' {Middle Age}

3) Decision Trees Classification:

Below shown is the code:

#import libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import tree
```

 # importing our csv dataset

```
mydata=pd.read_csv('classification_problem.csv')
X=mydata.iloc[:,[0,1,2]].values   # loading Category,Gender,Purpose
y=mydata.iloc[:,[3]].values         # Loading Age {Y,M,O}
```

 # splitting data into training and testing data (75:25)

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.25, random_state=0)
```

 # Feature scaling

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

#Classification algorithm area

```
from sklearn import tree
```

classifier=tree.DecisionTreeClassifier(criterion='gini')

clf=classifier.fit(X_train,y_train)

 # get output of test results

y_pred=classifier.predict([[0,0,0]])

print(y_pred)

OUTPUT:-

Prediction for input [0,0,0]:

Category=Personal, Purpose = Meal/Entertainment, Gender = Female

```
.... # Feature Scaling
...: from sklearn.preprocessing import StandardScaler
...: sc=StandardScaler()
...: X_train=sc.fit_transform(X_train)
...: X_test=sc.transform(X_test)
...:
...: #Classification algorithm area
...:
...: from sklearn import tree
...: classifier=tree.DecisionTreeClassifier(criterion='gini')
...: clf=classifier.fit(X_train,y_train)
...: #classifier.predict([[1,1,1]])
...:
...: # get output of test results
...:
...: #y_pred=classifier.predict(X_test)
...: y_pred=classifier.predict([[0,0,0]])
...: print(y_pred)
...:
['Y']
```

Prediction for input [0,0,0] → 'Y' {Young Age}

4.) Support Vector Machine (SVM):

Below is the code for same:

```
# -*- coding: utf-8 -*-
"""
Created on Sun Apr 22 16:17:23 2018
@author: neera
"""
# -*- coding: utf-8 -*-
"""
Created on Thu Apr 19 00:31:27 2018
```

#import libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

 # importing our csv dataset

```
mydata=pd.read_csv('classification_problem_Full.csv')
X=mydata.iloc[:,[0,1,2]].values  # loading Category,Gender,Purpose
y=mydata.iloc[:,[3]].values        # Loading Age feature {Y,M,O}
```

 # splitting data into training and testing data (75:25)

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.25, random_state=0)
```

```
# Feature scaling
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)

#Classification algorithm area
from sklearn import svm
classifier=svm.SVC()
classifier.fit(X_train,y_train)

# get output of test results
y_pred=classifier.predict([[1,0,0]])
print(y_pred)
```

OUTPUT:-

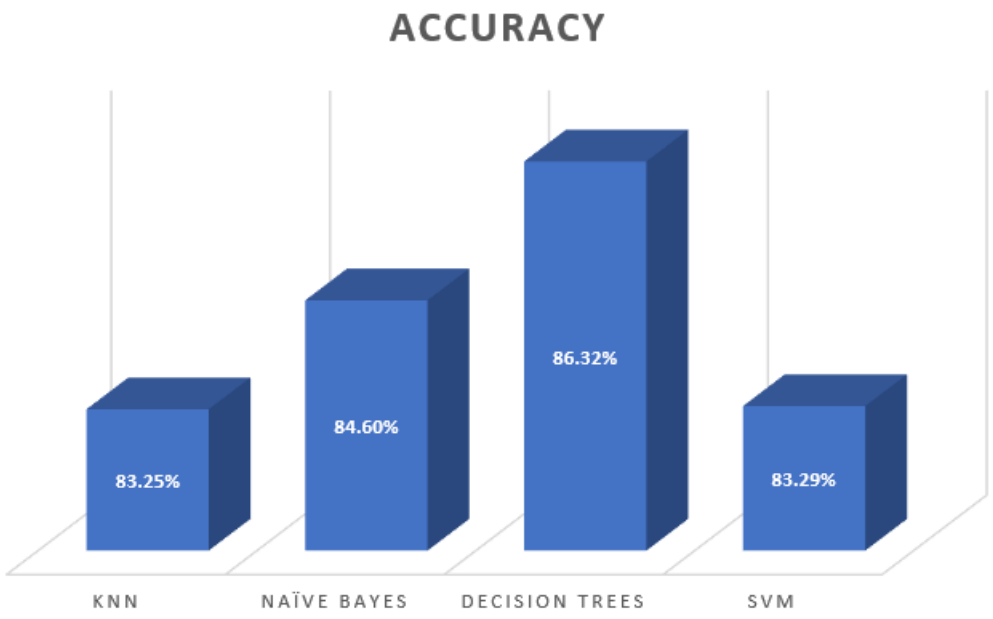
Prediction for input [0,0,0]:

Category=Personal, Purpose = Between offices, Gender = Female

```
...: # Feature scaling
...: from sklearn.preprocessing import StandardScaler
...: sc=StandardScaler()
...: X_train=sc.fit_transform(X_train)
...: X_test=sc.transform(X_test)
...:
...: #Classification algorithm area
...:
...: from sklearn import svm
...: classifier=svm.SVC()
...: classifier.fit(X_train,y_train)
...:
...: # get output of test results
...:
...: #y_pred=classifier.predict(X_test)
...: y_pred=classifier.predict([[1,0,0]])
...: print(y_pred)
...:
['O']
```

Prediction for input [0,0,0] → 'O' {Old Age}

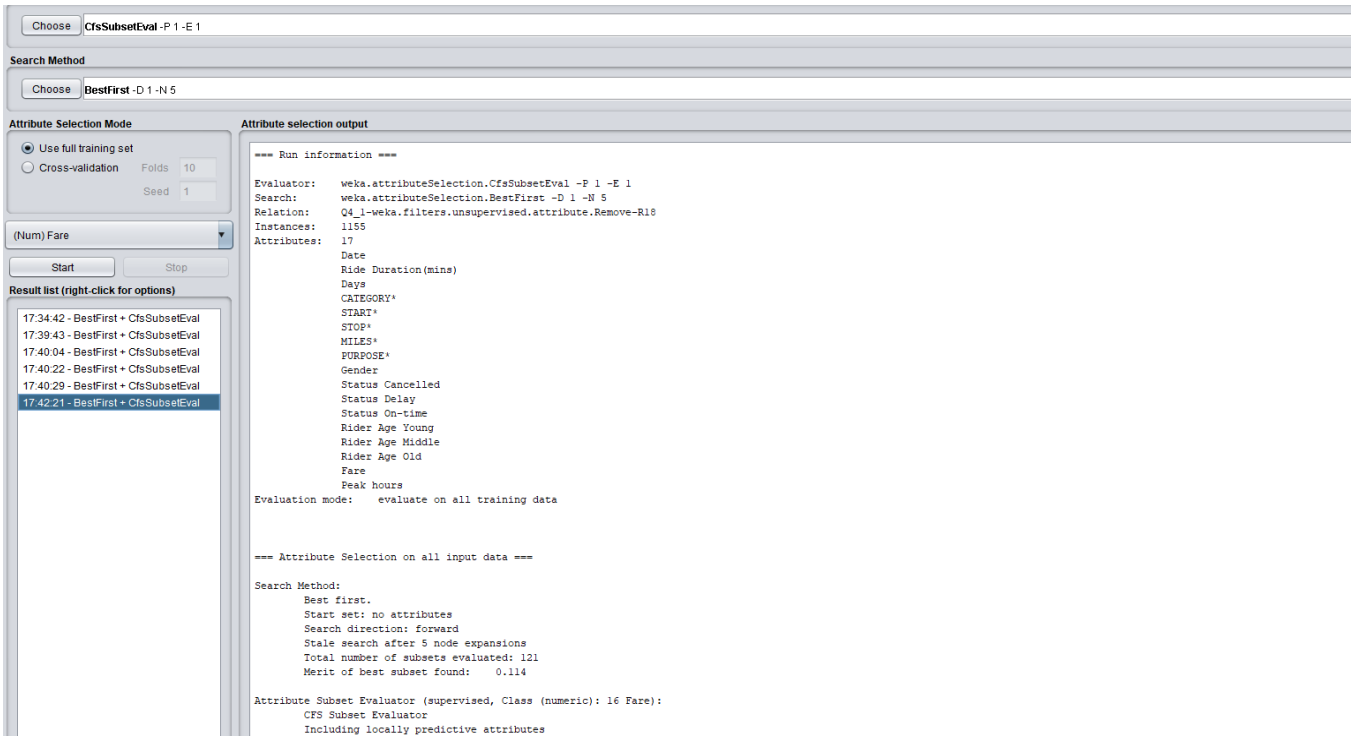
The accuracy obtained for all the 4 algorithms are as below:

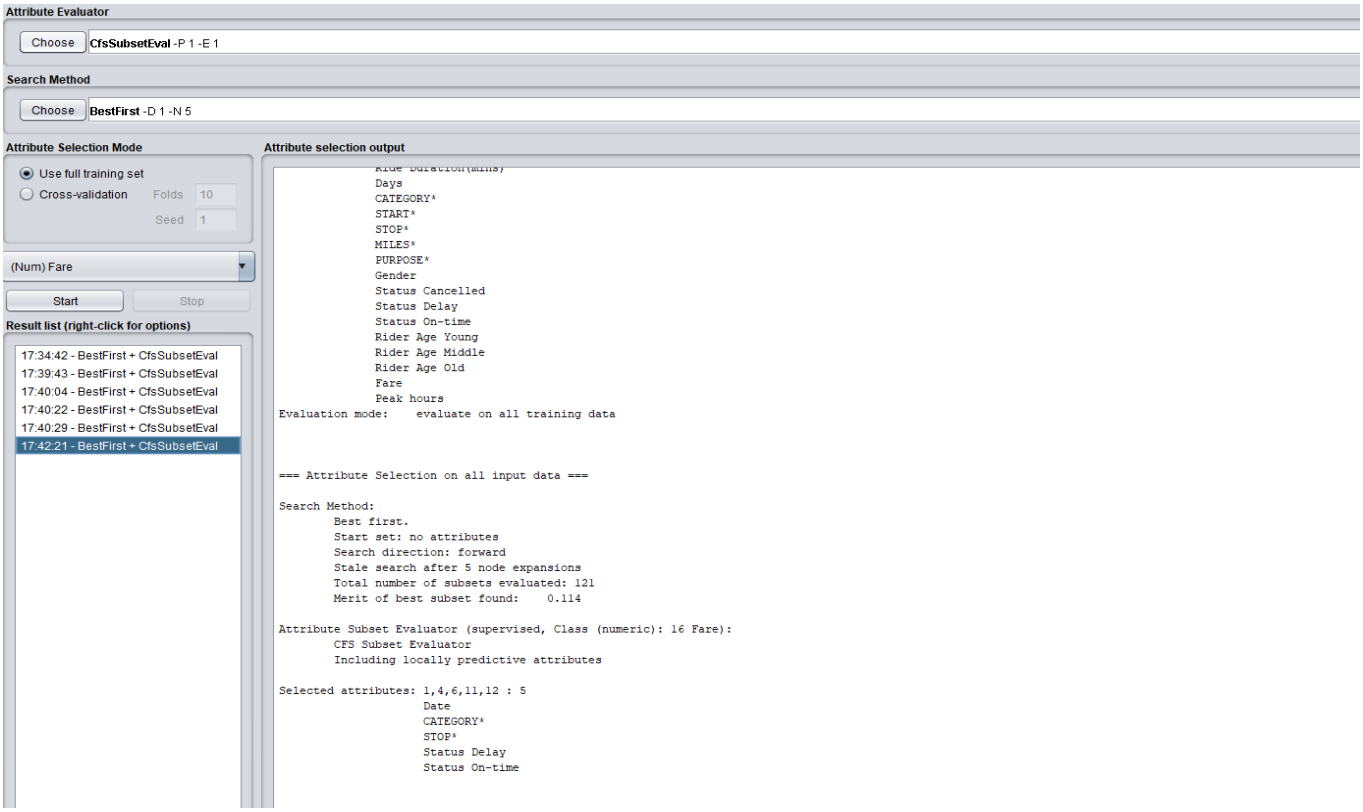


From the graph we can say that Decision tree gives the best Accuracy.

Solution 4) (FEATURE SELECTION)

It is features selection and feature reduction problem where we will eliminate the features which are not useful for finding the cab fare. For example: we will remove the feature gender to calculate cab fares as fare is not dependent on the person who is travelling.





From Feature selection we have found out that,

The features essential for estimating cab fare is:

Date

Category

Stop

Status

Delay

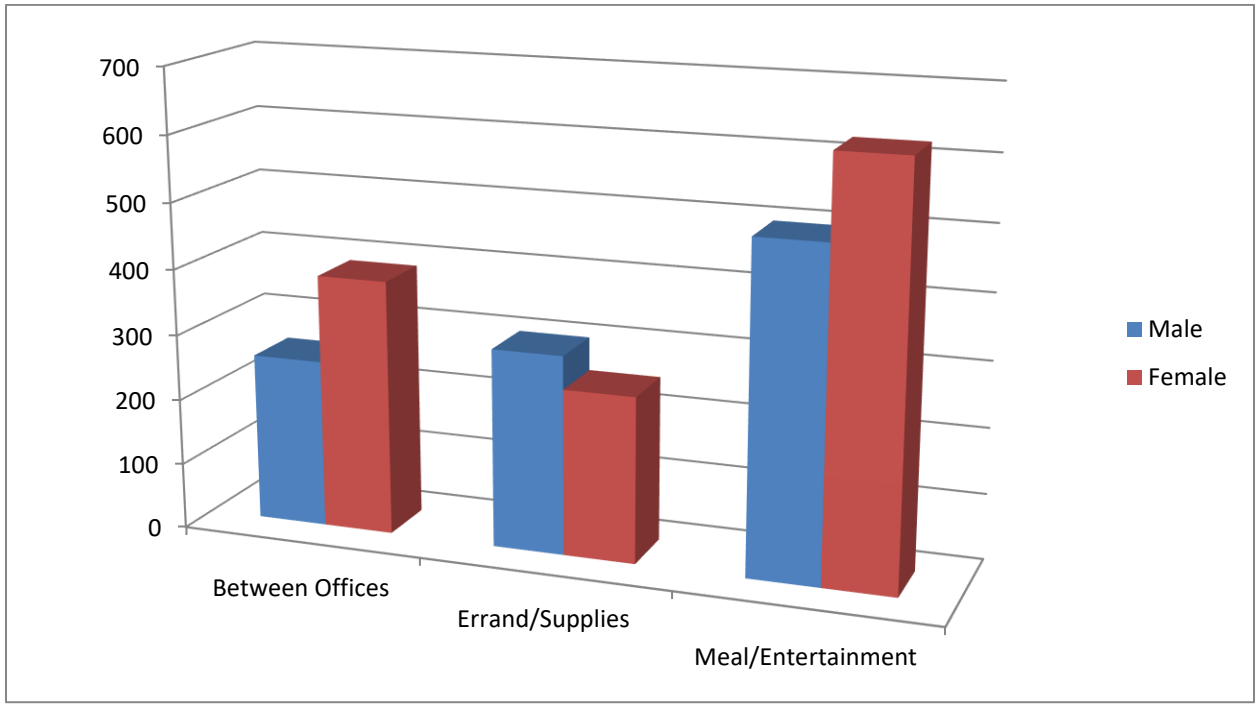
On-Time

Solution 5) (CROSS TABULATION)

Cross tabulation of two gender variables vs three purpose variables.

	Between Offices	Errand/Supplies	Meal/Entertainment
Male	256	302	498
Female	388	252	624

Now we will draw bar graph using above table:



From above graph we conclude that:

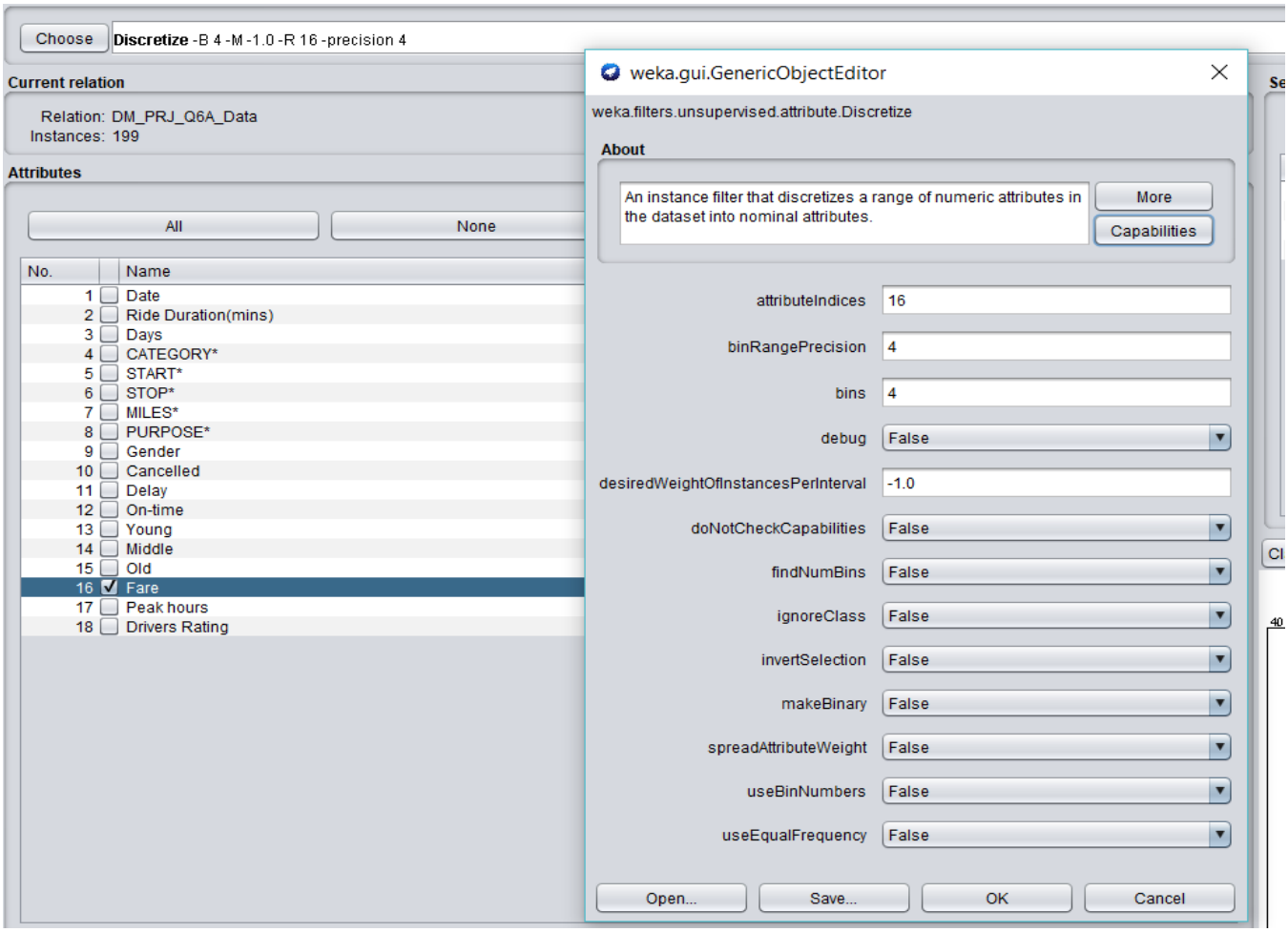
1) Females prefer Uber maximum for using it between offices.

2) Males prefer Uber maximum for using it In Errand/Supplies.

3) Females prefer Uber maximum for using It for Meal/Entertainment.

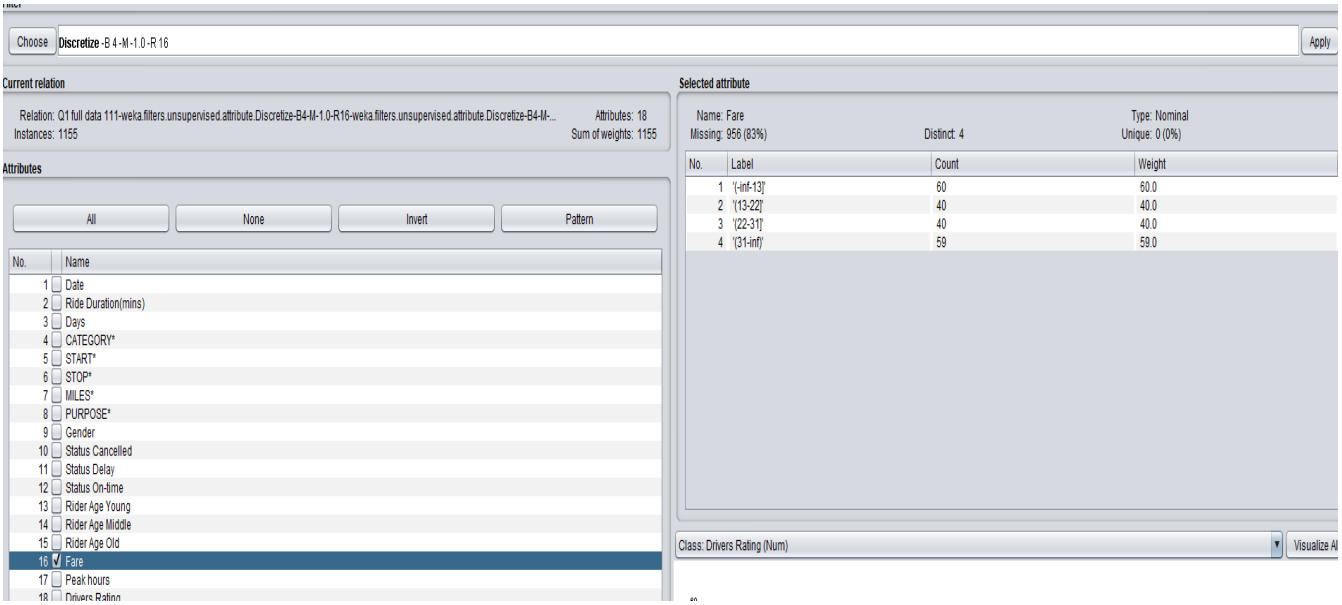
Solution 6): (DATA PREPROCESSING)

Using Smoothing by bin means on Fare attribute through weka.



In this we did smoothing by bin means method on fare attribute. We will divide the date in the bin of 4 and then replace by the mean value.

The bin depth is 9 which means number of values in one bin is 9.



Above result shows smoothing by bin means method on fare attribute.

Solution 7) (MULTIPLE LINEAR REGRESSION)

Average ride duration Using Linear Regression:

Below is the code:

```
library(pROC)

library(corrplot)

library(caTools)

library(caret)

library(e1071)

library(rpart)

library(ROCR)

library(reshape2)
```

```
library(dplyr)

library(tidyr)

library(magrittr)

library(BSDA)
```

#Setting working directory

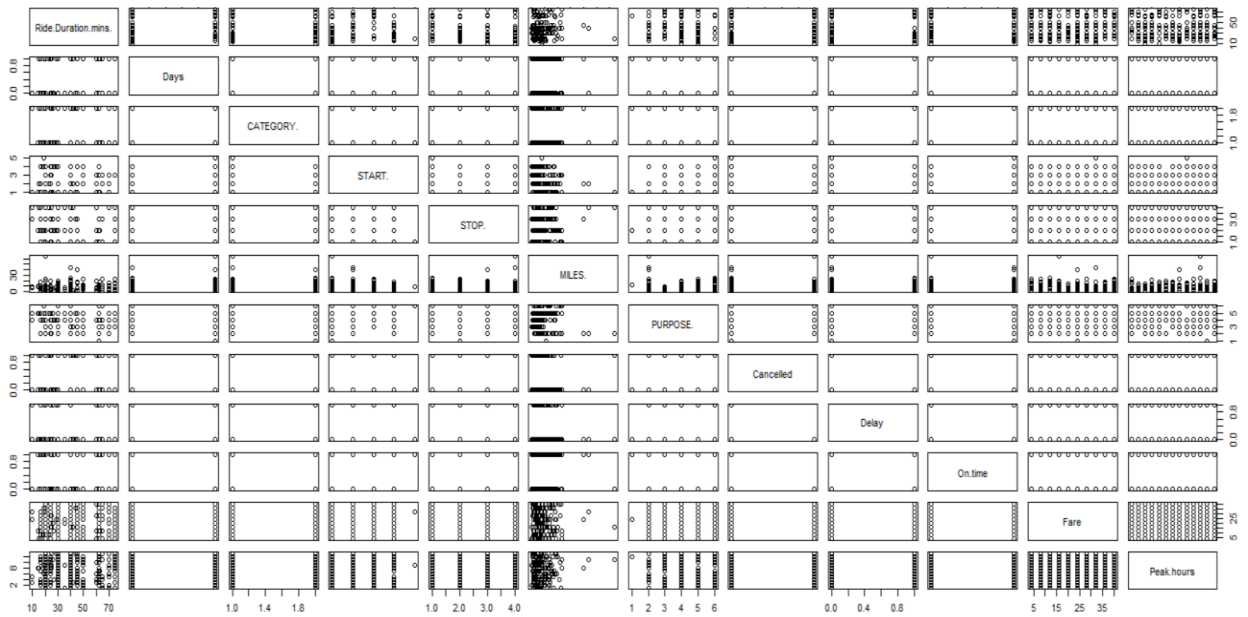
```
workdir<-"F:\Arpit\Arpit_Stuff1"

setwd(workdir)
```

 # Import DM_PRJ_Q7_DATA.csv

```
AvgFare=read.csv("DM_PRJ_Q7_DATA.csv", header=TRUE,sep=",")
```

 # Summary for AvgFare

```
summary(AvgFare)

Plot(AvgFare)
```



#Defining variables.

```
Duration=AvgFare$Ride.Duration

Days=AvgFare$Days

Days=AvgFare$Days

Category=AvgFare$CATEGORY

Start=AvgFare$START

Stop=AvgFare$STOP

Miles=AvgFare$MILES

purpose=AvgFare$PURPOSE

peakhours=AvgFare$Peak.hours
```

 # To build a model(fit)

```
fit=lm(Duration~ Days+ Category+ Start+Stop+ Miles+ purpose+ peakhours,data=
AvgFare)

summary(fit)
```

```
> fit=lm(Duration~ Days+ Category+ Start+Stop+ Miles+ purpose+ peakhours,data= AvgFare)
> summary(fit)

Call:
lm(formula = Duration ~ Days + Category + Start + Stop + Miles +
    purpose + peakhours, data = AvgFare)

Residuals:
    Min      1Q  Median      3Q     Max
-35.641 -12.806  -2.006  10.846  33.674

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)            35.249446   4.402759   8.006 1.27e-13 ***
Days                    2.031265   2.331379   0.871   0.3847
Categorypersonal       -2.985784   2.388045  -1.250   0.2128
StartNorth              6.763149   3.284231   2.059   0.0409 *
StartSouth             19.301841   3.411471   5.658 5.75e-08 ***
StartWest              -2.996901   3.262689  -0.919   0.3595
StartWest             -19.337758  16.370601  -1.181   0.2390
StopNorth              -1.674381   3.627390  -0.462   0.6449
StopSouth              -4.057502   3.287907  -1.234   0.2187
StopWest                2.592926   3.707453   0.699   0.4852
Miles                  -0.007079   0.153972  -0.046   0.9634
purposeErrand/Supplies -1.126160   3.074059  -0.366   0.7145
purposeMeal/Entertain  -0.783587   2.859915  -0.274   0.7844
peakhoursMorning        1.899387   2.377050   0.799   0.4253
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.85 on 185 degrees of freedom
Multiple R-squared:  0.248,     Adjusted R-squared:  0.1952
F-statistic: 4.694 on 13 and 185 DF,  p-value: 5.462e-07
```

 # Next we will build a base model (only one dependent and independent variable)

basemodel= lm(Duration~Miles)

```
> summary(basemodel)

Call:
lm(formula = Duration ~ Miles)

Residuals:
    Min      1Q  Median      3Q     Max
-28.466 -16.670   0.952   8.748  37.153

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 37.71050    1.82263   20.69   <2e-16 ***
Miles        0.09098    0.16554    0.55    0.583
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 17.7 on 197 degrees of freedom
Multiple R-squared:  0.001531,  Adjusted R-squared:  -0.003537
F-statistic: 0.3021 on 1 and 197 DF,  p-value: 0.5832
```

 # Next steps is to build a forward model using step function

forwardmodel<- step (basemodel,direction="forward", trace= T)

summary(forwardmodel)

```
> forwardmodel<- step(basemodel,direction="forward", trace= T)
Start:  AIC=1145.62
Duration ~ Miles

> summary(forwardmodel)

Call:
lm(formula = Duration ~ Miles)

Residuals:
    Min      1Q  Median      3Q     Max
-28.466 -16.670   0.952   8.748  37.153

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 37.71050    1.82263   20.69   <2e-16 ***
Miles        0.09098    0.16554    0.55    0.583
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 17.7 on 197 degrees of freedom
Multiple R-squared:  0.001531,  Adjusted R-squared:  -0.003537
F-statistic: 0.3021 on 1 and 197 DF,  p-value: 0.5832
```

 # Build a backward model using step function

backwardmodel<- step(fit,direction="backward", trace= T)

summary(backwardmodel)

```
> backwardmodel<- step(fit,direction="backward", trace= T)
Start:  AIC=1113.2
Duration ~ Days + Category + Start + Stop + Miles + purpose +
    peakhours

            Df Sum of Sq   RSS    AIC
- purpose    2      35.8 46506 1109.3
- Miles      1       0.5 46471 1111.2
- Stop       3    1014.7 47485 1111.5
- peakhours  1     160.4 46631 1111.9
- Days       1     190.7 46661 1112.0
- Category   1     392.7 46863 1112.9
<none>                   46470 1113.2
- Start      4   11655.8 58126 1149.7

Step:  AIC=1109.35
Duration ~ Days + Category + Start + Stop + Miles + peakhours

            Df Sum of Sq   RSS    AIC
- Miles      1       0.1 46506 1107.3
- Stop       3     985.8 47492 1107.5
- peakhours  1     140.2 46646 1108.0
- Days       1     175.6 46682 1108.1
- Category   1     388.2 46894 1109.0
<none>                   46506 1109.3
- Start      4   11763.9 58270 1146.2

Step:  AIC=1107.35
Duration ~ Days + Category + Start + Stop + peakhours

            Df Sum of Sq   RSS    AIC
- Stop       3     994.4 47500 1105.6
- peakhours  1     140.5 46647 1106.0
- Days       1     175.6 46682 1106.1
- Category   1     388.2 46894 1107.0
<none>                   46506 1107.3
- Start      4   11776.1 58282 1144.3

Step:  AIC=1105.56
Duration ~ Days + Category + Start + peakhours

            Df Sum of Sq   RSS    AIC
- Days       1     201.2 47702 1104.4
- peakhours  1     210.1 47711 1104.4
```

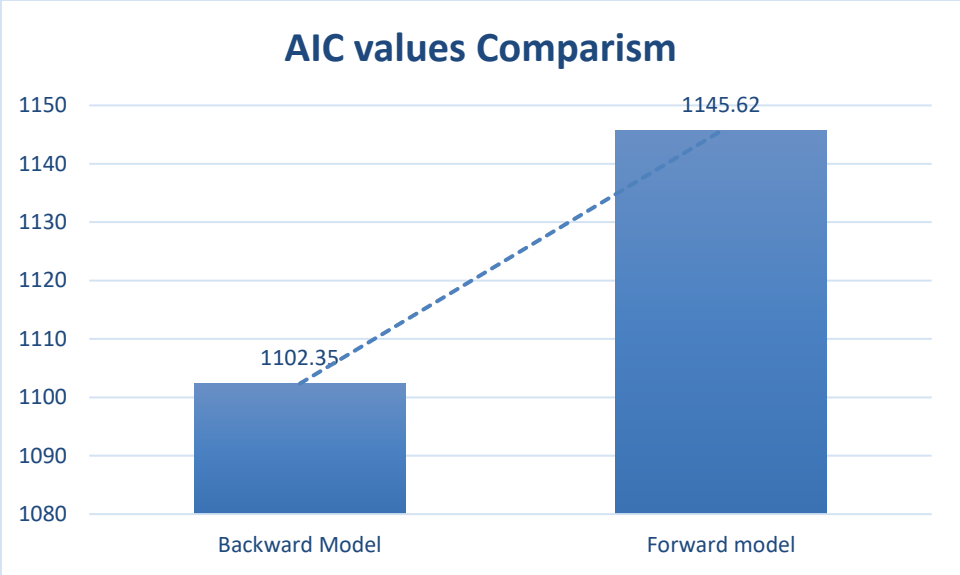
```
Step:  AIC=1105.56
Duration ~ Days + Category + Start + peakhours

            Df Sum of Sq   RSS    AIC
- Days       1     201.2 47702 1104.4
- peakhours  1     210.1 47711 1104.4
- Category   1     323.6 47824 1104.9
<none>                  47500 1105.6
- Start      4   13101.3 60602 1146.0

Step:  AIC=1104.4
Duration ~ Category + Start + peakhours

            Df Sum of Sq   RSS    AIC
- peakhours  1     175.4 47877 1103.1
- Category   1     324.0 48026 1103.8
<none>                  47702 1104.4
- Start      4   13430.1 61132 1145.8

Step:  AIC=1103.13
Duration ~ Category + Start

           Df Sum of Sq   RSS    AIC
- Category  1     292.6 48170 1102.3
<none>                 47877 1103.1
- Start     4   13387.0 61264 1144.2

Step:  AIC=1102.35
Duration ~ Start

        Df Sum of Sq   RSS    AIC
<none>               48170 1102.3
- Start  4     13627 61797 1143.9
> 
```

```
> summary(backwardmodel)

Call:
lm(formula = Duration ~ Start, data = AvgFare)

Residuals:
   Min     1Q Median     3Q    Max
-33.74 -13.74  -1.50  10.46  35.66

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   34.343      1.883  18.235  < 2e-16 ***
StartNorth     7.157      2.918   2.453   0.0151 *
StartSouth    19.400      3.262   5.947 1.25e-08 ***
StartWest     -4.808      3.053  -1.575   0.1169
StartWest    -15.343     15.870  -0.967   0.3348
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.76 on 194 degrees of freedom
Multiple R-squared:  0.2205,    Adjusted R-squared:  0.2044
F-statistic: 13.72 on 4 and 194 DF,  p-value: 7.166e-10
```

We have compared different models with the help of AIC values.

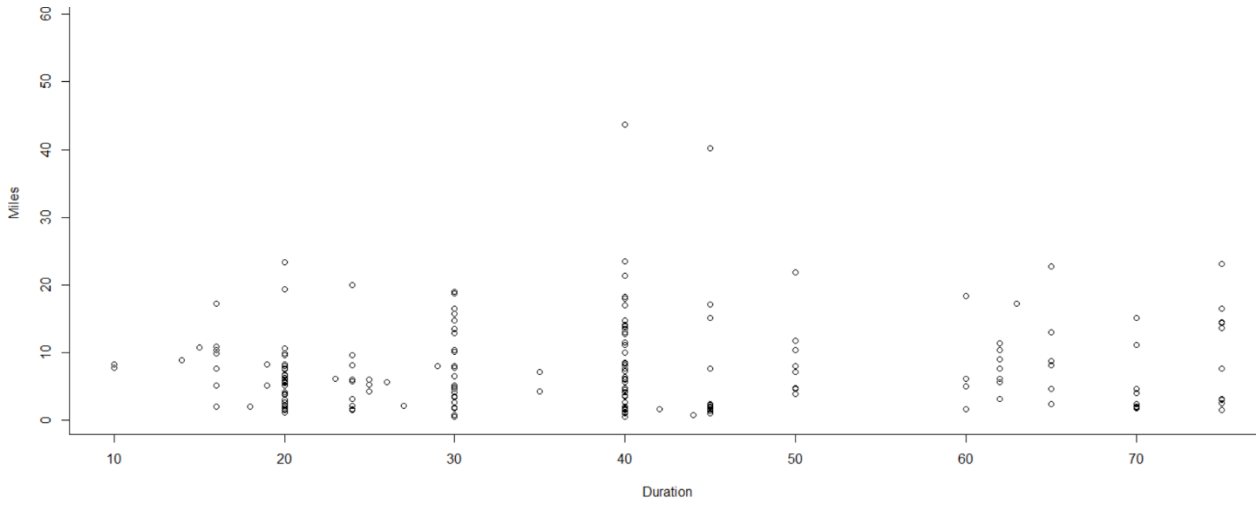
We found that AIC value is less for backward model compared to forward model, hence model having lowest AIC is better model, in our case backward model is better than forward model.

We have drawn a bar graph showing comparison of both:

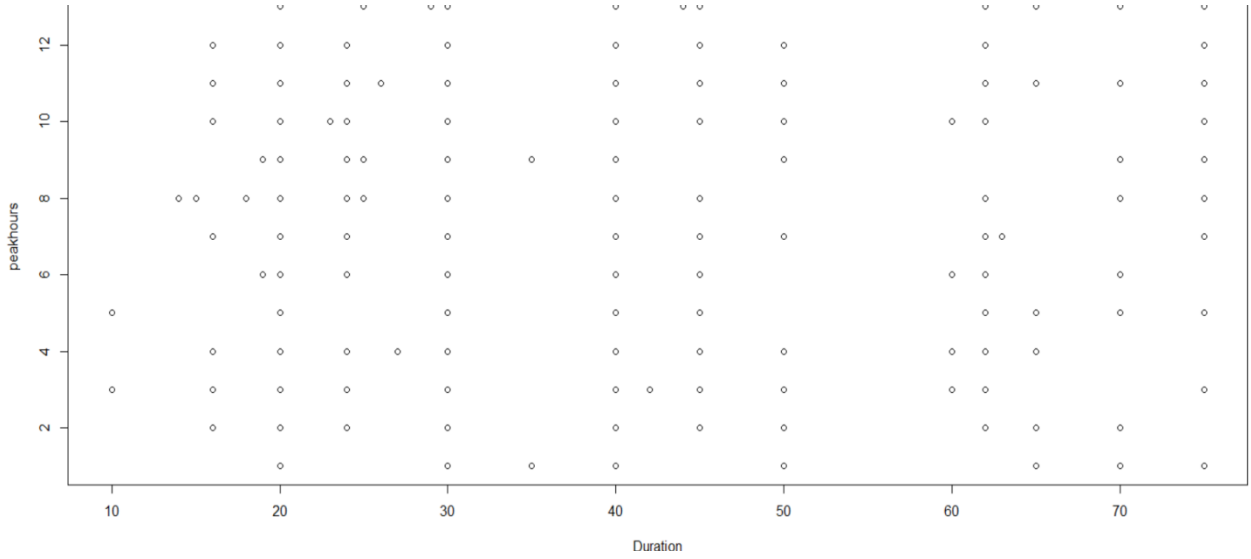


#We will examine the relationship between duration and miles,

Plot(Duration,Miles)



Plot(Duration,Peakhours)



Solution 8)

K-Means Clustering:

#importing libraries

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from matplotlib.colors import ListedColormap

from sklearn.cluster import KMeans
```

 # importing our csv dataset

```
mydata=pd.read_csv('DM_PR8.csv')

X=mydata.iloc[:,[6,13]].values   # loading Category and Age
```

 # Find out the best number of clusters (Using Elbow method)

```
Array=[]     # to store sum of squares within the groups
```

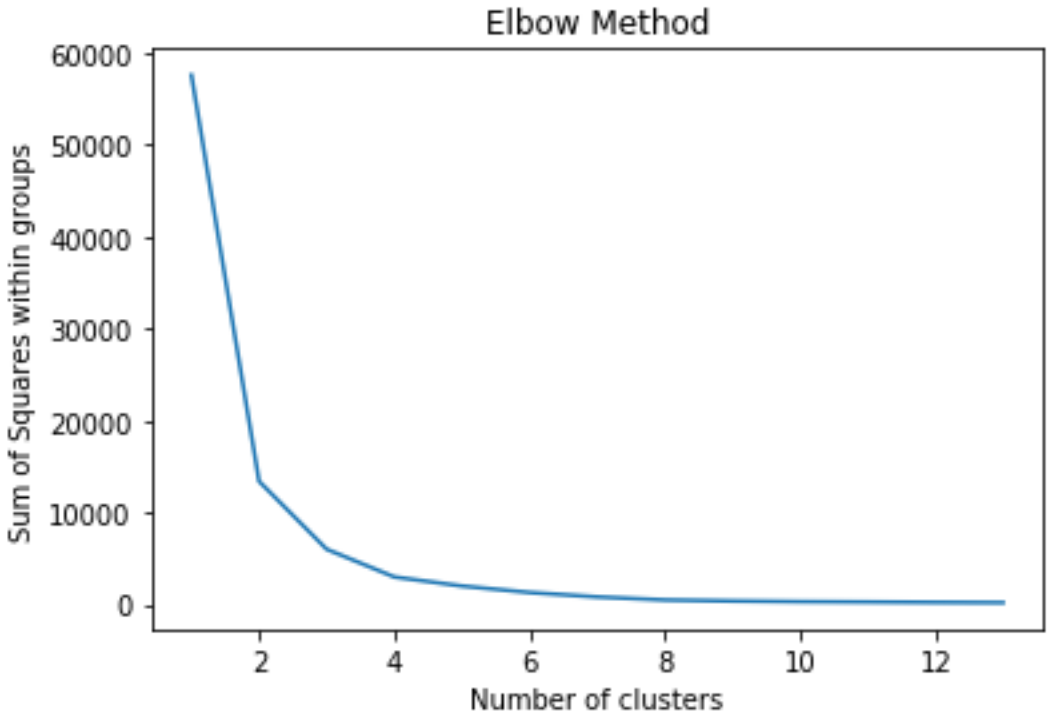
```
for i in range(1,14):
    kmeans=KMeans(n_clusters=i,init='k-means++',random_state=0)
    kmeans.fit(X)
    Array.append(kmeans.inertia_)    # inertia --> Sum of squared distances of samples to their closest cluster center
plt.plot(range(1,14),Array)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('Sum of Squares within groups')
plt.show()
```

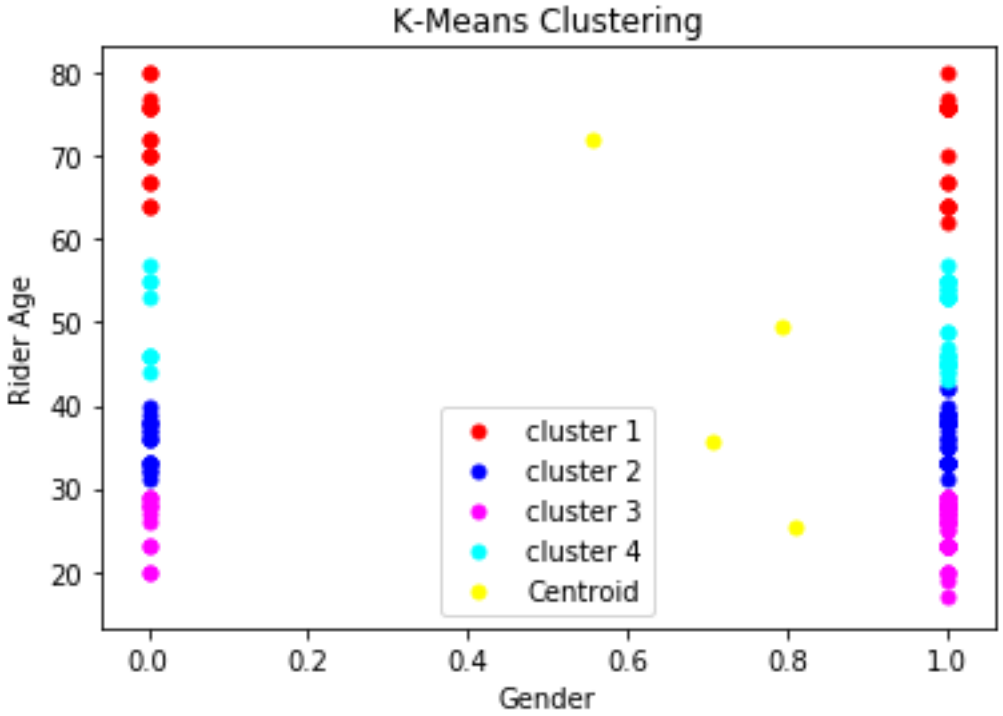
 # K-Means clustering algorithm on Rider age and Gender using Uber cabs data

```
kmeans=KMeans(n_clusters=4,init='k-means++',random_state=0)

# fit function will give output of kmeans but fit_predict will give the cluster index for each sample
Y=kmeans.fit_predict(X)
plt.scatter(X[Y == 0,0], X[Y == 0,1],s=25,c='red',label='cluster 1')   #s --> zoom level
plt.scatter(X[Y == 1,0], X[Y == 1,1],s=25,c='blue',label='cluster 2')
plt.scatter(X[Y == 2,0], X[Y == 2,1],s=25,c='magenta',label='cluster 3')
plt.scatter(X[Y == 3,0], X[Y == 3,1],s=25,c='cyan',label='cluster 4')
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],s=25,c='yellow',label='Centroid')
plt.title('K-Means Clustering')
plt.xlabel('Gender')
```

plt.ylabel('Rider Age')

plt.legend()



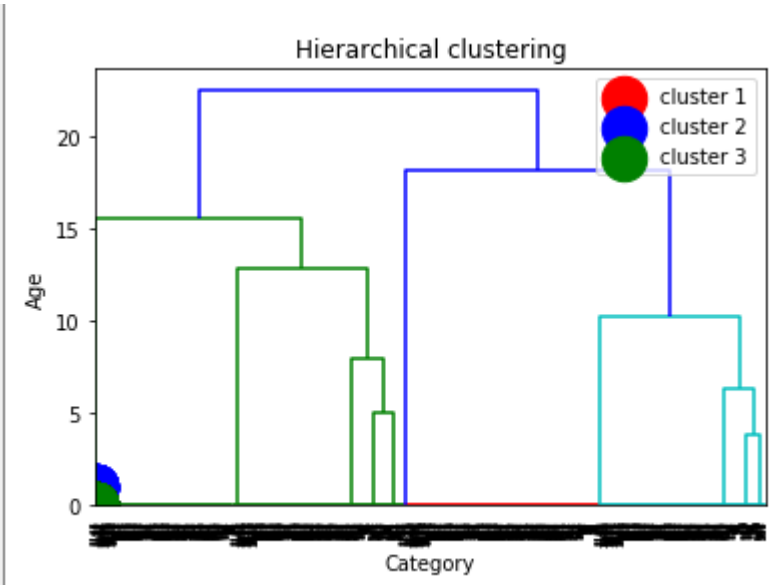


Hierarchical Clustering:

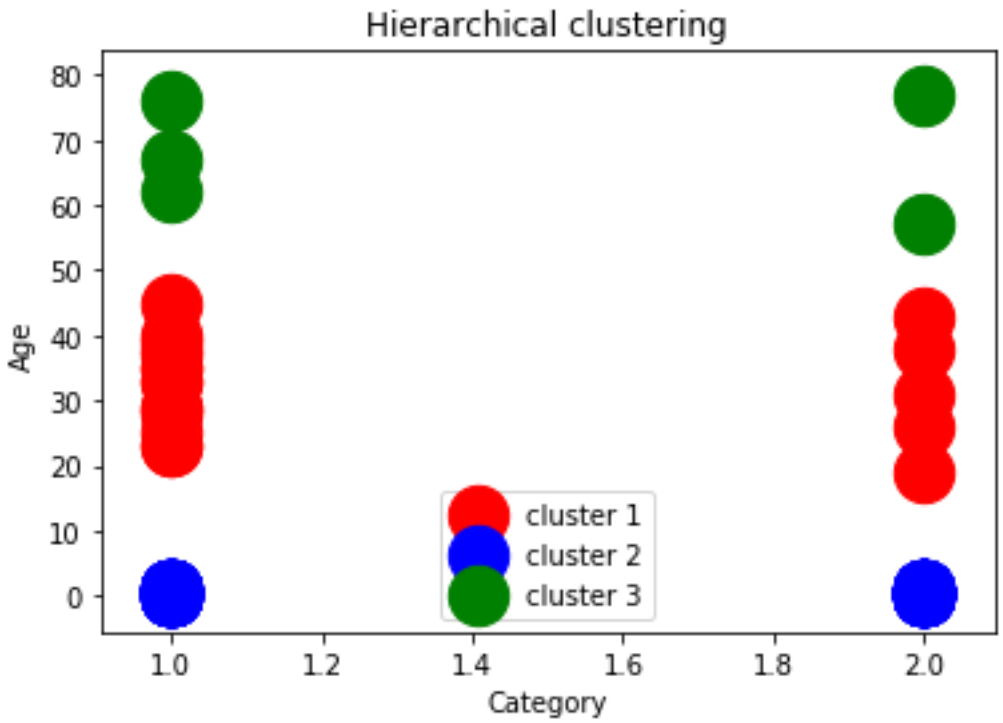
```
# -*- coding: utf-8 -*-
"""
Created on Sat Apr 21 14:24:56 2018

@author: neera
"""
#importing libraries
import numpy as np
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt
import pandas as pd
#Loading Data from CSV file
mydata=pd.read_csv('DM_PR8.csv')
X=mydata.iloc[:,[2,10,11,12]].values #  loading category and age columns
# finding best number of clusters
import scipy.cluster.hierarchy as sc
dendrogram=sc.dendrogram(sc.linkage(X,method='ward'))
# Run until finding best number of clusters then run below code
   # applying hierarchical clustering to our dataset
from sklearn.cluster import AgglomerativeClustering
Hierarchical=AgglomerativeClustering(n_clusters=2,affinity='euclidean',linkage='ward')
PredictH=Hierarchical.fit_predict(X)
```

 # visualization of our hierarchical clustering

```
plt.scatter(X[PredictH==0,0],X[PredictH==0,1],s=500,c='red',label='cluster 1')

plt.scatter(X[PredictH==1,0],X[PredictH==1,1],s=500,c='blue',label='cluster 2')

plt.scatter(X[PredictH==2,0],X[PredictH==2,1],s=500,c='green',label='cluster 3')

plt.title('Hierarchical clustering')

#plt.xlabel('Category')

#plt.ylabel('Age')

plt.legend()

plt.show()
```



plt.show()

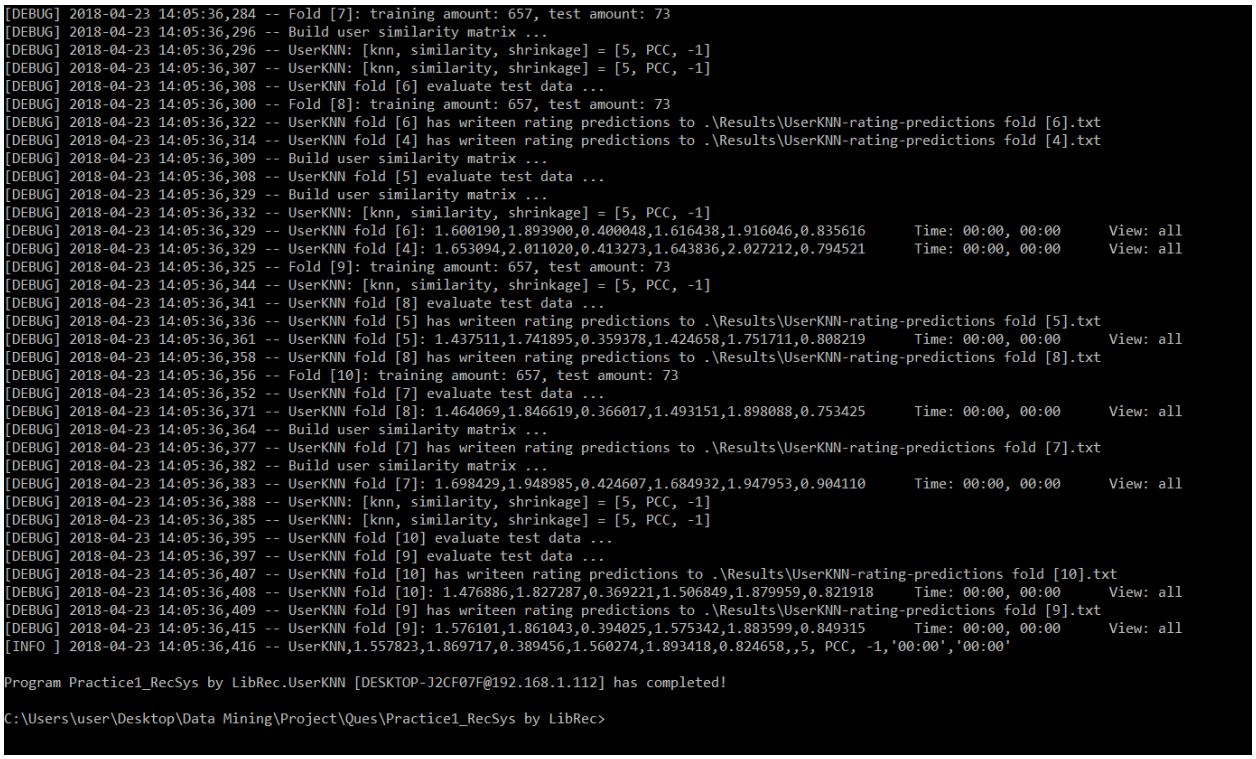


SOLUTION 9)

To recommend the cabs based on drivers rating. For this we will find which algorithm i.e. UserKNN or Biased Matrix Factorization algorithm is best to recommend cabs based on driver's rating.

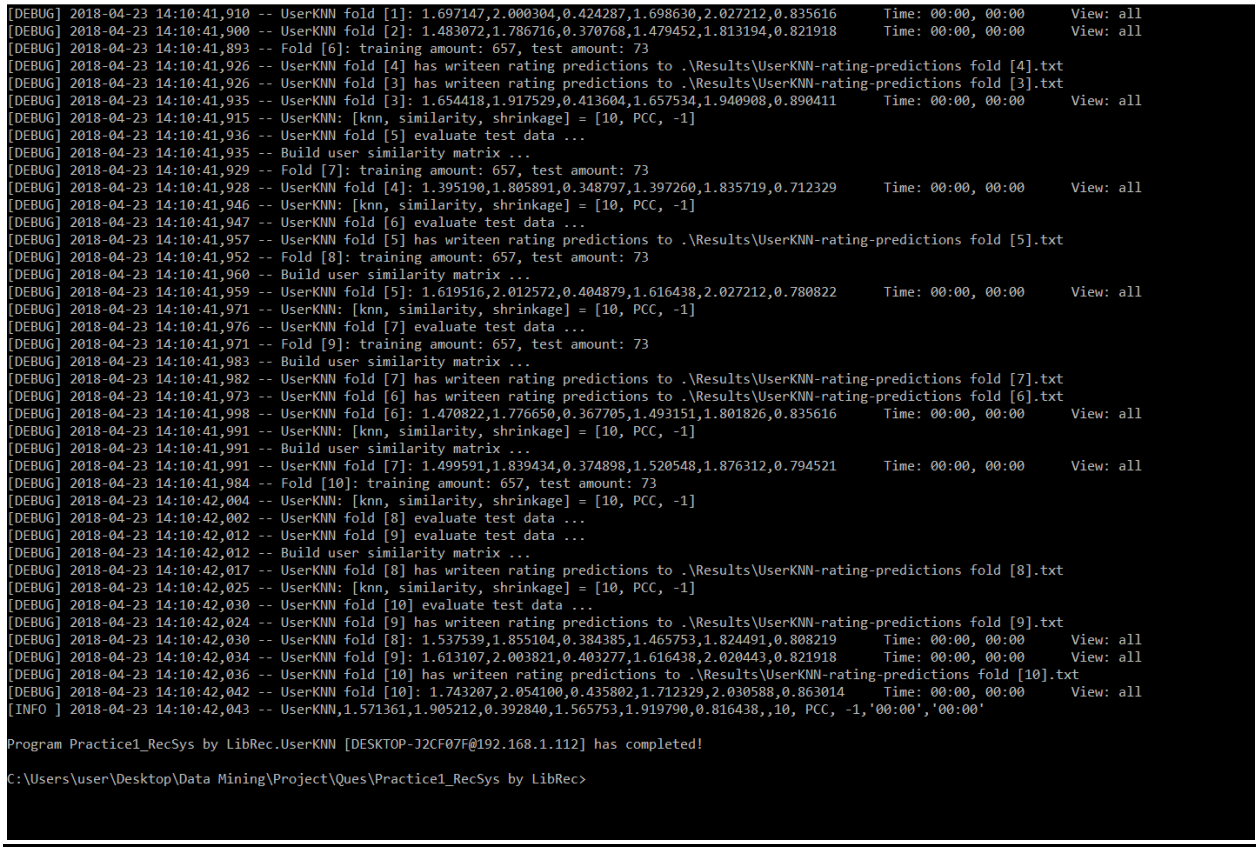
First, we will find MAE value for USERKNN algorithm:

- **<u>Using Similarity metrics: PCC at K=5 and 10 fold cross evaluation:</u>**



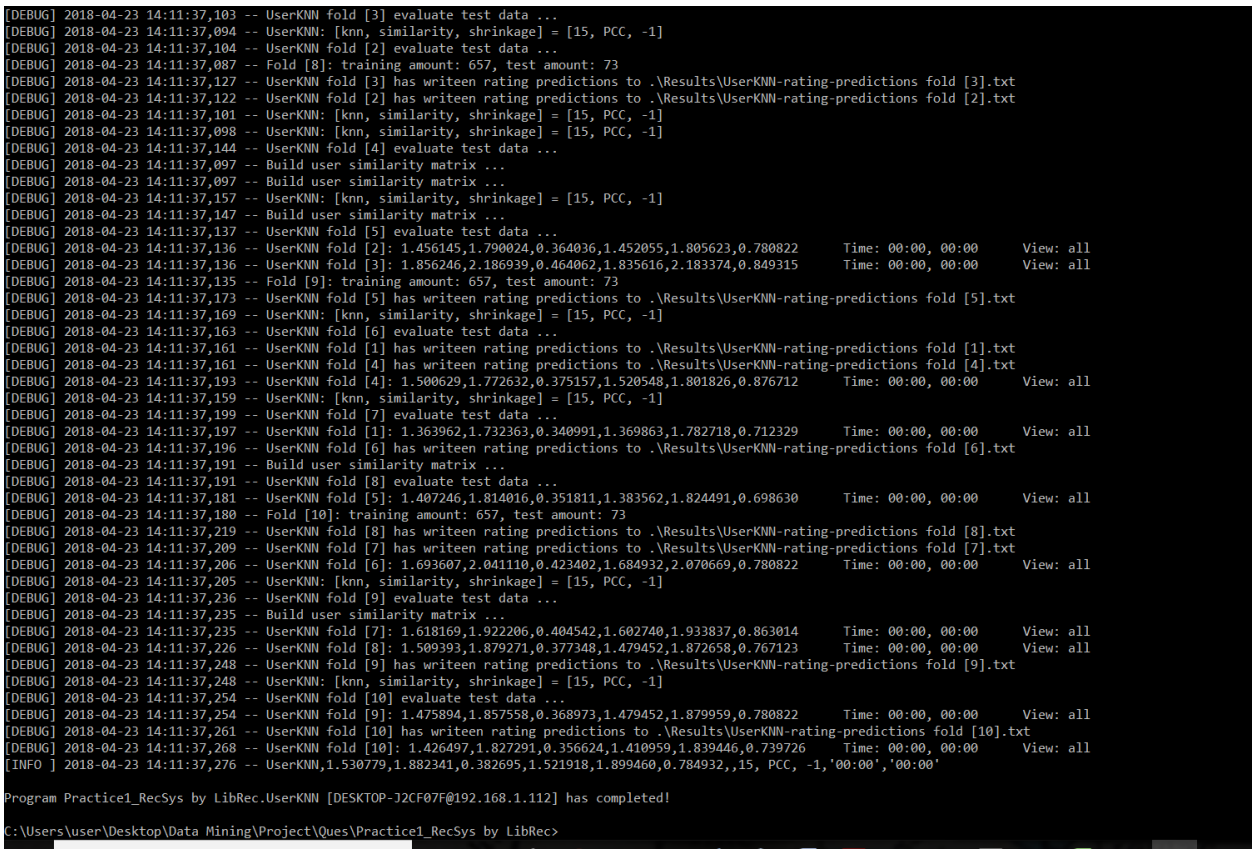
Here MAE value is **1.557823.**

- **<u>Using Similarity metrics: PCC at K=10 and 10 fold cross evaluation</u>**



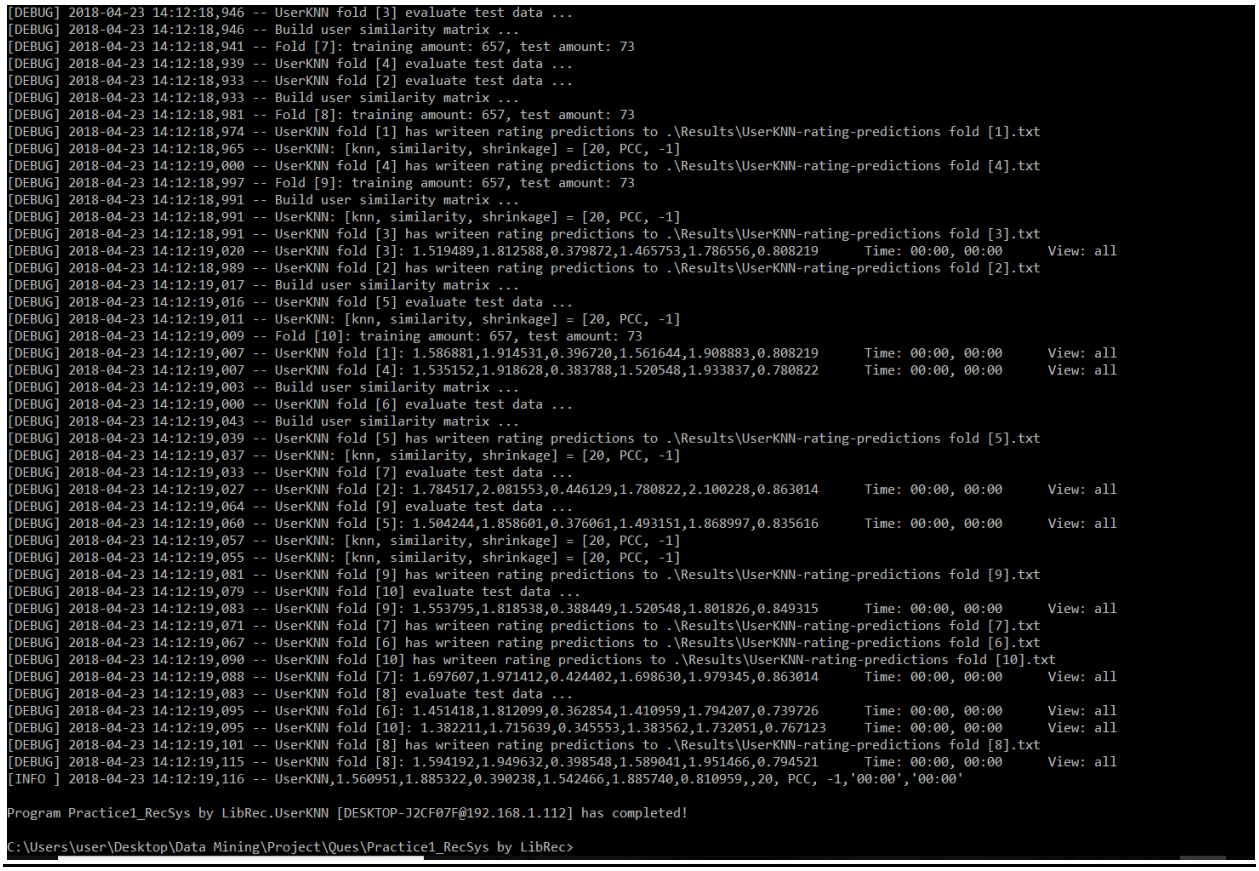
Here MAE value Is **1.571361**

- **<u>Using Similarity metrics: PCC at K=15 and 10 fold cross evaluation</u>**



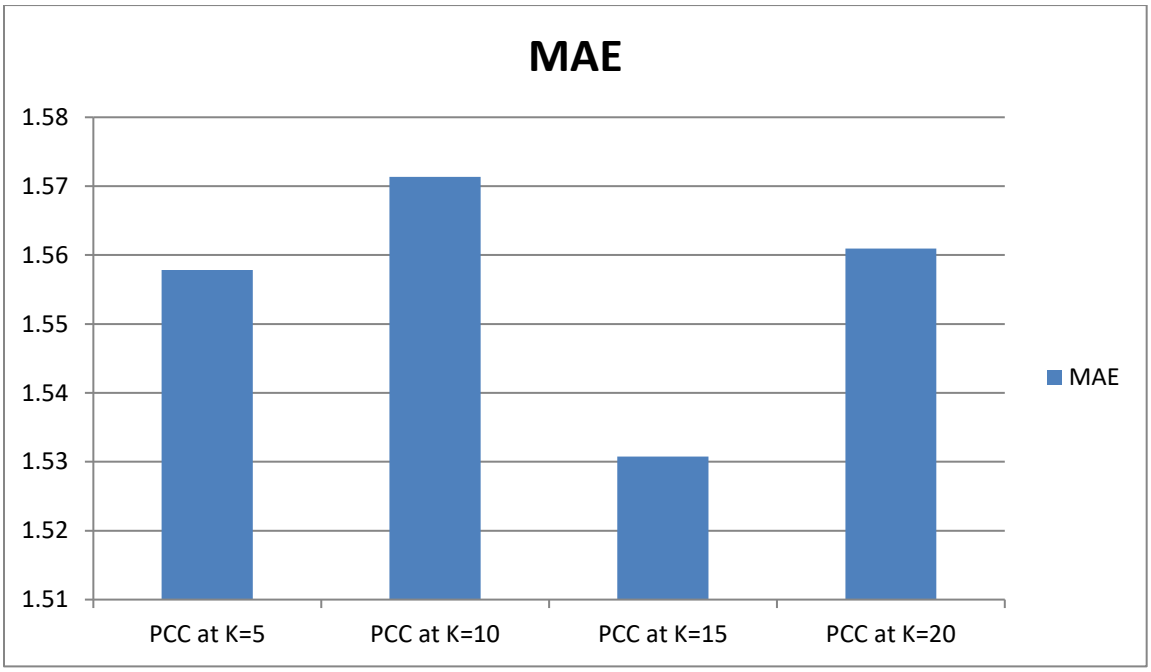
Here MAE value is **1.530779**

- **<u>Using Similarity metrics: PCC at K=20 and 10 fold cross evaluation</u>**



Here MAE value is **1.560951**

Now we will draw MAE BAR Graph for PCC at K=5,10,15,20

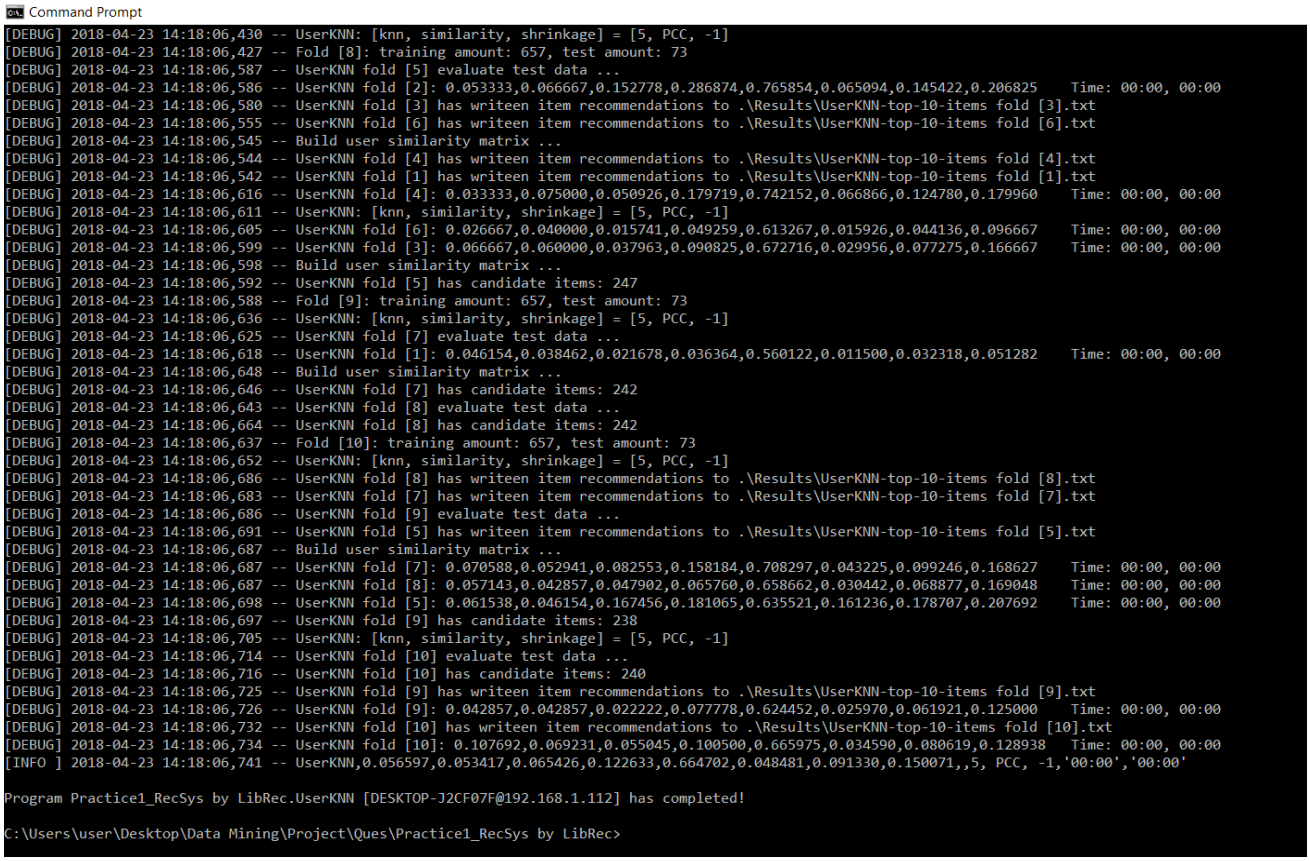


By comparing all the values we find that lowest MAE value is for PCC at K=15

Now we will find:

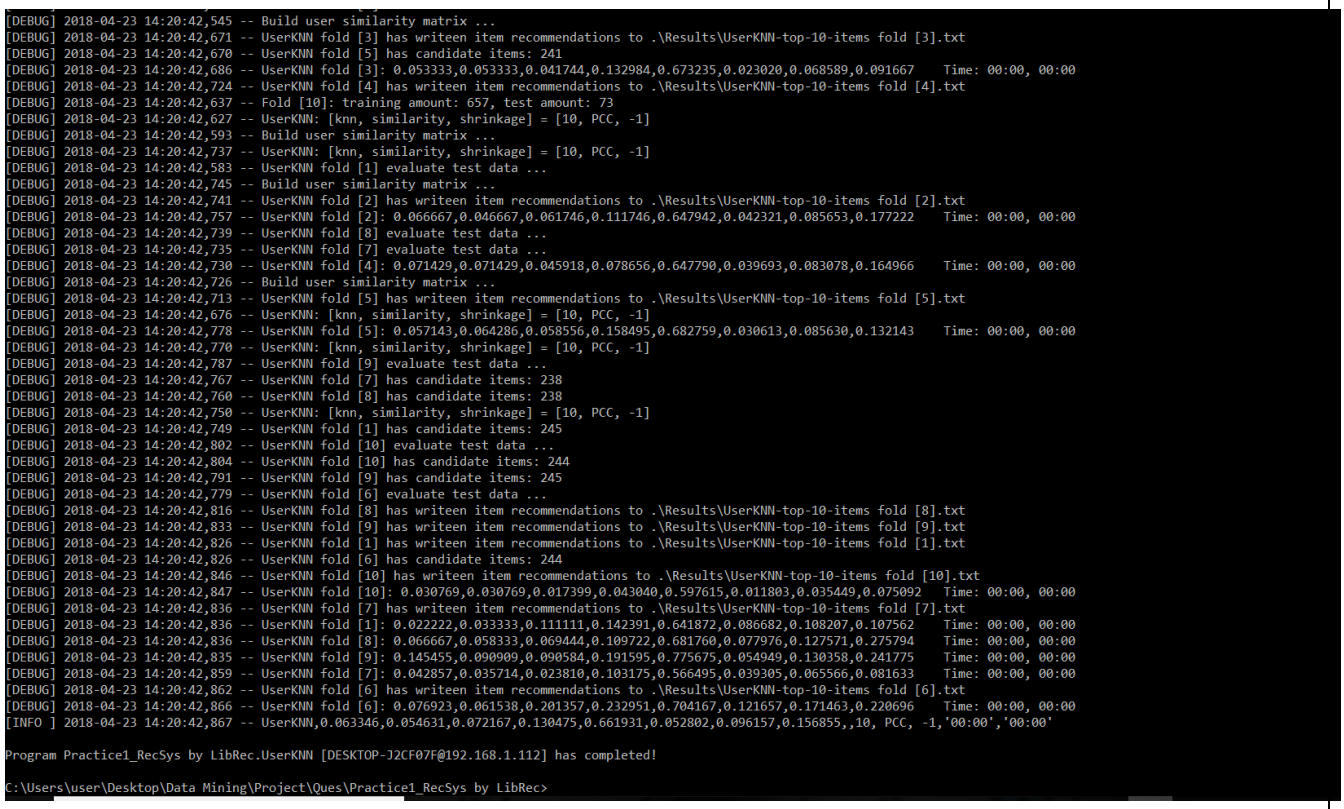
- **Precision@10 for PCC at K=5**

```
22
23  # main option: is ranking prediction
24  item.ranking=on -topN -1 -ignore -1
25
26  # main option: is writing out recommendation results; [--fold-data --save-model]
27  output.setup=on -dir ./Results/ -verbose on, off --to-clipboard --to-file filmtrust-cv.txt
28
29  # Guava cache configuration
30  guava.cache.spec=maximumSize=200,expireAfterAccess=2m
31
32  # main option: is email notification enabled
33  email.setup=off -host smtp.email.com -port 465 -user xxx@email.com -password yyyy -auth true -to xxx@email.com
34
35  ################################################ Model-based Methods ########################################
36  num.factors=5
37  num.max.iter=100
38
39  # options: -bold-driver, -decay ratio, -moment value
40  learn.rate=0.001 -max -1 -bold-driver
41
42  reg.lambda=0.1 -u 0.001 -i 0.001 -b 0.001 -s 0.001
43
44  # probabilistic graphic models
45  pgm.setup=-alpha 2 -beta 0.5 -burn-in 300 -sample-lag 10 -interval 100
46
47  ################################################ Memory-based Methods ########################################
48  # similarity method: PCC, COS, COS-Binary, MSD, CPC, exJaccard; -1 to disable shrinking;
49  similarity=PCC
50  num.shrinkage=-1
51
52  # neighborhood size; -1 to use as many as possible.
53  num.neighbors=5
54
```



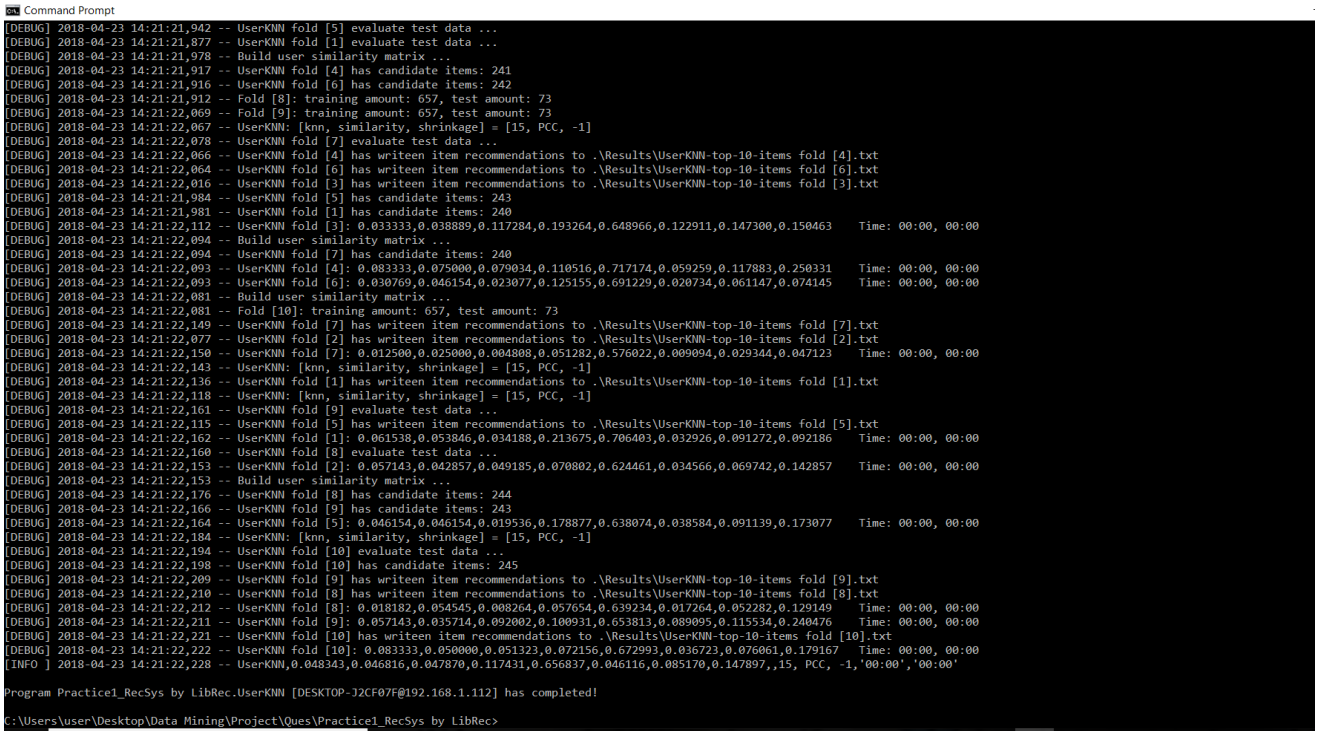
Here Precision@10 is **0.053417**

- **Precisio@10 for PCC at K=10**



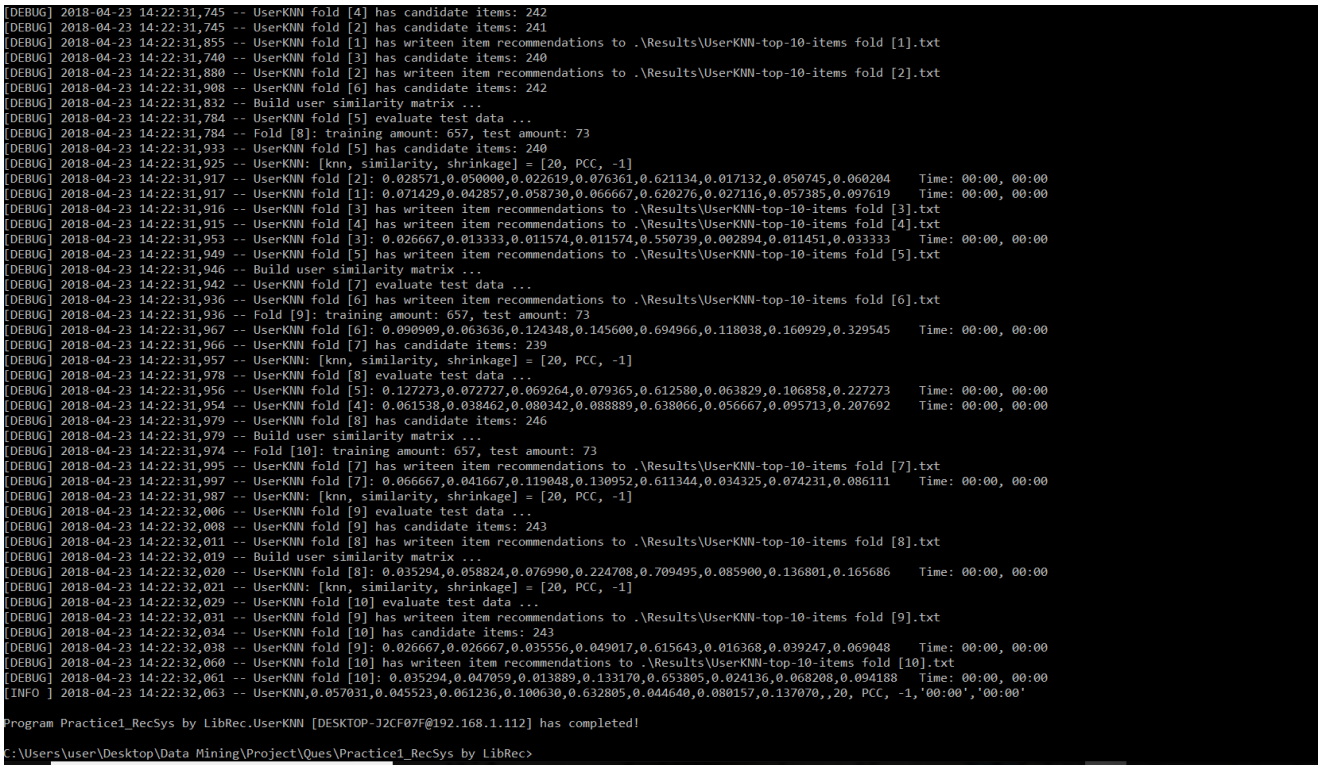
Here precision@10 is **0.054631**

- **Precisio@10 for PCC at K=15**



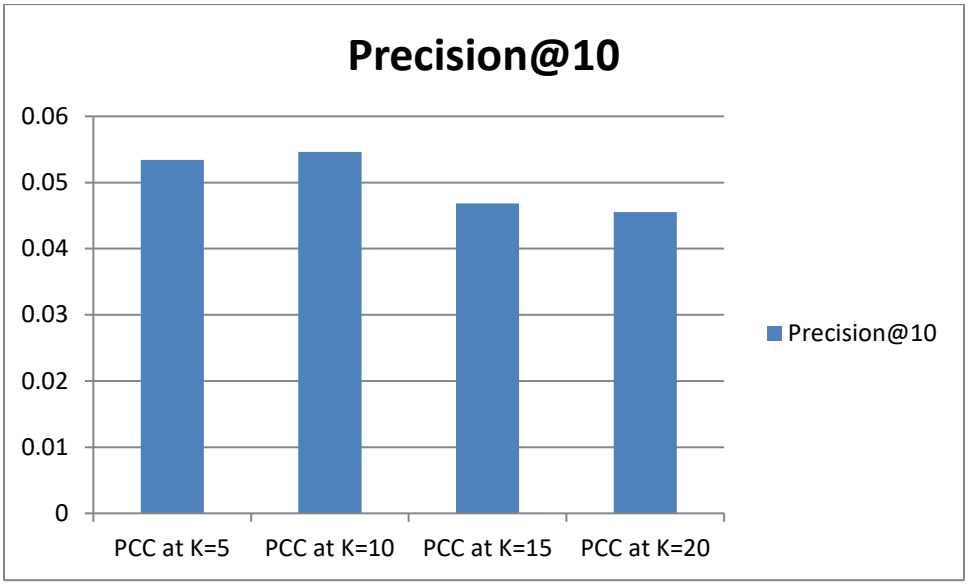
Here Precision@10 is **0.046816**

- **Precisio@10 for PCC at K=20**



Here precisin@10 is **0.045523**

We will now compare all 4 precision values using Bar graph:

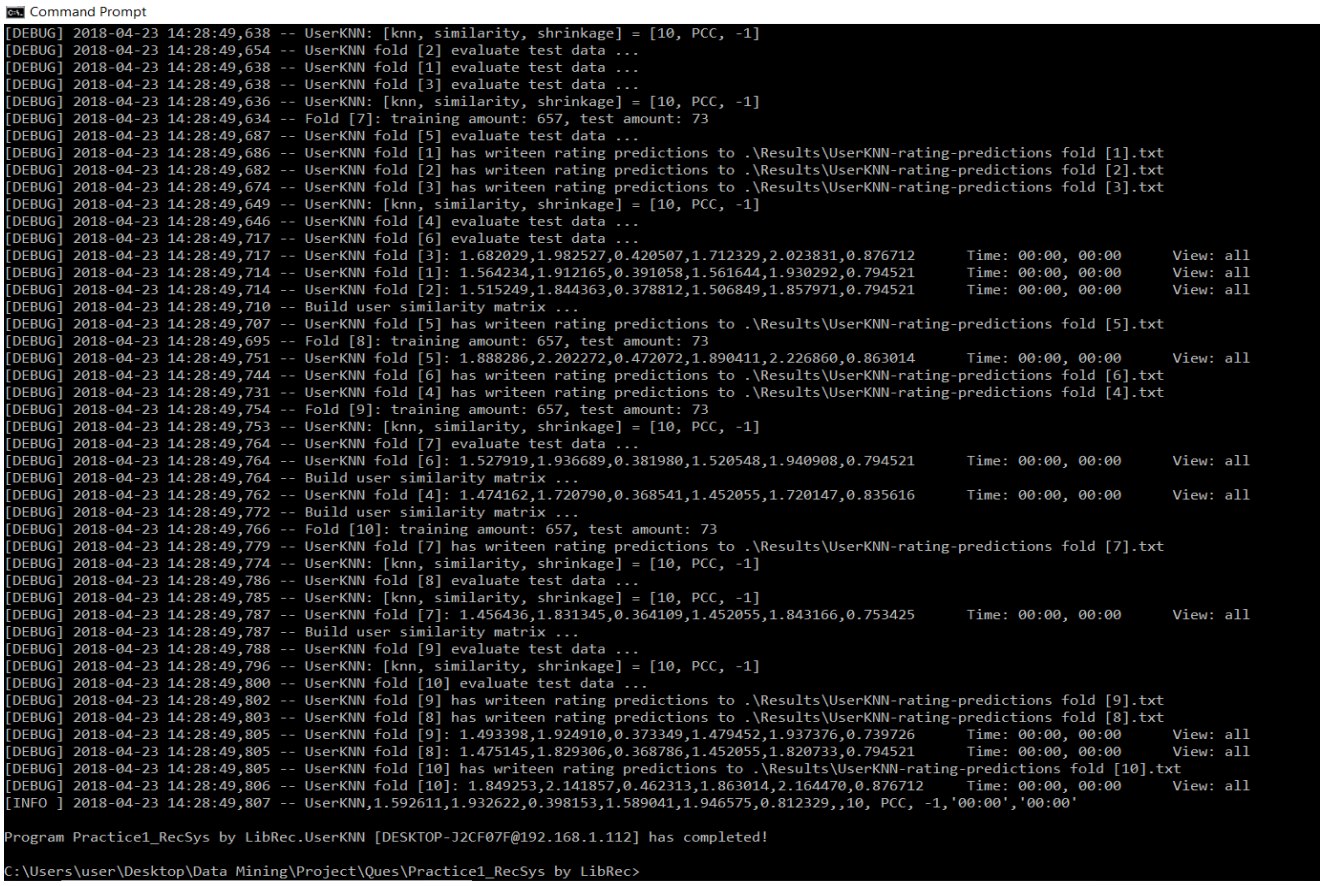


By comparing Precision@10 values for PCC at different factors we find that its highest for PCC at k=10, so it is better than PCC at rest other values.

b) <u>Now we will find MAE and Precision@10 for different similarity Matrix i.e. PCC, COS and COS-Binary for k=10</u>

MAE at PCC k=10

```
recommender=UserKNN

# main option: 1. test-set -f test-file-path; 2. cv (cross validation) -k k-folds [-p on, off]
# 3. leave-one-out -t threads -target u, i, r [--by-date]; 4. given-ratio -r ratio -target u, i, r [--by-date]; 5. gi
# optional: [-v validation-ratio] [--repeat n] [-cpu n] [--rand-seed n] [--test-view all, cold-start, trust-degree 1
evaluation.setup=cv -k 10 --test-view all --early-stop RMSE

# main option: is ranking prediction
item.ranking=off -topN -1 -ignore -1

# main option: is writing out recommendation results; [--fold-data --save-model]
output.setup=on -dir ./Results/ -verbose on, off --to-clipboard --to-file filmtrust-cv.txt

# Guava cache configuration
guava.cache.spec=maximumSize=200,expireAfterAccess=2m

# main option: is email notification enabled
email.setup=off -host smtp.email.com -port 465 -user xxx@email.com -password yyyy -auth true -to xxx@email.com

################################################## Model-based Methods ########################################
num.factors=5
num.max.iter=100

# options: -bold-driver, -decay ratio, -moment value
learn.rate=0.001 -max -1 -bold-driver

reg.lambda=0.1 -u 0.001 -i 0.001 -b 0.001 -s 0.001

# probabilistic graphic models
pgm.setup=-alpha 2 -beta 0.5 -burn-in 300 -sample-lag 10 -interval 100

################################################## Memory-based Methods ########################################
# similarity method: PCC, COS, COS-Binary, MSD, CPC, exJaccard; -1 to disable shrinking;
similarity=PCC
num.shrinkage=-1

# neighborhood size; -1 to use as many as possible.
num.neighbors=10
```



Here MAE value is **1.592611**

- **MAE for COS at k=10**

```
# 3. leave-one-out -t threads -target u, i, r [--by-date]; 4. given-ratio -r ratio -target u, i, r [--by-date]; 5. given-n -N n -target u, i [--
# optional: [-v validation-ratio] [--repeat n] [-cpu n] [--rand-seed n] [--test-view all, cold-start, trust-degree 1 5] [--early-stop loss, MAE,
evaluation.setup=cv -k 10 --test-view all --early-stop RMSE

# main option: is ranking prediction
item.ranking=off -topN -1 -ignore -1

# main option: is writing out recommendation results; [--fold-data --save-model]
output.setup=on -dir ./Results/ -verbose on, off --to-clipboard --to-file filmtrust-cv.txt

# Guava cache configuration
guava.cache.spec=maximumSize=200,expireAfterAccess=2m

# main option: is email notification enabled
email.setup=off -host smtp.email.com -port 465 -user xxx@email.com -password yyyy -auth true -to xxx@email.com

################################################## Model-based Methods ########################################
num.factors=5
num.max.iter=100

# options: -bold-driver, -decay ratio, -moment value
learn.rate=0.001 -max -1 -bold-driver

reg.lambda=0.1 -u 0.001 -i 0.001 -b 0.001 -s 0.001

# probabilistic graphic models
pgm.setup=-alpha 2 -beta 0.5 -burn-in 300 -sample-lag 10 -interval 100

################################################## Memory-based Methods ########################################
# similarity method: PCC, COS, COS-Binary, MSD, CPC, exJaccard; -1 to disable shrinking;
similarity=COS
num.shrinkage=-1

# neighborhood size; -1 to use as many as possible.
num.neighbors=10

################################################## Method-specific Settings ########################################
AoBPR=-lambda 0.3
```

```
[DEBUG] 2018-04-23 14:30:47,686 -- Build user similarity matrix ...
[DEBUG] 2018-04-23 14:30:47,681 -- Fold [7]: training amount: 657, test amount: 73
[DEBUG] 2018-04-23 14:30:47,679 -- Build user similarity matrix ...
[DEBUG] 2018-04-23 14:30:47,679 -- UserKNN: [knn, similarity, shrinkage] = [10, COS, -1]
[DEBUG] 2018-04-23 14:30:47,720 -- UserKNN fold [4] evaluate test data ...
[DEBUG] 2018-04-23 14:30:47,673 -- UserKNN fold [2] evaluate test data ...
[DEBUG] 2018-04-23 14:30:47,731 -- Build user similarity matrix ...
[DEBUG] 2018-04-23 14:30:47,725 -- UserKNN fold [3] has writeen rating predictions to .\Results\UserKNN-rating-predictions fold [3].txt
[DEBUG] 2018-04-23 14:30:47,723 -- Fold [8]: training amount: 657, test amount: 73
[DEBUG] 2018-04-23 14:30:47,722 -- UserKNN: [knn, similarity, shrinkage] = [10, COS, -1]
[DEBUG] 2018-04-23 14:30:47,709 -- UserKNN fold [1] has writeen rating predictions to .\Results\UserKNN-rating-predictions fold [1].txt
[DEBUG] 2018-04-23 14:30:47,706 -- UserKNN: [knn, similarity, shrinkage] = [10, COS, -1]
[DEBUG] 2018-04-23 14:30:47,766 -- UserKNN fold [6] evaluate test data ...
[DEBUG] 2018-04-23 14:30:47,758 -- Fold [9]: training amount: 657, test amount: 73
[DEBUG] 2018-04-23 14:30:47,758 -- UserKNN fold [1]: 1.410237,1.670711,0.352559,1.424658,1.704145,0.821918      Time: 00:00, 00:00      View: all
[DEBUG] 2018-04-23 14:30:47,757 -- UserKNN fold [5] evaluate test data ...
[DEBUG] 2018-04-23 14:30:47,755 -- UserKNN fold [3]: 1.302078,1.583496,0.325520,1.315068,1.604787,0.767123      Time: 00:00, 00:00      View: all
[DEBUG] 2018-04-23 14:30:47,746 -- UserKNN: [knn, similarity, shrinkage] = [10, COS, -1]
[DEBUG] 2018-04-23 14:30:47,781 -- UserKNN fold [7] evaluate test data ...
[DEBUG] 2018-04-23 14:30:47,779 -- Build user similarity matrix ...
[DEBUG] 2018-04-23 14:30:47,778 -- Fold [10]: training amount: 657, test amount: 73
[DEBUG] 2018-04-23 14:30:47,773 -- UserKNN fold [4] has writeen rating predictions to .\Results\UserKNN-rating-predictions fold [4].txt
[DEBUG] 2018-04-23 14:30:47,798 -- UserKNN fold [7] has writeen rating predictions to .\Results\UserKNN-rating-predictions fold [7].txt
[DEBUG] 2018-04-23 14:30:47,795 -- UserKNN fold [5] has writeen rating predictions to .\Results\UserKNN-rating-predictions fold [5].txt
[DEBUG] 2018-04-23 14:30:47,793 -- UserKNN: [knn, similarity, shrinkage] = [10, COS, -1]
[DEBUG] 2018-04-23 14:30:47,808 -- UserKNN fold [8] evaluate test data ...
[DEBUG] 2018-04-23 14:30:47,787 -- UserKNN fold [2] has writeen rating predictions to .\Results\UserKNN-rating-predictions fold [2].txt
[DEBUG] 2018-04-23 14:30:47,786 -- Build user similarity matrix ...
[DEBUG] 2018-04-23 14:30:47,785 -- UserKNN fold [6] has writeen rating predictions to .\Results\UserKNN-rating-predictions fold [6].txt
[DEBUG] 2018-04-23 14:30:47,817 -- UserKNN: [knn, similarity, shrinkage] = [10, COS, -1]
[DEBUG] 2018-04-23 14:30:47,819 -- UserKNN fold [9] evaluate test data ...
[DEBUG] 2018-04-23 14:30:47,815 -- UserKNN fold [2]: 1.529699,1.782794,0.382425,1.506849,1.782718,0.876712      Time: 00:00, 00:00      View: all
[DEBUG] 2018-04-23 14:30:47,814 -- UserKNN fold [8] has writeen rating predictions to .\Results\UserKNN-rating-predictions fold [8].txt
[DEBUG] 2018-04-23 14:30:47,808 -- UserKNN fold [5]: 1.510973,1.733962,0.377743,1.506849,1.759514,0.863014      Time: 00:00, 00:00      View: all
[DEBUG] 2018-04-23 14:30:47,802 -- UserKNN fold [7]: 1.336418,1.637472,0.334104,1.369863,1.687991,0.794521      Time: 00:00, 00:00      View: all
[DEBUG] 2018-04-23 14:30:47,801 -- Build user similarity matrix ...
[DEBUG] 2018-04-23 14:30:47,800 -- UserKNN fold [4]: 1.381013,1.660687,0.345253,1.369863,1.679856,0.794521      Time: 00:00, 00:00      View: all
[DEBUG] 2018-04-23 14:30:47,826 -- UserKNN: [knn, similarity, shrinkage] = [10, COS, -1]
[DEBUG] 2018-04-23 14:30:47,823 -- UserKNN fold [9] has writeen rating predictions to .\Results\UserKNN-rating-predictions fold [9].txt
[DEBUG] 2018-04-23 14:30:47,823 -- UserKNN fold [8]: 1.371568,1.689780,0.342892,1.397260,1.767283,0.753425      Time: 00:00, 00:00      View: all
[DEBUG] 2018-04-23 14:30:47,819 -- UserKNN fold [6]: 1.653466,1.925271,0.413367,1.712329,1.996572,0.876712      Time: 00:00, 00:00      View: all
[DEBUG] 2018-04-23 14:30:47,835 -- UserKNN fold [9]: 1.308246,1.638399,0.327061,1.301370,1.667580,0.726027      Time: 00:00, 00:00      View: all
[DEBUG] 2018-04-23 14:30:47,834 -- UserKNN fold [10] evaluate test data ...
[DEBUG] 2018-04-23 14:30:47,847 -- UserKNN fold [10] has writeen rating predictions to .\Results\UserKNN-rating-predictions fold [10].txt
[DEBUG] 2018-04-23 14:30:47,850 -- UserKNN fold [10]: 1.532901,1.751599,0.383225,1.520548,1.771154,0.849315      Time: 00:00, 00:00      View: all
[INFO ] 2018-04-23 14:30:47,852 -- UserKNN,1.433660,1.707417,0.358415,1.442466,1.742160,0.812329,,10, COS, -1,'00:00','00:00'

Program Practice1_RecSys by LibRec.UserKNN [DESKTOP-J2CF07F@192.168.1.112] has completed!

C:\Users\user\Desktop\Data Mining\Project\Ques\Practice1_RecSys by LibRec>
```

Here MAE value is 1.433660

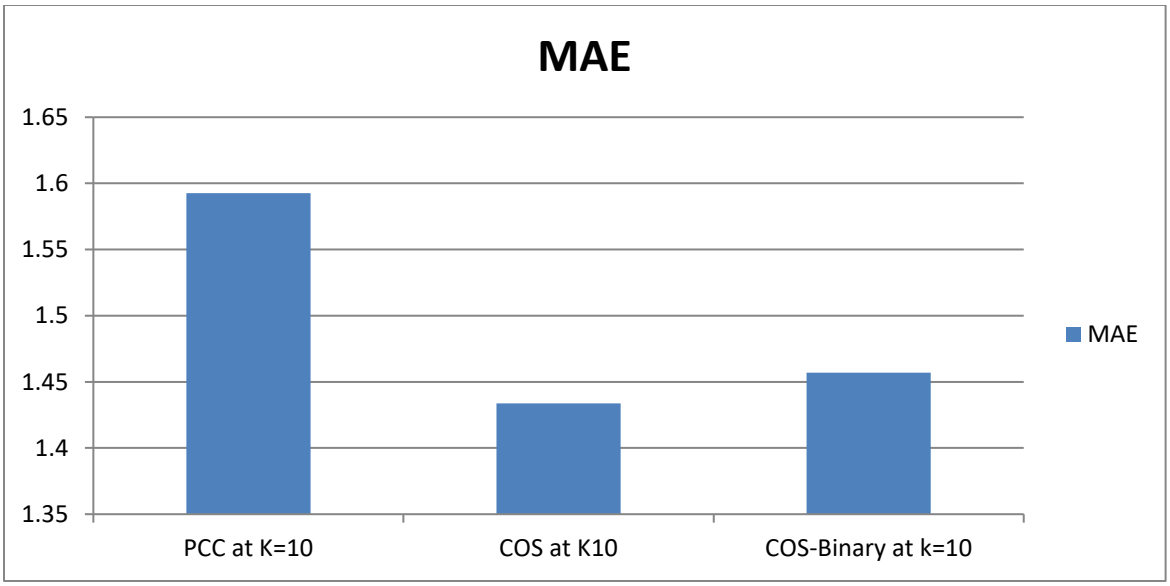
- **MAE for COS-Binary at k=10**

```
# 3. leave-one-out -t threads -target u, i, r [--by-date]; 4. given-ratio -r ratio -target u, i, r [--by-date]; 5. given-n -N n -target u
# optional: [-v validation-ratio] [--repeat n] [-cpu n] [--rand-seed n] [--test-view all, cold-start, trust-degree 1 5] [--early-stop los
evaluation.setup=cv -k 10 --test-view all --early-stop RMSE

# main option: is ranking prediction
item.ranking=off -topN -1 -ignore -1

# main option: is writing out recommendation results; [--fold-data --save-model]
output.setup=on -dir ./Results/ -verbose on, off --to-clipboard --to-file filmtrust-cv.txt

# Guava cache configuration
guava.cache.spec=maximumSize=200,expireAfterAccess=2m

# main option: is email notification enabled
email.setup=off -host smtp.email.com -port 465 -user xxx@email.com -password yyyy -auth true -to xxx@email.com

################################################## Model-based Methods ########################################
num.factors=5
num.max.iter=100

# options: -bold-driver, -decay ratio, -moment value
learn.rate=0.001 -max -1 -bold-driver

reg.lambda=0.1 -u 0.001 -i 0.001 -b 0.001 -s 0.001

# probabilistic graphic models
pgm.setup=-alpha 2 -beta 0.5 -burn-in 300 -sample-lag 10 -interval 100

################################################## Memory-based Methods ########################################
# similarity method: PCC, COS, COS-Binary, MSD, CPC, exJaccard; -1 to disable shrinking;
similarity=COS-Binary
num.shrinkage=-1

# neighborhood size; -1 to use as many as possible.
num.neighbors=10

################################################## Method-specific Settings ####################################
AoBPR=-lambda 0.3
```

```
[DEBUG] 2018-04-23 14:31:55,201 -- Build user similarity matrix ...
[DEBUG] 2018-04-23 14:31:55,201 -- Build user similarity matrix ...
[DEBUG] 2018-04-23 14:31:55,200 -- Fold [8]: training amount: 657, test amount: 73
[DEBUG] 2018-04-23 14:31:55,197 -- UserKNN: [knn, similarity, shrinkage] = [10, COS-Binary, -1]
[DEBUG] 2018-04-23 14:31:55,223 -- UserKNN fold [4] evaluate test data ...
[DEBUG] 2018-04-23 14:31:55,197 -- UserKNN: [knn, similarity, shrinkage] = [10, COS-Binary, -1]
[DEBUG] 2018-04-23 14:31:55,231 -- UserKNN fold [5] evaluate test data ...
[DEBUG] 2018-04-23 14:31:55,193 -- UserKNN: [knn, similarity, shrinkage] = [10, COS-Binary, -1]
[DEBUG] 2018-04-23 14:31:55,238 -- Build user similarity matrix ...
[DEBUG] 2018-04-23 14:31:55,238 -- UserKNN fold [1] has writeen rating predictions to .\Results\UserKNN-rating-predictions fold [1].txt
[DEBUG] 2018-04-23 14:31:55,237 -- UserKNN fold [4] has writeen rating predictions to .\Results\UserKNN-rating-predictions fold [4].txt
[DEBUG] 2018-04-23 14:31:55,226 -- UserKNN: [knn, similarity, shrinkage] = [10, COS-Binary, -1]
[DEBUG] 2018-04-23 14:31:55,258 -- UserKNN fold [6] evaluate test data ...
[DEBUG] 2018-04-23 14:31:55,223 -- Fold [9]: training amount: 657, test amount: 73
[DEBUG] 2018-04-23 14:31:55,216 -- UserKNN: [knn, similarity, shrinkage] = [10, COS-Binary, -1]
[DEBUG] 2018-04-23 14:31:55,212 -- UserKNN fold [3] evaluate test data ...
[DEBUG] 2018-04-23 14:31:55,279 -- Build user similarity matrix ...
[DEBUG] 2018-04-23 14:31:55,287 -- UserKNN fold [3] has writeen rating predictions to .\Results\UserKNN-rating-predictions fold [3].txt
[DEBUG] 2018-04-23 14:31:55,289 -- UserKNN fold [3]: 1.324597,1.597577,0.331149,1.315068,1.630195,0.794521      Time: 00:00, 00:00      View: all
[DEBUG] 2018-04-23 14:31:55,291 -- UserKNN: [knn, similarity, shrinkage] = [10, COS-Binary, -1]
[DEBUG] 2018-04-23 14:31:55,295 -- UserKNN fold [9] evaluate test data ...
[DEBUG] 2018-04-23 14:31:55,277 -- UserKNN fold [7] evaluate test data ...
[DEBUG] 2018-04-23 14:31:55,268 -- Fold [10]: training amount: 657, test amount: 73
[DEBUG] 2018-04-23 14:31:55,311 -- UserKNN fold [7] has writeen rating predictions to .\Results\UserKNN-rating-predictions fold [7].txt
[DEBUG] 2018-04-23 14:31:55,262 -- UserKNN fold [1]: 1.437733,1.719228,0.359433,1.438356,1.739942,0.808219      Time: 00:00, 00:00      View: all
[DEBUG] 2018-04-23 14:31:55,262 -- UserKNN fold [4]: 1.515194,1.800844,0.378798,1.534247,1.850583,0.821918      Time: 00:00, 00:00      View: all
[DEBUG] 2018-04-23 14:31:55,258 -- UserKNN fold [5] has writeen rating predictions to .\Results\UserKNN-rating-predictions fold [5].txt
[DEBUG] 2018-04-23 14:31:55,249 -- UserKNN: [knn, similarity, shrinkage] = [10, COS-Binary, -1]
[DEBUG] 2018-04-23 14:31:55,240 -- UserKNN fold [2] evaluate test data ...
[DEBUG] 2018-04-23 14:31:55,329 -- UserKNN fold [8] evaluate test data ...
[DEBUG] 2018-04-23 14:31:55,327 -- UserKNN fold [5]: 1.319800,1.620349,0.329950,1.315068,1.638576,0.753425      Time: 00:00, 00:00      View: all
[DEBUG] 2018-04-23 14:31:55,317 -- Build user similarity matrix ...
[DEBUG] 2018-04-23 14:31:55,317 -- UserKNN fold [7]: 1.499766,1.809660,0.374941,1.479452,1.857971,0.794521      Time: 00:00, 00:00      View: all
[DEBUG] 2018-04-23 14:31:55,304 -- UserKNN fold [9] has writeen rating predictions to .\Results\UserKNN-rating-predictions fold [9].txt
[DEBUG] 2018-04-23 14:31:55,280 -- UserKNN fold [6] has writeen rating predictions to .\Results\UserKNN-rating-predictions fold [6].txt
[DEBUG] 2018-04-23 14:31:55,348 -- UserKNN fold [9]: 1.378705,1.637641,0.344676,1.356164,1.634391,0.835616      Time: 00:00, 00:00      View: all
[DEBUG] 2018-04-23 14:31:55,344 -- UserKNN: [knn, similarity, shrinkage] = [10, COS-Binary, -1]
[DEBUG] 2018-04-23 14:31:55,340 -- UserKNN fold [8] has writeen rating predictions to .\Results\UserKNN-rating-predictions fold [8].txt
[DEBUG] 2018-04-23 14:31:55,333 -- UserKNN fold [2] has writeen rating predictions to .\Results\UserKNN-rating-predictions fold [2].txt
[DEBUG] 2018-04-23 14:31:55,359 -- UserKNN fold [8]: 1.354269,1.653902,0.338567,1.315068,1.655212,0.767123      Time: 00:00, 00:00      View: all
[DEBUG] 2018-04-23 14:31:55,360 -- UserKNN fold [2]: 1.640336,1.885415,0.410084,1.616438,1.894476,0.863014      Time: 00:00, 00:00      View: all
[DEBUG] 2018-04-23 14:31:55,349 -- UserKNN fold [10] evaluate test data ...
[DEBUG] 2018-04-23 14:31:55,348 -- UserKNN fold [6]: 1.501238,1.772756,0.375310,1.534247,1.813194,0.849315      Time: 00:00, 00:00      View: all
[DEBUG] 2018-04-23 14:31:55,364 -- UserKNN fold [10] has writeen rating predictions to .\Results\UserKNN-rating-predictions fold [10].txt
[DEBUG] 2018-04-23 14:31:55,364 -- UserKNN fold [10]: 1.598200,1.915212,0.399550,1.643836,1.993139,0.808219      Time: 00:00, 00:00      View: all
[INFO ] 2018-04-23 14:31:55,366 -- UserKNN,1.456984,1.741258,0.364246,1.454795,1.770768,0.809589,,10, COS-Binary, -1,'00:00','00:00'

Program Practice1_RecSys by LibRec.UserKNN [DESKTOP-J2CF07F@192.168.1.112] has completed!

C:\Users\user\Desktop\Data Mining\Project\Ques\Practice1_RecSys by LibRec>
```

Here MAE value is 1.456984

Now we will compare MAE value of all 3 similarities:



From above graph we see that Cos at k=10 has the lowest value as compared to others.

Precision@10 for PCC, COS and COS-Binary at k=10

- **Precision@10 for PCC at k=10**

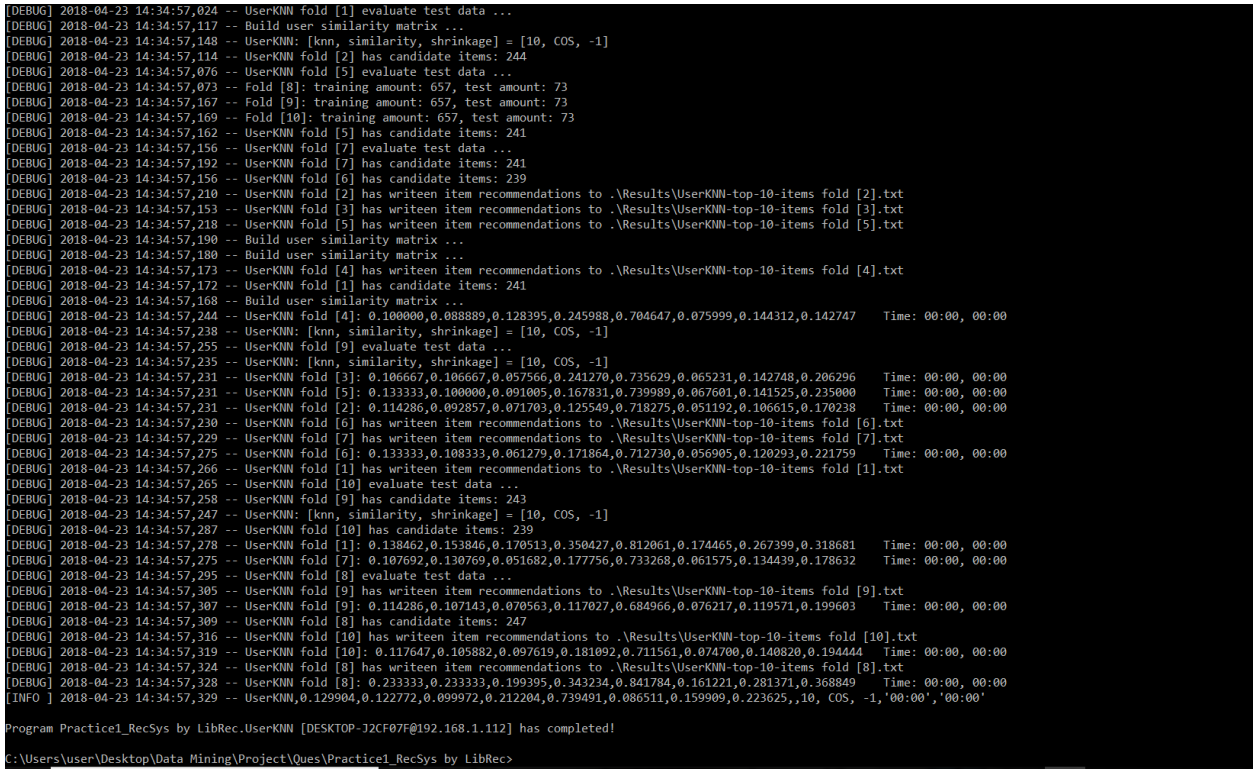
```
# 3. leave-one-out -t threads -target u, i, r [--by-date]; 4. given-ratio -r ratio -target u, i, r [--by-date]; 5. given-n -N n
# optional: [-v validation-ratio] [--repeat n] [-cpu n] [--rand-seed n] [--test-view all, cold-start, trust-degree 1 5] [--early
evaluation.setup=cv -k 10 --test-view all --early-stop RMSE

# main option: is ranking prediction
item.ranking=on -topN -1 -ignore -1

# main option: is writing out recommendation results; [--fold-data --save-model]
output.setup=on -dir ./Results/ -verbose on, off --to-clipboard --to-file filmtrust-cv.txt

# Guava cache configuration
guava.cache.spec=maximumSize=200,expireAfterAccess=2m

# main option: is email notification enabled
email.setup=off -host smtp.email.com -port 465 -user xxx@email.com -password yyyy -auth true -to xxx@email.com

################################################ Model-based Methods ########################################
num.factors=5
num.max.iter=100

# options: -bold-driver, -decay ratio, -moment value
learn.rate=0.001 -max -1 -bold-driver

reg.lambda=0.1 -u 0.001 -i 0.001 -b 0.001 -s 0.001

# probabilistic graphic models
pgm.setup=-alpha 2 -beta 0.5 -burn-in 300 -sample-lag 10 -interval 100

################################################ Memory-based Methods ########################################
# similarity method: PCC, COS, COS-Binary, MSD, CPC, exJaccard; -1 to disable shrinking;
similarity=PCC
num.shrinkage=-1

# neighborhood size; -1 to use as many as possible.
num.neighbors=10

################################################ Method-specific Settings ####################################
AoBPR=-lambda 0.3
```

```
[DEBUG] 2018-04-23 14:34:12,661 -- UserKNN fold [6] evaluate test data ...
[DEBUG] 2018-04-23 14:34:12,580 -- Fold [8]: training amount: 657, test amount: 73
[DEBUG] 2018-04-23 14:34:12,579 -- UserKNN: [knn, similarity, shrinkage] = [10, PCC, -1]
[DEBUG] 2018-04-23 14:34:12,675 -- UserKNN fold [5] evaluate test data ...
[DEBUG] 2018-04-23 14:34:12,666 -- UserKNN fold [3] has candidate items: 244
[DEBUG] 2018-04-23 14:34:12,665 -- UserKNN fold [6] has candidate items: 244
[DEBUG] 2018-04-23 14:34:12,663 -- UserKNN: [knn, similarity, shrinkage] = [10, PCC, -1]
[DEBUG] 2018-04-23 14:34:12,740 -- UserKNN fold [7] evaluate test data ...
[DEBUG] 2018-04-23 14:34:12,740 -- UserKNN fold [3] has writeen item recommendations to .\Results\UserKNN-top-10-items fold [3].txt
[DEBUG] 2018-04-23 14:34:12,737 -- UserKNN fold [5] has candidate items: 242
[DEBUG] 2018-04-23 14:34:12,703 -- UserKNN fold [2] has writeen item recommendations to .\Results\UserKNN-top-10-items fold [2].txt
[DEBUG] 2018-04-23 14:34:12,702 -- Build user similarity matrix ...
[DEBUG] 2018-04-23 14:34:12,693 -- UserKNN fold [4] has candidate items: 242
[DEBUG] 2018-04-23 14:34:12,692 -- UserKNN fold [1] has writeen item recommendations to .\Results\UserKNN-top-10-items fold [1].txt
[DEBUG] 2018-04-23 14:34:12,755 -- UserKNN fold [1]: 0.090909,0.072727,0.130165,0.166704,0.695277,0.082095,0.139977,0.255411    Time: 00:00, 00:00
[DEBUG] 2018-04-23 14:34:12,677 -- Fold [9]: training amount: 657, test amount: 73
[DEBUG] 2018-04-23 14:34:12,759 -- UserKNN: [knn, similarity, shrinkage] = [10, PCC, -1]
[DEBUG] 2018-04-23 14:34:12,764 -- UserKNN fold [8] evaluate test data ...
[DEBUG] 2018-04-23 14:34:12,751 -- UserKNN fold [2]: 0.040000,0.020000,0.015079,0.015079,0.552660,0.012222,0.029567,0.133333    Time: 00:00, 00:00
[DEBUG] 2018-04-23 14:34:12,747 -- UserKNN fold [3]: 0.109091,0.081818,0.073737,0.184848,0.710195,0.069565,0.138437,0.238636    Time: 00:00, 00:00
[DEBUG] 2018-04-23 14:34:12,745 -- UserKNN fold [7] has candidate items: 245
[DEBUG] 2018-04-23 14:34:12,743 -- UserKNN fold [6] has writeen item recommendations to .\Results\UserKNN-top-10-items fold [6].txt
[DEBUG] 2018-04-23 14:34:12,777 -- UserKNN fold [8] has candidate items: 242
[DEBUG] 2018-04-23 14:34:12,775 -- Build user similarity matrix ...
[DEBUG] 2018-04-23 14:34:12,788 -- UserKNN: [knn, similarity, shrinkage] = [10, PCC, -1]
[DEBUG] 2018-04-23 14:34:12,789 -- UserKNN fold [9] evaluate test data ...
[DEBUG] 2018-04-23 14:34:12,795 -- UserKNN fold [9] has candidate items: 243
[DEBUG] 2018-04-23 14:34:12,775 -- UserKNN fold [5] has writeen item recommendations to .\Results\UserKNN-top-10-items fold [5].txt
[DEBUG] 2018-04-23 14:34:12,799 -- UserKNN fold [5]: 0.022222,0.027778,0.012500,0.033929,0.567838,0.007981,0.026052,0.043981    Time: 00:00, 00:00
[DEBUG] 2018-04-23 14:34:12,801 -- UserKNN fold [7] has writeen item recommendations to .\Results\UserKNN-top-10-items fold [7].txt
[DEBUG] 2018-04-23 14:34:12,773 -- UserKNN fold [4] has writeen item recommendations to .\Results\UserKNN-top-10-items fold [4].txt
[DEBUG] 2018-04-23 14:34:12,766 -- Fold [10]: training amount: 657, test amount: 73
[DEBUG] 2018-04-23 14:34:12,811 -- UserKNN fold [4]: 0.042857,0.035714,0.060065,0.143398,0.663317,0.027552,0.070449,0.092262    Time: 00:00, 00:00
[DEBUG] 2018-04-23 14:34:12,809 -- UserKNN fold [7]: 0.076923,0.069231,0.105851,0.155999,0.733871,0.099141,0.143301,0.232692    Time: 00:00, 00:00
[DEBUG] 2018-04-23 14:34:12,804 -- UserKNN fold [8] has writeen item recommendations to .\Results\UserKNN-top-10-items fold [8].txt
[DEBUG] 2018-04-23 14:34:12,783 -- UserKNN fold [6]: 0.050000,0.031250,0.137153,0.142361,0.616137,0.135417,0.150912,0.218750    Time: 00:00, 00:00
[DEBUG] 2018-04-23 14:34:12,821 -- UserKNN fold [8]: 0.028571,0.035714,0.031746,0.052670,0.591538,0.019481,0.045335,0.079365    Time: 00:00, 00:00
[DEBUG] 2018-04-23 14:34:12,820 -- Build user similarity matrix ...
[DEBUG] 2018-04-23 14:34:12,817 -- UserKNN fold [9] has writeen item recommendations to .\Results\UserKNN-top-10-items fold [9].txt
[DEBUG] 2018-04-23 14:34:12,828 -- UserKNN fold [9]: 0.053333,0.046667,0.048091,0.153219,0.679642,0.032236,0.082326,0.119444    Time: 00:00, 00:00
[DEBUG] 2018-04-23 14:34:12,829 -- UserKNN: [knn, similarity, shrinkage] = [10, PCC, -1]
[DEBUG] 2018-04-23 14:34:12,829 -- UserKNN fold [10] evaluate test data ...
[DEBUG] 2018-04-23 14:34:12,833 -- UserKNN fold [10] has candidate items: 238
[DEBUG] 2018-04-23 14:34:12,852 -- UserKNN fold [10] has writeen item recommendations to .\Results\UserKNN-top-10-items fold [10].txt
[DEBUG] 2018-04-23 14:34:12,853 -- UserKNN fold [10]: 0.066667,0.066667,0.180556,0.289439,0.789570,0.174579,0.236348,0.332143    Time: 00:00, 00:00
[INFO ] 2018-04-23 14:34:12,866 -- UserKNN,0.058057,0.048757,0.079494,0.133765,0.660004,0.066027,0.106270,0.174602,,10, PCC, -1,'00:00','00:00'

Program Practice1_RecSys by LibRec.UserKNN [DESKTOP-J2CF07F@192.168.1.112] has completed!

C:\Users\user\Desktop\Data Mining\Project\Ques\Practice1_RecSys by LibRec>
```

Here precision@10 value is 0.048757

- **Precision@10 for COS at k=10**

```
# 3. leave-one-out -t threads -target u, i, r [--by-date]; 4. given-ratio -r ratio -target u, i, r [--by-date]; 5. given-n -N n -target u, i [--b
# optional: [-v validation-ratio] [--repeat n] [-cpu n] [--rand-seed n] [--test-view all, cold-start, trust-degree 1 5] [--early-stop loss, MAE,
evaluation.setup=cv -k 10 --test-view all --early-stop RMSE

# main option: is ranking prediction
item.ranking=on -topN -1 -ignore -1

# main option: is writing out recommendation results; [--fold-data --save-model]
output.setup=on -dir ./Results/ -verbose on, off --to-clipboard --to-file filmtrust-cv.txt

# Guava cache configuration
guava.cache.spec=maximumSize=200,expireAfterAccess=2m

# main option: is email notification enabled
email.setup=off -host smtp.email.com -port 465 -user xxx@email.com -password yyyy -auth true -to xxx@email.com

################################################ Model-based Methods ########################################
num.factors=5
num.max.iter=100

# options: -bold-driver, -decay ratio, -moment value
learn.rate=0.001 -max -1 -bold-driver

reg.lambda=0.1 -u 0.001 -i 0.001 -b 0.001 -s 0.001

# probabilistic graphic models
pgm.setup=-alpha 2 -beta 0.5 -burn-in 300 -sample-lag 10 -interval 100

################################################ Memory-based Methods ########################################
# similarity method: PCC, COS, COS-Binary, MSD, CPC, exJaccard; -1 to disable shrinking;
similarity=COS
num.shrinkage=-1

# neighborhood size; -1 to use as many as possible.
num.neighbors=10

################################################ Method-specific Settings ####################################
AoBPR=-lambda 0.3
```



Here Precision@10 is 0.099972

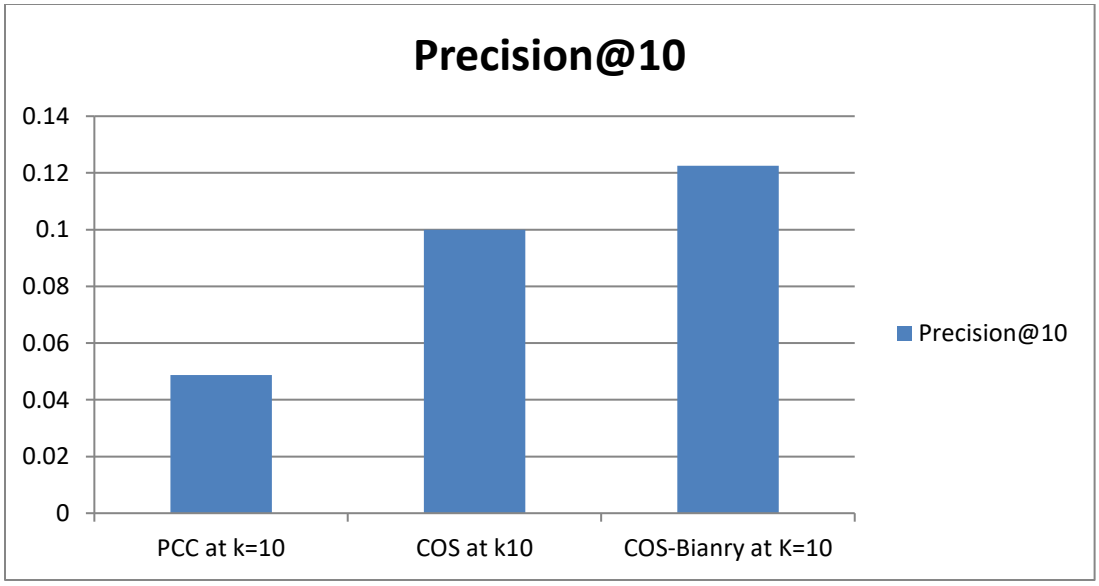
- **Precision@10 for COS-Binary at k=10**

```
# 3. leave-one-out -t threads -target u, i, r [--by-date]; 4. given-ratio -r ratio -target u, i, r [--by-date]; 5. given-n -N n -targ
# optional: [-v validation-ratio] [--repeat n] [-cpu n] [--rand-seed n] [--test-view all, cold-start, trust-degree 1 5] [--early-stop
evaluation.setup=cv -k 10 --test-view all --early-stop RMSE

# main option: is ranking prediction
item.ranking=on -topN -1 -ignore -1

# main option: is writing out recommendation results; [--fold-data --save-model]
output.setup=on -dir ./Results/ -verbose on, off --to-clipboard --to-file filmtrust-cv.txt

# Guava cache configuration
guava.cache.spec=maximumSize=200,expireAfterAccess=2m

# main option: is email notification enabled
email.setup=off -host smtp.email.com -port 465 -user xxx@email.com -password yyyy -auth true -to xxx@email.com

################################################ Model-based Methods ########################################
num.factors=5
num.max.iter=100

# options: -bold-driver, -decay ratio, -moment value
learn.rate=0.001 -max -1 -bold-driver

reg.lambda=0.1 -u 0.001 -i 0.001 -b 0.001 -s 0.001

# probabilistic graphic models
pgm.setup=-alpha 2 -beta 0.5 -burn-in 300 -sample-lag 10 -interval 100

################################################ Memory-based Methods ########################################
# similarity method: PCC, COS, COS-Binary, MSD, CPC, exJaccard; -1 to disable shrinking;
similarity=COS-Binary
num.shrinkage=-1

# neighborhood size; -1 to use as many as possible.
num.neighbors=10

################################################ Method-specific Settings ########################################
AoBPR=-lambda 0.3
```

```
[DEBUG] 2018-04-23 14:36:04,458 -- Fold [9]: training amount: 657, test amount: 73
[DEBUG] 2018-04-23 14:36:04,377 -- UserKNN: [knn, similarity, shrinkage] = [10, COS-Binary, -1]
[DEBUG] 2018-04-23 14:36:04,495 -- UserKNN fold [4] evaluate test data ...
[DEBUG] 2018-04-23 14:36:04,463 -- UserKNN fold [3] has candidate items: 242
[DEBUG] 2018-04-23 14:36:04,461 -- UserKNN fold [5] has candidate items: 248
[DEBUG] 2018-04-23 14:36:04,460 -- UserKNN: [knn, similarity, shrinkage] = [10, COS-Binary, -1]
[DEBUG] 2018-04-23 14:36:04,511 -- UserKNN fold [6] evaluate test data ...
[DEBUG] 2018-04-23 14:36:04,532 -- UserKNN fold [6] has candidate items: 242
[DEBUG] 2018-04-23 14:36:04,451 -- UserKNN: [knn, similarity, shrinkage] = [10, COS-Binary, -1]
[DEBUG] 2018-04-23 14:36:04,556 -- UserKNN fold [3] has writeen item recommendations to .\Results\UserKNN-top-10-items fold [3].txt
[DEBUG] 2018-04-23 14:36:04,510 -- Build user similarity matrix ...
[DEBUG] 2018-04-23 14:36:04,510 -- UserKNN fold [2] has writeen item recommendations to .\Results\UserKNN-top-10-items fold [2].txt
[DEBUG] 2018-04-23 14:36:04,509 -- UserKNN fold [1] has writeen item recommendations to .\Results\UserKNN-top-10-items fold [1].txt
[DEBUG] 2018-04-23 14:36:04,505 -- UserKNN fold [4] has candidate items: 239
[DEBUG] 2018-04-23 14:36:04,502 -- Build user similarity matrix ...
[DEBUG] 2018-04-23 14:36:04,498 -- Fold [10]: training amount: 657, test amount: 73
[DEBUG] 2018-04-23 14:36:04,582 -- UserKNN: [knn, similarity, shrinkage] = [10, COS-Binary, -1]
[DEBUG] 2018-04-23 14:36:04,579 -- UserKNN fold [1]: 0.107692,0.130769,0.084127,0.172910,0.731527,0.056668,0.128881,0.166117    Time: 00:00, 00:00
[DEBUG] 2018-04-23 14:36:04,572 -- UserKNN: [knn, similarity, shrinkage] = [10, COS-Binary, -1]
[DEBUG] 2018-04-23 14:36:04,594 -- Build user similarity matrix ...
[DEBUG] 2018-04-23 14:36:04,572 -- UserKNN fold [2]: 0.171429,0.114286,0.167687,0.207993,0.756034,0.108328,0.183327,0.313889    Time: 00:00, 00:00
[DEBUG] 2018-04-23 14:36:04,566 -- UserKNN fold [3]: 0.153846,0.138462,0.112762,0.257576,0.773235,0.084172,0.182408,0.266026    Time: 00:00, 00:00
[DEBUG] 2018-04-23 14:36:04,565 -- UserKNN fold [5] has writeen item recommendations to .\Results\UserKNN-top-10-items fold [5].txt
[DEBUG] 2018-04-23 14:36:04,559 -- UserKNN fold [7] evaluate test data ...
[DEBUG] 2018-04-23 14:36:04,605 -- UserKNN: [knn, similarity, shrinkage] = [10, COS-Binary, -1]
[DEBUG] 2018-04-23 14:36:04,604 -- UserKNN fold [4] has writeen item recommendations to .\Results\UserKNN-top-10-items fold [4].txt
[DEBUG] 2018-04-23 14:36:04,594 -- UserKNN fold [9] evaluate test data ...
[DEBUG] 2018-04-23 14:36:04,588 -- UserKNN fold [6] has writeen item recommendations to .\Results\UserKNN-top-10-items fold [6].txt
[DEBUG] 2018-04-23 14:36:04,623 -- UserKNN fold [6]: 0.137500,0.112500,0.129593,0.231581,0.700664,0.116149,0.187464,0.290179    Time: 00:00, 00:00
[DEBUG] 2018-04-23 14:36:04,582 -- UserKNN fold [8] evaluate test data ...
[DEBUG] 2018-04-23 14:36:04,623 -- UserKNN fold [9] has candidate items: 244
[DEBUG] 2018-04-23 14:36:04,615 -- UserKNN fold [4]: 0.153846,0.130769,0.122955,0.181807,0.739605,0.089974,0.160484,0.239316    Time: 00:00, 00:00
[DEBUG] 2018-04-23 14:36:04,614 -- UserKNN fold [10] evaluate test data ...
[DEBUG] 2018-04-23 14:36:04,613 -- UserKNN fold [7] has candidate items: 241
[DEBUG] 2018-04-23 14:36:04,612 -- UserKNN fold [5]: 0.100000,0.128571,0.059019,0.149062,0.689498,0.063986,0.124366,0.188776    Time: 00:00, 00:00
[DEBUG] 2018-04-23 14:36:04,627 -- UserKNN fold [8] has candidate items: 237
[DEBUG] 2018-04-23 14:36:04,640 -- UserKNN fold [10] has candidate items: 244
[DEBUG] 2018-04-23 14:36:04,658 -- UserKNN fold [9] has writeen item recommendations to .\Results\UserKNN-top-10-items fold [9].txt
[DEBUG] 2018-04-23 14:36:04,658 -- UserKNN fold [9]: 0.200000,0.133333,0.142301,0.187771,0.763103,0.115055,0.198759,0.350000    Time: 00:00, 00:00
[DEBUG] 2018-04-23 14:36:04,662 -- UserKNN fold [7] has writeen item recommendations to .\Results\UserKNN-top-10-items fold [7].txt
[DEBUG] 2018-04-23 14:36:04,665 -- UserKNN fold [7]: 0.171429,0.142857,0.138095,0.202381,0.726786,0.139953,0.221400,0.391156    Time: 00:00, 00:00
[DEBUG] 2018-04-23 14:36:04,677 -- UserKNN fold [8] has writeen item recommendations to .\Results\UserKNN-top-10-items fold [8].txt
[DEBUG] 2018-04-23 14:36:04,677 -- UserKNN fold [8]: 0.075000,0.087500,0.050000,0.192262,0.691878,0.053728,0.115687,0.124876    Time: 00:00, 00:00
[DEBUG] 2018-04-23 14:36:04,678 -- UserKNN fold [10] has writeen item recommendations to .\Results\UserKNN-top-10-items fold [10].txt
[DEBUG] 2018-04-23 14:36:04,685 -- UserKNN fold [10]: 0.150000,0.106250,0.117163,0.159077,0.697675,0.071830,0.132698,0.179167    Time: 00:00, 00:00
[INFO ] 2018-04-23 14:36:04,687 -- UserKNN,0.142074,0.122530,0.112370,0.194242,0.727001,0.089984,0.163547,0.250950,,10, COS-Binary, -1,'00:00','00:00'

Program Practice1_RecSys by LibRec.UserKNN [DESKTOP-J2CF07F@192.168.1.112] has completed!

C:\Users\user\Desktop\Data Mining\Project\Ques\Practice1_RecSys by LibRec>
```

Here Precision@10 is 0.122530

Next we will draw bar graph comparing precision@10 for all 3 similarities:



From above we see that Cos-Binary has the highest value as compared to other two.

Next we will use Biased Matrix Factorization and find MAE:

- **BMF when num.factor =10**

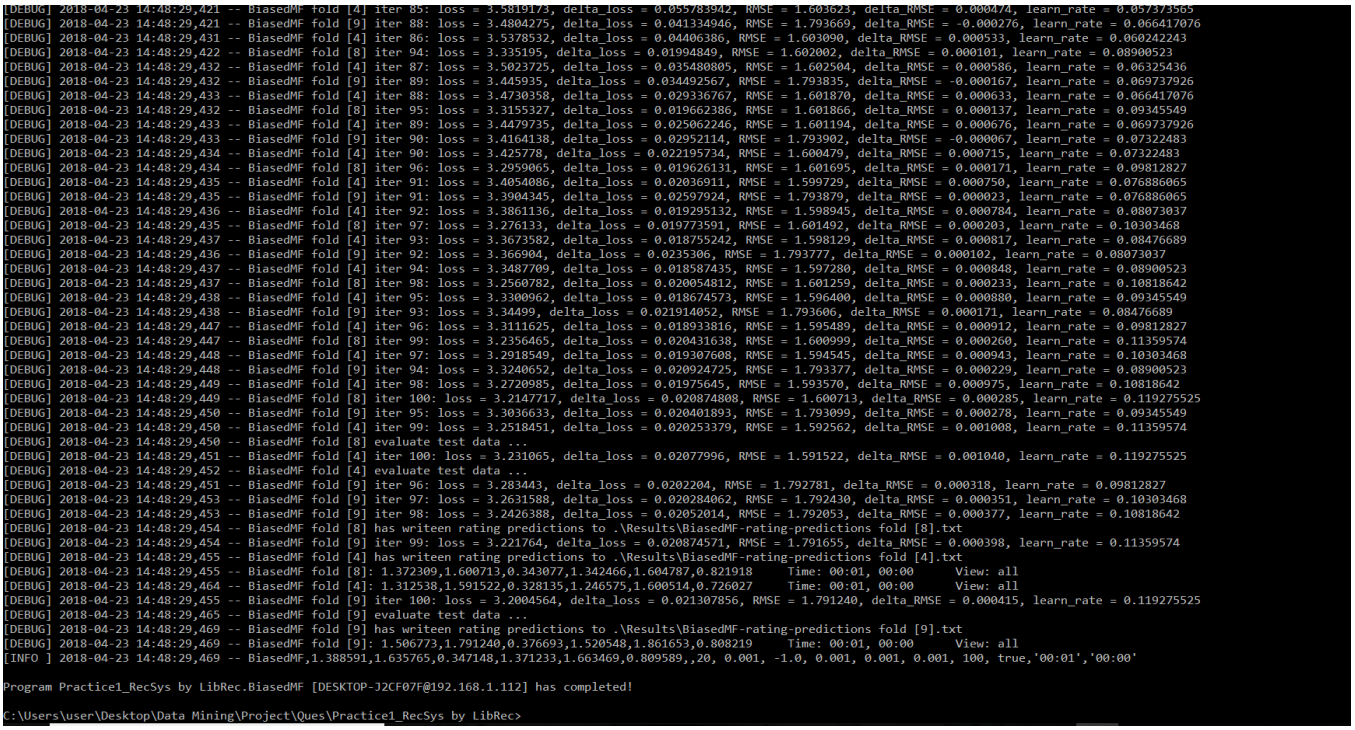
```
# ranking: BPR, WBPR, GBPR, SBPR, CLiMF, WRMF, RankALS, RankSGD, FISMrmse, FISMauc, LDA, BHfree, BUCM, LRMF;
# rating:  UserKNN, ItemKNN, BiasedMF, LDCC, PMF, BPMF, SVD++, timeSVD++, SocialMF, RSTE, TrustMF, SoRec, SoReg, TrustSVD, URP, G
# others:  GlobalAvg, UserAvg, ItemAvg, UserCluster, ItemCluster, Random, Constant, MostPop; NMF, SlopeOne, Hybrid, PD, AR, PRank
recommender=BiasedMF

# main option: 1. test-set -f test-file-path; 2. cv (cross validation) -k k-folds [-p on, off]
# 3. leave-one-out -t threads -target u, i, r [--by-date]; 4. given-ratio -r ratio -target u, i, r [--by-date]; 5. given-n -N n -
# optional: [-v validation-ratio] [--repeat n] [-cpu n] [--rand-seed n] [--test-view all, cold-start, trust-degree 1 5] [--early-
evaluation.setup=cv -k 10 --test-view all --early-stop RMSE

# main option: is ranking prediction
item.ranking=off -topN -1 -ignore -1

# main option: is writing out recommendation results; [--fold-data --save-model]
output.setup=on -dir ./Results/ -verbose on, off --to-clipboard --to-file filmtrust-cv.txt

# Guava cache configuration
guava.cache.spec=maximumSize=200,expireAfterAccess=2m

# main option: is email notification enabled
email.setup=off -host smtp.email.com -port 465 -user xxx@email.com -password yyyy -auth true -to xxx@email.com

################################################## Model-based Methods ########################################
num.factors=10
num.max.iter=100

# options: -bold-driver, -decay ratio, -moment value
learn.rate=0.001 -max -1 -bold-driver

reg.lambda=0.1 -u 0.001 -i 0.001 -b 0.001 -s 0.001

# probabilistic graphic models
pgm.setup=-alpha 2 -beta 0.5 -burn-in 300 -sample-lag 10 -interval 100

################################################## Memory-based Methods ########################################
# similarity method: PCC, COS, COS-Binary, MSD, CPC, exJaccard; -1 to disable shrinking;
similarity=COS-Binary
num.shrinkage=-1
```

```
[DEBUG] 2018-04-23 14:45:48,972 -- BiasedMF fold [4] iter 96: loss = 3.2779543, delta_loss = 0.020859238, RMSE = 1.666564, delta_RMSE = 0.000687, learn_rate = 0.09812827
[DEBUG] 2018-04-23 14:45:48,959 -- BiasedMF fold [8] iter 89: loss = 3.5757868, delta_loss = 0.11554303, RMSE = 1.664872, delta_RMSE = -0.001258, learn_rate = 0.069737926
[DEBUG] 2018-04-23 14:45:48,999 -- BiasedMF fold [4] iter 97: loss = 3.257675, delta_loss = 0.02027943, RMSE = 1.665838, delta_RMSE = 0.000726, learn_rate = 0.10303468
[DEBUG] 2018-04-23 14:45:48,992 -- BiasedMF fold [5]: 1.377811,1.623380,0.344453,1.410959,1.692044,0.821918     Time: 00:01, 00:00      View: all
[DEBUG] 2018-04-23 14:45:48,986 -- BiasedMF fold [2] iter 83: loss = 4.352048, delta_loss = 0.36160088, RMSE = 1.636091, delta_RMSE = -0.000454, learn_rate = 0.052039515
[DEBUG] 2018-04-23 14:45:49,016 -- BiasedMF fold [4] iter 98: loss = 3.2375984, delta_loss = 0.02007654, RMSE = 1.665076, delta_RMSE = 0.000762, learn_rate = 0.10818642
[DEBUG] 2018-04-23 14:45:49,008 -- BiasedMF fold [8] iter 90: loss = 3.4817317, delta_loss = 0.094055235, RMSE = 1.665948, delta_RMSE = -0.001076, learn_rate = 0.07322483
[DEBUG] 2018-04-23 14:45:49,039 -- BiasedMF fold [4] iter 99: loss = 3.2174726, delta_loss = 0.020125665, RMSE = 1.664279, delta_RMSE = 0.000797, learn_rate = 0.11359574
[DEBUG] 2018-04-23 14:45:49,042 -- BiasedMF fold [4] iter 100: loss = 3.1971288, delta_loss = 0.020343984, RMSE = 1.663447, delta_RMSE = 0.000832, learn_rate = 0.119275525
[DEBUG] 2018-04-23 14:45:49,029 -- BiasedMF fold [2] iter 84: loss = 4.07696, delta_loss = 0.2750879, RMSE = 1.636438, delta_RMSE = -0.000347, learn_rate = 0.054641493
[DEBUG] 2018-04-23 14:45:49,044 -- BiasedMF fold [4] evaluate test data ...
[DEBUG] 2018-04-23 14:45:49,041 -- BiasedMF fold [8] iter 91: loss = 3.4057417, delta_loss = 0.07598994, RMSE = 1.666848, delta_RMSE = -0.000900, learn_rate = 0.076886065
[DEBUG] 2018-04-23 14:45:49,056 -- BiasedMF fold [2] iter 85: loss = 3.870726, delta_loss = 0.20623417, RMSE = 1.636681, delta_RMSE = -0.000243, learn_rate = 0.057373565
[DEBUG] 2018-04-23 14:45:49,068 -- BiasedMF fold [2] iter 86: loss = 3.7179112, delta_loss = 0.15281479, RMSE = 1.636692, delta_RMSE = -0.000011, learn_rate = 0.060242243
[DEBUG] 2018-04-23 14:45:49,073 -- BiasedMF fold [2] iter 87: loss = 3.605448, delta_loss = 0.11246331, RMSE = 1.636582, delta_RMSE = 0.000110, learn_rate = 0.06325436
[DEBUG] 2018-04-23 14:45:49,069 -- BiasedMF fold [8] iter 92: loss = 3.3446226, delta_loss = 0.06111912, RMSE = 1.667580, delta_RMSE = -0.000732, learn_rate = 0.08073037
[DEBUG] 2018-04-23 14:45:49,083 -- BiasedMF fold [4] has writeen rating predictions to .\Results\BiasedMF-rating-predictions fold [4].txt
[DEBUG] 2018-04-23 14:45:49,076 -- BiasedMF fold [2] iter 88: loss = 3.5226624, delta_loss = 0.0827855, RMSE = 1.636401, delta_RMSE = 0.000181, learn_rate = 0.066417076
[DEBUG] 2018-04-23 14:45:49,092 -- BiasedMF fold [4]: 1.375454,1.663447,0.343863,1.356164,1.724124,0.780822     Time: 00:02, 00:00      View: all
[DEBUG] 2018-04-23 14:45:49,089 -- BiasedMF fold [8] iter 93: loss = 3.2954233, delta_loss = 0.049199164, RMSE = 1.668155, delta_RMSE = -0.000574, learn_rate = 0.08476689
[DEBUG] 2018-04-23 14:45:49,093 -- BiasedMF fold [2] iter 89: loss = 3.461148, delta_loss = 0.061514374, RMSE = 1.636160, delta_RMSE = 0.000242, learn_rate = 0.069737926
[DEBUG] 2018-04-23 14:45:49,106 -- BiasedMF fold [8] iter 94: loss = 3.2554808, delta_loss = 0.039942678, RMSE = 1.668582, delta_RMSE = -0.000427, learn_rate = 0.08900523
[DEBUG] 2018-04-23 14:45:49,114 -- BiasedMF fold [2] iter 90: loss = 3.4145052, delta_loss = 0.0466429, RMSE = 1.635867, delta_RMSE = 0.000293, learn_rate = 0.07322483
[DEBUG] 2018-04-23 14:45:49,114 -- BiasedMF fold [8] iter 95: loss = 3.2224653, delta_loss = 0.0330153, RMSE = 1.668847, delta_RMSE = -0.000265, learn_rate = 0.09345549
[DEBUG] 2018-04-23 14:45:49,117 -- BiasedMF fold [2] iter 91: loss = 3.3780065, delta_loss = 0.036498755, RMSE = 1.635531, delta_RMSE = 0.000336, learn_rate = 0.076886065
[DEBUG] 2018-04-23 14:45:49,118 -- BiasedMF fold [8] iter 96: loss = 3.1944149, delta_loss = 0.028050534, RMSE = 1.668889, delta_RMSE = -0.000042, learn_rate = 0.09812827
[DEBUG] 2018-04-23 14:45:49,125 -- BiasedMF fold [2] iter 92: loss = 3.3482487, delta_loss = 0.029757747, RMSE = 1.635128, delta_RMSE = 0.000403, learn_rate = 0.08073037
[DEBUG] 2018-04-23 14:45:49,125 -- BiasedMF fold [8] iter 97: loss = 3.169741, delta_loss = 0.02467381, RMSE = 1.668866, delta_RMSE = 0.000024, learn_rate = 0.10303468
[DEBUG] 2018-04-23 14:45:49,128 -- BiasedMF fold [2] iter 93: loss = 3.3228354, delta_loss = 0.0254133, RMSE = 1.634699, delta_RMSE = 0.000429, learn_rate = 0.08476689
[DEBUG] 2018-04-23 14:45:49,139 -- BiasedMF fold [8] iter 98: loss = 3.1472132, delta_loss = 0.02252784, RMSE = 1.668785, delta_RMSE = 0.000081, learn_rate = 0.10818642
[DEBUG] 2018-04-23 14:45:49,141 -- BiasedMF fold [2] iter 94: loss = 3.300111, delta_loss = 0.022724362, RMSE = 1.634247, delta_RMSE = 0.000452, learn_rate = 0.08900523
[DEBUG] 2018-04-23 14:45:49,150 -- BiasedMF fold [8] iter 99: loss = 3.1259196, delta_loss = 0.021293497, RMSE = 1.668656, delta_RMSE = 0.000129, learn_rate = 0.11359574
[DEBUG] 2018-04-23 14:45:49,152 -- BiasedMF fold [2] iter 95: loss = 3.278952, delta_loss = 0.021159105, RMSE = 1.633774, delta_RMSE = 0.000473, learn_rate = 0.09345549
[DEBUG] 2018-04-23 14:45:49,159 -- BiasedMF fold [8] iter 100: loss = 3.1052165, delta_loss = 0.020703174, RMSE = 1.668485, delta_RMSE = 0.000170, learn_rate = 0.119275525
[DEBUG] 2018-04-23 14:45:49,159 -- BiasedMF fold [8] evaluate test data ...
[DEBUG] 2018-04-23 14:45:49,159 -- BiasedMF fold [2] iter 96: loss = 3.2586079, delta_loss = 0.02034405, RMSE = 1.633281, delta_RMSE = 0.000493, learn_rate = 0.09812827
[DEBUG] 2018-04-23 14:45:49,160 -- BiasedMF fold [2] iter 97: loss = 3.2385857, delta_loss = 0.020022107, RMSE = 1.632768, delta_RMSE = 0.000512, learn_rate = 0.10303468
[DEBUG] 2018-04-23 14:45:49,166 -- BiasedMF fold [8] has writeen rating predictions to .\Results\BiasedMF-rating-predictions fold [8].txt
[DEBUG] 2018-04-23 14:45:49,169 -- BiasedMF fold [2] iter 98: loss = 3.218566, delta_loss = 0.020019678, RMSE = 1.632238, delta_RMSE = 0.000530, learn_rate = 0.10818642
[DEBUG] 2018-04-23 14:45:49,171 -- BiasedMF fold [2] iter 99: loss = 3.1983445, delta_loss = 0.020221652, RMSE = 1.631691, delta_RMSE = 0.000547, learn_rate = 0.11359574
[DEBUG] 2018-04-23 14:45:49,171 -- BiasedMF fold [2] iter 100: loss = 3.1777916, delta_loss = 0.020552866, RMSE = 1.631127, delta_RMSE = 0.000563, learn_rate = 0.119275525
[DEBUG] 2018-04-23 14:45:49,170 -- BiasedMF fold [8]: 1.411936,1.668485,0.352984,1.383562,1.651069,0.849315     Time: 00:02, 00:00      View: all
[DEBUG] 2018-04-23 14:45:49,171 -- BiasedMF fold [2] evaluate test data ...
[DEBUG] 2018-04-23 14:45:49,175 -- BiasedMF fold [2] has writeen rating predictions to .\Results\BiasedMF-rating-predictions fold [2].txt
[DEBUG] 2018-04-23 14:45:49,175 -- BiasedMF fold [2]: 1.365085,1.631127,0.341271,1.397260,1.679856,0.849315     Time: 00:02, 00:00      View: all
[INFO ] 2018-04-23 14:45:49,176 -- BiasedMF,1.388014,1.638807,0.347003,1.383562,1.664981,0.824658,,10, 0.001, -1.0, 0.001, 0.001, 0.001, 100, true,'00:01','00:00'

Program Practice1_RecSys by LibRec.BiasedMF [DESKTOP-J2CF07F@192.168.1.112] has completed!

C:\Users\user\Desktop\Data Mining\Project\Ques\Practice1_RecSys by LibRec>
```

Here MAE value is 1.388014

- **BMF when num.factor =20**

```
35 ################################################ Model-based Methods ########################################
36 num.factors=20
37 num.max.iter=100
38
39 # options: -bold-driver, -decay ratio, -moment value
40 learn.rate=0.001 -max -1 -bold-driver
41
42 reg.lambda=0.1 -u 0.001 -i 0.001 -b 0.001 -s 0.001
43
44 # probabilistic graphic models
45 pgm.setup=-alpha 2 -beta 0.5 -burn-in 300 -sample-lag 10 -interval 100
46
```



Here MAE value is 1.388591

- **BMF when num.factor =5**

```
35 ################################################ Model-based Methods ########################################
36 num.factors=5
37 num.max.iter=100
38
39 # options: -bold-driver, -decay ratio, -moment value
40 learn.rate=0.001 -max -1 -bold-driver
41
42 reg.lambda=0.1 -u 0.001 -i 0.001 -b 0.001 -s 0.001
43
44 # probabilistic graphic models
45 pgm.setup=-alpha 2 -beta 0.5 -burn-in 300 -sample-lag 10 -interval 100
46
```

```
[DEBUG] 2018-04-23 14:50:05,870 -- BiasedMF fold [6] evaluate test data ...
[DEBUG] 2018-04-23 14:50:05,873 -- BiasedMF fold [8] iter 90: loss = 26.233522, delta_loss = 0.37837678, RMSE = 1.720462, delta_RMSE = -0.001042, learn_rate = 0.07322483
[DEBUG] 2018-04-23 14:50:05,871 -- BiasedMF fold [3] iter 79: loss = 28.009998, delta_loss = 2.0627341, RMSE = 1.610952, delta_RMSE = -0.002534, learn_rate = 0.042813037
[DEBUG] 2018-04-23 14:50:05,880 -- BiasedMF fold [8] iter 91: loss = 25.899286, delta_loss = 0.33423632, RMSE = 1.719396, delta_RMSE = 0.001065, learn_rate = 0.076886065
[DEBUG] 2018-04-23 14:50:05,875 -- BiasedMF fold [6] has writeen rating predictions to .\Results\BiasedMF-rating-predictions fold [6].txt
[DEBUG] 2018-04-23 14:50:05,881 -- BiasedMF fold [8] iter 92: loss = 25.579008, delta_loss = 0.32027864, RMSE = 1.718538, delta_RMSE = 0.000858, learn_rate = 0.08073037
[DEBUG] 2018-04-23 14:50:05,880 -- BiasedMF fold [3] iter 80: loss = 26.159777, delta_loss = 1.8502212, RMSE = 1.613336, delta_RMSE = -0.002384, learn_rate = 0.04495369
[DEBUG] 2018-04-23 14:50:05,893 -- BiasedMF fold [8] iter 93: loss = 25.22892, delta_loss = 0.35008717, RMSE = 1.718332, delta_RMSE = 0.000206, learn_rate = 0.08476689
[DEBUG] 2018-04-23 14:50:05,883 -- BiasedMF fold [6]: 1.498119,1.784644,0.374530,1.479452,1.805623,0.821918     Time: 00:02, 00:00      View: all
[DEBUG] 2018-04-23 14:50:05,895 -- BiasedMF fold [8] iter 94: loss = 24.789032, delta_loss = 0.4398883, RMSE = 1.718965, delta_RMSE = -0.000632, learn_rate = 0.08900523
[DEBUG] 2018-04-23 14:50:05,894 -- BiasedMF fold [3] iter 81: loss = 24.49169, delta_loss = 1.668088, RMSE = 1.616277, delta_RMSE = -0.002941, learn_rate = 0.047201376
[DEBUG] 2018-04-23 14:50:05,900 -- BiasedMF fold [8] iter 95: loss = 24.185812, delta_loss = 0.6032209, RMSE = 1.720655, delta_RMSE = -0.001690, learn_rate = 0.09345549
[DEBUG] 2018-04-23 14:50:05,901 -- BiasedMF fold [3] iter 82: loss = 22.980854, delta_loss = 1.5108352, RMSE = 1.619060, delta_RMSE = -0.002783, learn_rate = 0.049561445
[DEBUG] 2018-04-23 14:50:05,908 -- BiasedMF fold [8] iter 96: loss = 23.350471, delta_loss = 0.83533996, RMSE = 1.723594, delta_RMSE = -0.002939, learn_rate = 0.09812827
[DEBUG] 2018-04-23 14:50:05,908 -- BiasedMF fold [3] iter 83: loss = 21.60571, delta_loss = 1.3751451, RMSE = 1.621749, delta_RMSE = -0.002689, learn_rate = 0.052039515
[DEBUG] 2018-04-23 14:50:05,909 -- BiasedMF fold [8] iter 97: loss = 22.264135, delta_loss = 1.0863352, RMSE = 1.727840, delta_RMSE = -0.004247, learn_rate = 0.10303468
[DEBUG] 2018-04-23 14:50:05,912 -- BiasedMF fold [3] iter 84: loss = 20.347467, delta_loss = 1.2582412, RMSE = 1.625110, delta_RMSE = -0.003361, learn_rate = 0.054641493
[DEBUG] 2018-04-23 14:50:05,921 -- BiasedMF fold [8] iter 98: loss = 21.016071, delta_loss = 1.248065, RMSE = 1.732078, delta_RMSE = -0.004238, learn_rate = 0.10818642
[DEBUG] 2018-04-23 14:50:05,924 -- BiasedMF fold [3] iter 85: loss = 19.190231, delta_loss = 1.1572362, RMSE = 1.629190, delta_RMSE = -0.004080, learn_rate = 0.057373565
[DEBUG] 2018-04-23 14:50:05,931 -- BiasedMF fold [8] iter 99: loss = 19.803476, delta_loss = 1.2125953, RMSE = 1.732398, delta_RMSE = -0.000320, learn_rate = 0.11359574
[DEBUG] 2018-04-23 14:50:05,932 -- BiasedMF fold [3] iter 86: loss = 18.121141, delta_loss = 1.0690906, RMSE = 1.634033, delta_RMSE = -0.004843, learn_rate = 0.060242243
[DEBUG] 2018-04-23 14:50:05,936 -- BiasedMF fold [8] iter 100: loss = 18.810163, delta_loss = 0.9933115, RMSE = 1.732509, delta_RMSE = -0.000111, learn_rate = 0.119275525
[DEBUG] 2018-04-23 14:50:05,943 -- BiasedMF fold [8] evaluate test data ...
[DEBUG] 2018-04-23 14:50:05,936 -- BiasedMF fold [3] iter 87: loss = 17.13026, delta_loss = 0.9908801, RMSE = 1.639669, delta_RMSE = -0.005636, learn_rate = 0.06325436
[DEBUG] 2018-04-23 14:50:05,954 -- BiasedMF fold [8] has writeen rating predictions to .\Results\BiasedMF-rating-predictions fold [8].txt
[DEBUG] 2018-04-23 14:50:05,954 -- BiasedMF fold [3] iter 88: loss = 16.210146, delta_loss = 0.9201152, RMSE = 1.644933, delta_RMSE = -0.005264, learn_rate = 0.066417076
[DEBUG] 2018-04-23 14:50:05,956 -- BiasedMF fold [8]: 1.419153,1.732509,0.354788,1.424658,1.751711,0.794521     Time: 00:02, 00:00      View: all
[DEBUG] 2018-04-23 14:50:05,963 -- BiasedMF fold [3] iter 89: loss = 15.355217, delta_loss = 0.8549286, RMSE = 1.648169, delta_RMSE = -0.003236, learn_rate = 0.069737926
[DEBUG] 2018-04-23 14:50:05,976 -- BiasedMF fold [3] iter 90: loss = 14.561147, delta_loss = 0.79407036, RMSE = 1.652068, delta_RMSE = -0.003898, learn_rate = 0.07322483
[DEBUG] 2018-04-23 14:50:05,988 -- BiasedMF fold [3] iter 91: loss = 13.824392, delta_loss = 0.7367542, RMSE = 1.656642, delta_RMSE = -0.004574, learn_rate = 0.076886065
[DEBUG] 2018-04-23 14:50:05,993 -- BiasedMF fold [3] iter 92: loss = 13.141932, delta_loss = 0.682461, RMSE = 1.661879, delta_RMSE = -0.005237, learn_rate = 0.08073037
[DEBUG] 2018-04-23 14:50:06,002 -- BiasedMF fold [3] iter 93: loss = 12.511127, delta_loss = 0.6308043, RMSE = 1.667736, delta_RMSE = -0.005857, learn_rate = 0.08476689
[DEBUG] 2018-04-23 14:50:06,007 -- BiasedMF fold [3] iter 94: loss = 11.929614, delta_loss = 0.5815133, RMSE = 1.674131, delta_RMSE = -0.006394, learn_rate = 0.08900523
[DEBUG] 2018-04-23 14:50:06,014 -- BiasedMF fold [3] iter 95: loss = 11.395101, delta_loss = 0.5345136, RMSE = 1.680941, delta_RMSE = -0.006810, learn_rate = 0.09345549
[DEBUG] 2018-04-23 14:50:06,019 -- BiasedMF fold [3] iter 96: loss = 10.905101, delta_loss = 0.48999932, RMSE = 1.688010, delta_RMSE = -0.007069, learn_rate = 0.09812827
[DEBUG] 2018-04-23 14:50:06,020 -- BiasedMF fold [3] iter 97: loss = 10.456746, delta_loss = 0.44835487, RMSE = 1.695159, delta_RMSE = -0.007149, learn_rate = 0.10303468
[DEBUG] 2018-04-23 14:50:06,027 -- BiasedMF fold [3] iter 98: loss = 10.046903, delta_loss = 0.40984404, RMSE = 1.702204, delta_RMSE = -0.007044, learn_rate = 0.10818642
[DEBUG] 2018-04-23 14:50:06,029 -- BiasedMF fold [3] iter 99: loss = 9.672762, delta_loss = 0.37414056, RMSE = 1.705472, delta_RMSE = -0.003269, learn_rate = 0.11359574
[DEBUG] 2018-04-23 14:50:06,037 -- BiasedMF fold [3] iter 100: loss = 9.332826, delta_loss = 0.33993638, RMSE = 1.707800, delta_RMSE = -0.002328, learn_rate = 0.119275525
[DEBUG] 2018-04-23 14:50:06,038 -- BiasedMF fold [3] evaluate test data ...
[DEBUG] 2018-04-23 14:50:06,046 -- BiasedMF fold [3] has writeen rating predictions to .\Results\BiasedMF-rating-predictions fold [3].txt
[DEBUG] 2018-04-23 14:50:06,050 -- BiasedMF fold [3]: 1.435855,1.707800,0.358964,1.369863,1.679856,0.808219     Time: 00:02, 00:00      View: all
[INFO ] 2018-04-23 14:50:06,056 -- BiasedMF,1.493506,1.784263,0.373377,1.473973,1.801463,0.813699,,5, 0.001, -1.0, 0.001, 0.001, 0.001, 100, true,'00:02','00:00'

Program Practice1_RecSys by LibRec.BiasedMF [DESKTOP-J2CF07F@192.168.1.112] has completed!

C:\Users\user\Desktop\Data Mining\Project\Ques\Practice1_RecSys by LibRec>
```

Here MAE value is 1.493506

- **Change learn.rate to 0.01:**

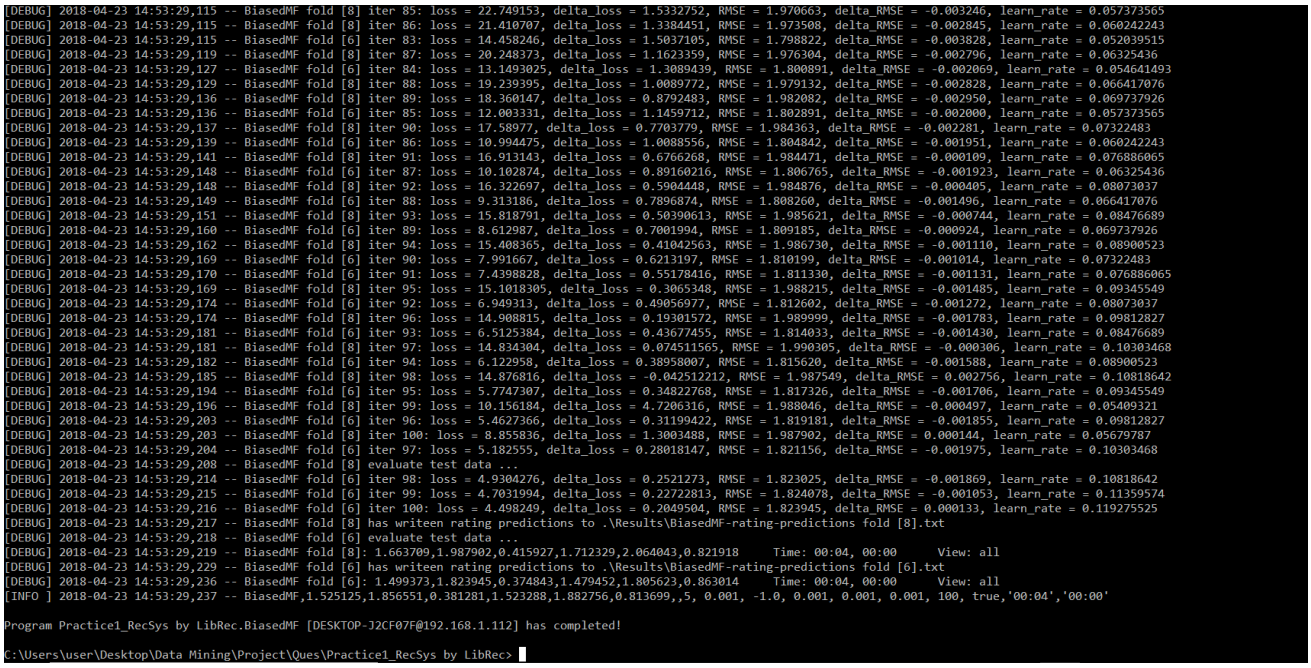
```
36 num.factors=5
37 num.max.iter=100
38
39 # options: -bold-driver, -decay ratio, -moment value
40 learn.rate=0.01 -max -1 -bold-driver
41
42 reg.lambda=0.1 -u 0.001 -i 0.001 -b 0.001 -s 0.001
43
44 # probabilistic graphic models
45 pgm.setup=-alpha 2 -beta 0.5 -burn-in 300 -sample-lag 10 -interval 100
```

```
[DEBUG] 2018-04-23 14:51:25,013 -- BiasedMF fold [7] iter 84: loss = 2.858342, delta_loss = 0.022193749, RMSE = 1.802891, delta_RMSE = 0.000445, learn_rate = 0.1239036
[DEBUG] 2018-04-23 14:51:25,013 -- BiasedMF fold [4] iter 95: loss = 2.48942, delta_loss = 0.0051622726, RMSE = 1.899224, delta_RMSE = 0.000643, learn_rate = 0.2119172
[DEBUG] 2018-04-23 14:51:25,014 -- BiasedMF fold [7] iter 85: loss = 2.8393745, delta_loss = 0.018967278, RMSE = 1.802452, delta_RMSE = 0.000439, learn_rate = 0.13009878
[DEBUG] 2018-04-23 14:51:25,014 -- BiasedMF fold [4] iter 96: loss = 2.4866066, delta_loss = 0.0028133637, RMSE = 1.898570, delta_RMSE = 0.000653, learn_rate = 0.22251306
[DEBUG] 2018-04-23 14:51:25,014 -- BiasedMF fold [7] iter 86: loss = 2.822812, delta_loss = 0.016562544, RMSE = 1.802029, delta_RMSE = 0.000423, learn_rate = 0.13660371
[DEBUG] 2018-04-23 14:51:25,015 -- BiasedMF fold [4] iter 97: loss = 2.4868772, delta_loss = -2.7066976E-4, RMSE = 1.897926, delta_RMSE = 0.000644, learn_rate = 0.23363872
[DEBUG] 2018-04-23 14:51:25,025 -- BiasedMF fold [7] iter 87: loss = 2.80817, delta_loss = 0.014642028, RMSE = 1.801620, delta_RMSE = 0.000409, learn_rate = 0.1434339
[DEBUG] 2018-04-23 14:51:25,025 -- BiasedMF fold [4] iter 98: loss = 2.4615693, delta_loss = 0.025307907, RMSE = 1.897255, delta_RMSE = 0.000671, learn_rate = 0.11681936
[DEBUG] 2018-04-23 14:51:25,026 -- BiasedMF fold [7] iter 88: loss = 2.7952623, delta_loss = 0.012907786, RMSE = 1.801219, delta_RMSE = 0.000401, learn_rate = 0.1506056
[DEBUG] 2018-04-23 14:51:25,027 -- BiasedMF fold [4] iter 99: loss = 2.4228005, delta_loss = 0.038768798, RMSE = 1.896734, delta_RMSE = 0.000521, learn_rate = 0.122660324
[DEBUG] 2018-04-23 14:51:25,033 -- BiasedMF fold [7] iter 89: loss = 2.7839048, delta_loss = 0.011357511, RMSE = 1.800846, delta_RMSE = 0.000374, learn_rate = 0.15813588
[DEBUG] 2018-04-23 14:51:25,034 -- BiasedMF fold [4] iter 100: loss = 2.4039218, delta_loss = 0.018878784, RMSE = 1.896458, delta_RMSE = 0.000276, learn_rate = 0.12879334
[DEBUG] 2018-04-23 14:51:25,034 -- BiasedMF fold [7] iter 90: loss = 2.7742229, delta_loss = 0.009681891, RMSE = 1.800519, delta_RMSE = 0.000327, learn_rate = 0.16604267
[DEBUG] 2018-04-23 14:51:25,034 -- BiasedMF fold [4] evaluate test data ...
[DEBUG] 2018-04-23 14:51:25,035 -- BiasedMF fold [7] iter 91: loss = 2.7666543, delta_loss = 0.007568544, RMSE = 1.800242, delta_RMSE = 0.000277, learn_rate = 0.17434481
[DEBUG] 2018-04-23 14:51:25,043 -- BiasedMF fold [7] iter 92: loss = 2.7620418, delta_loss = 0.0046124896, RMSE = 1.799997, delta_RMSE = 0.000245, learn_rate = 0.18306205
[DEBUG] 2018-04-23 14:51:25,044 -- BiasedMF fold [7] iter 93: loss = 2.761683, delta_loss = 3.5886874E-4, RMSE = 1.799762, delta_RMSE = 0.000235, learn_rate = 0.19221514
[DEBUG] 2018-04-23 14:51:25,044 -- BiasedMF fold [7] iter 94: loss = 2.7677283, delta_loss = -0.00604526, RMSE = 1.799517, delta_RMSE = 0.000245, learn_rate = 0.2018259
[DEBUG] 2018-04-23 14:51:25,045 -- BiasedMF fold [4] has writeen rating predictions to .\Results\BiasedMF-rating-predictions fold [4].txt
[DEBUG] 2018-04-23 14:51:25,045 -- BiasedMF fold [7] iter 95: loss = 2.7215834, delta_loss = 0.046144772, RMSE = 1.798668, delta_RMSE = 0.000849, learn_rate = 0.10091295
[DEBUG] 2018-04-23 14:51:25,045 -- BiasedMF fold [4]: 1.616047,1.896458,0.404012,1.616438,1.930292,0.863014     Time: 00:01, 00:00      View: all
[DEBUG] 2018-04-23 14:51:25,045 -- BiasedMF fold [7] iter 96: loss = 2.6719277, delta_loss = 0.04965576, RMSE = 1.798341, delta_RMSE = 0.000327, learn_rate = 0.1059586
[DEBUG] 2018-04-23 14:51:25,046 -- BiasedMF fold [7] iter 97: loss = 2.6509814, delta_loss = 0.020946253, RMSE = 1.797997, delta_RMSE = 0.000344, learn_rate = 0.11125653
[DEBUG] 2018-04-23 14:51:25,047 -- BiasedMF fold [7] iter 98: loss = 2.6358302, delta_loss = 0.01515123, RMSE = 1.797722, delta_RMSE = 0.000274, learn_rate = 0.11681936
[DEBUG] 2018-04-23 14:51:25,047 -- BiasedMF fold [7] iter 99: loss = 2.6243026, delta_loss = 0.011527606, RMSE = 1.797502, delta_RMSE = 0.000220, learn_rate = 0.122660324
[DEBUG] 2018-04-23 14:51:25,047 -- BiasedMF fold [7] iter 100: loss = 2.6141543, delta_loss = 0.010148391, RMSE = 1.797276, delta_RMSE = 0.000226, learn_rate = 0.12879334
[DEBUG] 2018-04-23 14:51:25,048 -- BiasedMF fold [7] evaluate test data ...
[DEBUG] 2018-04-23 14:51:25,050 -- BiasedMF fold [7] has writeen rating predictions to .\Results\BiasedMF-rating-predictions fold [7].txt
[DEBUG] 2018-04-23 14:51:25,058 -- BiasedMF fold [7]: 1.453611,1.797276,0.363403,1.479452,1.828241,0.835616     Time: 00:01, 00:00      View: all
[INFO ] 2018-04-23 14:51:25,059 -- BiasedMF,1.538398,1.856494,0.384599,1.519178,1.874715,0.809589,,5, 0.01, -1.0, 0.001, 0.001, 0.001, 100, true,'00:01','00:00'

Program Practice1_RecSys by LibRec.BiasedMF [DESKTOP-J2CF07F@192.168.1.112] has completed!

C:\Users\user\Desktop\Data Mining\Project\Ques\Practice1_RecSys by LibRec>
```

Here MAE value is 1.538398

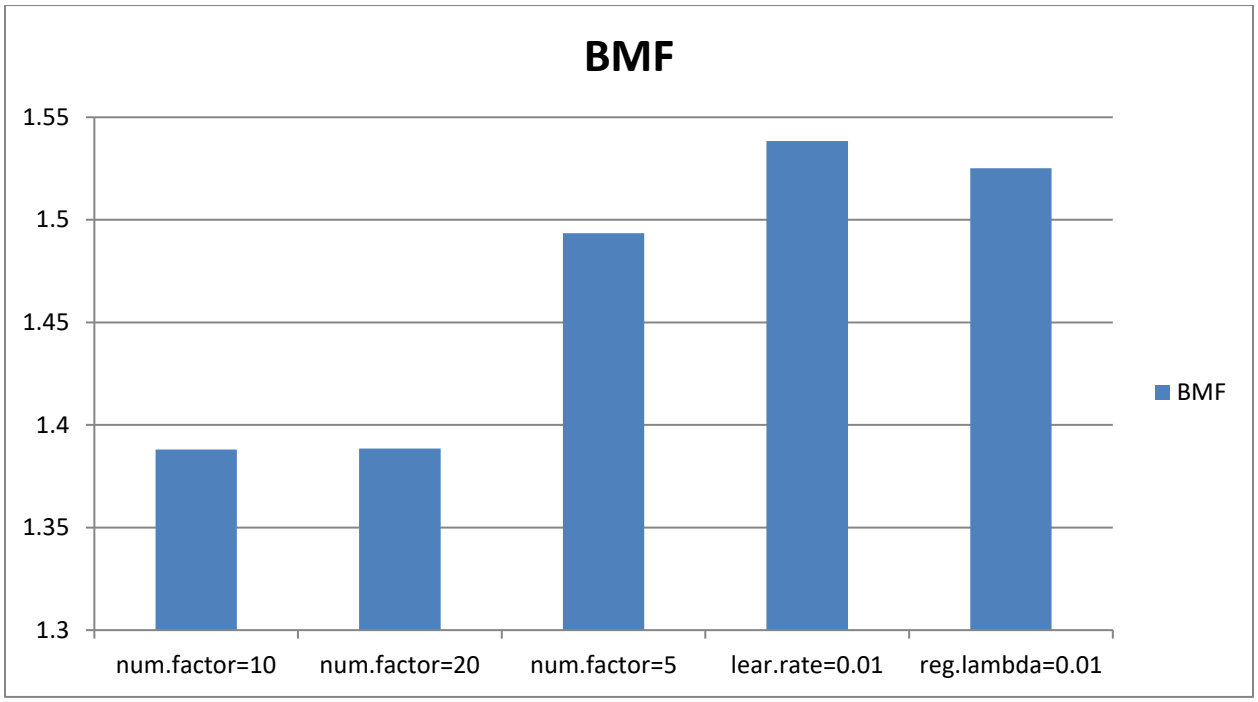
- **Change reg.lambda=0.01:**

```
################################################ Model-based Methods ########################################
num.factors=5
num.max.iter=100

# options: -bold-driver, -decay ratio, -moment value
learn.rate=0.001 -max -1 -bold-driver

reg.lambda=0.01 -u 0.001 -i 0.001 -b 0.001 -s 0.001

# probabilistic graphic models
pgm.setup=-alpha 2 -beta 0.5 -burn-in 300 -sample-lag 10 -interval 100

```



Here Mae value is 1.525125.

We will now compare all MAE values:

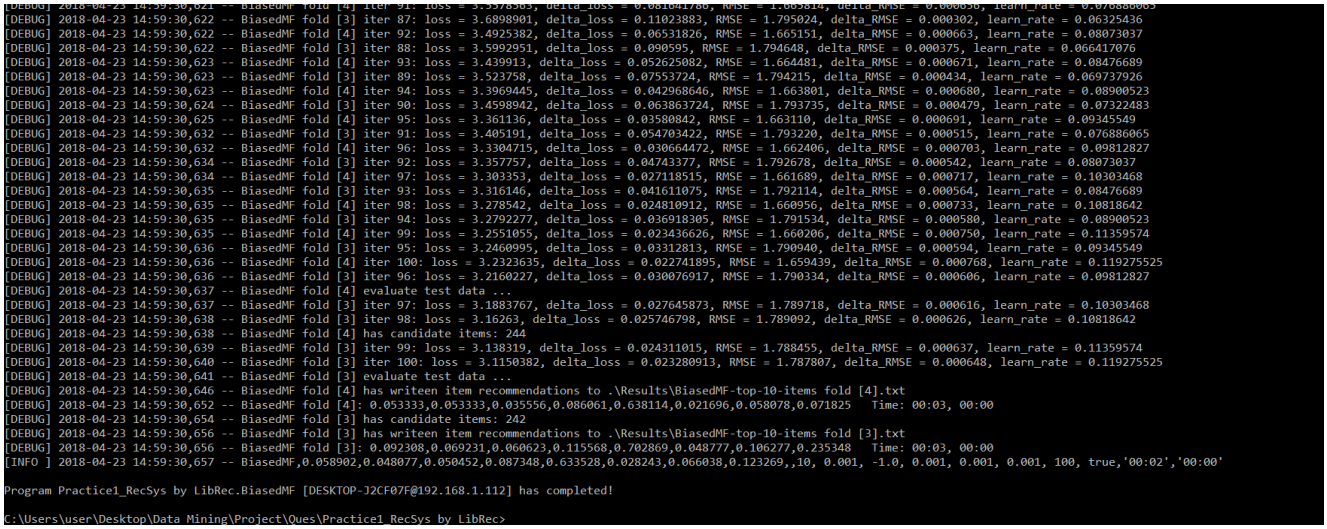


From above we see that num.factor=20 has the lowest value so it is better as compare to other two.

Now we will find Precision@10 at different factors:

- **When factor=10**

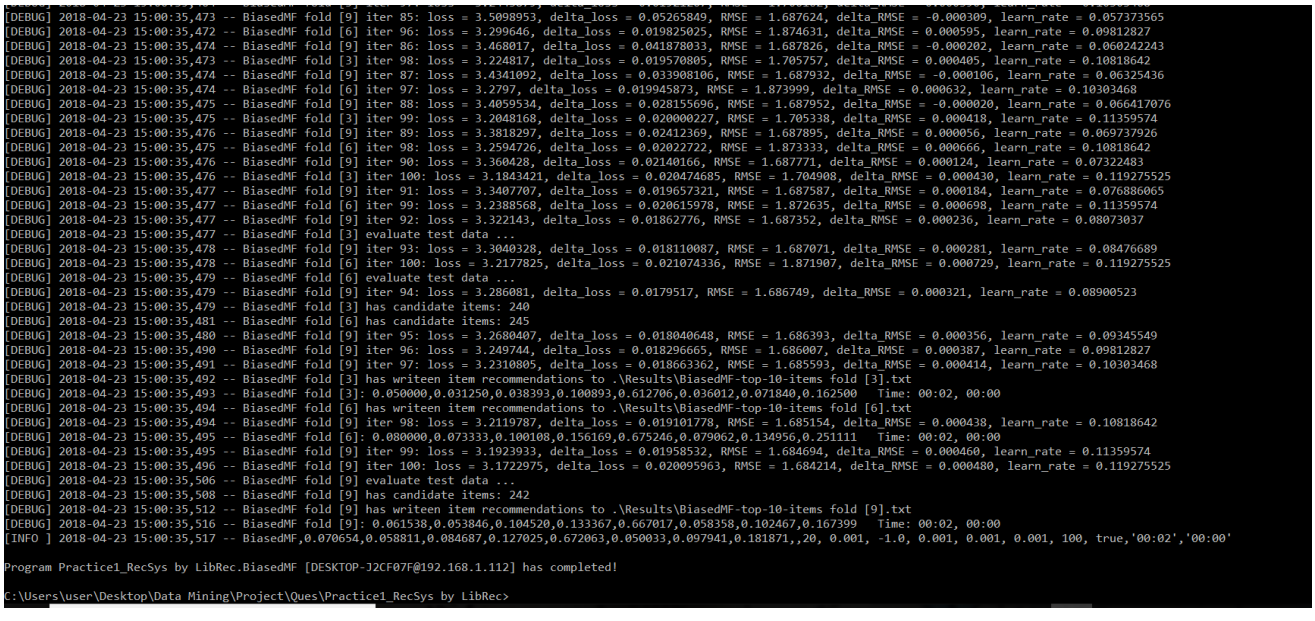
```
# others:  GlobalAvg, UserAvg, ItemAvg, UserCluster, ItemCluster, Random, Constant, MostPop; NMF, SlopeOne, HybridI, PD, AR, PRankD, E
recommender=BiasedMF

# main option: 1. test-set -f test-file-path; 2. cv (cross validation) -k k-folds [-p on, off]
# 3. leave-one-out -t threads -target u, i, r [--by-date]; 4. given-ratio -r ratio -target u, i, r [--by-date]; 5. given-n -N n -targe
# optional: [-v validation-ratio] [--repeat n] [-cpu n] [--rand-seed n] [--test-view all, cold-start, trust-degree 1 5] [--early-stop
evaluation.setup=cv -k 10 --test-view all --early-stop RMSE

# main option: is ranking prediction
item.ranking=on -topN -1 -ignore -1

# main option: is writing out recommendation results; [--fold-data --save-model]
output.setup=on -dir ./Results/ -verbose on, off --to-clipboard --to-file filmtrust-cv.txt

# Guava cache configuration
guava.cache.spec=maximumSize=200,expireAfterAccess=2m

# main option: is email notification enabled
email.setup=off -host smtp.email.com -port 465 -user xxx@email.com -password yyyy -auth true -to xxx@email.com

################################################## Model-based Methods ########################################
num.factors=10
num.max.iter=100

# options: -bold-driver -decay ratio -moment value
```



Here precision@10 0.048077

- **Change factor to 20:**

```
email.setup=off -host smtp.email.com -port 465 -user xxx@email.com -password yyyy -auth true -to xxx@email.com

################################################ Model-based Methods ########################################
num.factors=20
num.max.iter=100

# options: -bold-driver, -decay ratio, -moment value
learn.rate=0.001 -max -1 -bold-driver

reg.lambda=0.1 -u 0.001 -i 0.001 -b 0.001 -s 0.001

# probabilistic graphic models
pgm.setup=-alpha 2 -beta 0.5 -burn-in 300 -sample-lag 10 -interval 100
```



Here precision@10 is 0.058811

- **Change factor to 5**

```
################################################ Model-based Methods ########################################
num.factors=5
num.max.iter=100

# options: -bold-driver, -decay ratio, -moment value
learn.rate=0.001 -max -1 -bold-driver

reg.lambda=0.1 -u 0.001 -i 0.001 -b 0.001 -s 0.001
```

```
[DEBUG] 2018-04-23 15:01:21,292 -- BiasedMF fold [5] iter 98: loss = 11.627868, delta_loss = 0.15265292, RMSE = 2.035275, delta_RMSE = -0.002227, learn_rate = 0.10818642
[DEBUG] 2018-04-23 15:01:21,292 -- BiasedMF fold [7] iter 83: loss = 21.996881, delta_loss = 1.1663178, RMSE = 1.852204, delta_RMSE = -0.000892, learn_rate = 0.052039515
[DEBUG] 2018-04-23 15:01:21,292 -- BiasedMF fold [5] iter 99: loss = 11.501435, delta_loss = 0.1264227, RMSE = 2.038159, delta_RMSE = -0.002883, learn_rate = 0.11359574
[DEBUG] 2018-04-23 15:01:21,292 -- BiasedMF fold [7] iter 84: loss = 20.883703, delta_loss = 1.1131775, RMSE = 1.853513, delta_RMSE = -0.001309, learn_rate = 0.054641493
[DEBUG] 2018-04-23 15:01:21,293 -- BiasedMF fold [5] iter 100: loss = 11.39826, delta_loss = 0.103175506, RMSE = 2.041749, delta_RMSE = -0.003590, learn_rate = 0.119275525
[DEBUG] 2018-04-23 15:01:21,293 -- BiasedMF fold [7] iter 85: loss = 19.817715, delta_loss = 1.0659878, RMSE = 1.855250, delta_RMSE = -0.001737, learn_rate = 0.057373565
[DEBUG] 2018-04-23 15:01:21,293 -- BiasedMF fold [5] evaluate test data ...
[DEBUG] 2018-04-23 15:01:21,293 -- BiasedMF fold [7] iter 86: loss = 18.797022, delta_loss = 1.020693, RMSE = 1.857411, delta_RMSE = -0.002161, learn_rate = 0.060242243
[DEBUG] 2018-04-23 15:01:21,293 -- BiasedMF fold [7] iter 87: loss = 17.822853, delta_loss = 0.9741686, RMSE = 1.859972, delta_RMSE = -0.002560, learn_rate = 0.06325436
[DEBUG] 2018-04-23 15:01:21,297 -- BiasedMF fold [5] has candidate items: 246
[DEBUG] 2018-04-23 15:01:21,298 -- BiasedMF fold [7] iter 88: loss = 16.899084, delta_loss = 0.9237697, RMSE = 1.862885, delta_RMSE = -0.002913, learn_rate = 0.066417076
[DEBUG] 2018-04-23 15:01:21,305 -- BiasedMF fold [7] iter 89: loss = 16.031933, delta_loss = 0.86715126, RMSE = 1.866081, delta_RMSE = -0.003197, learn_rate = 0.069737926
[DEBUG] 2018-04-23 15:01:21,305 -- BiasedMF fold [7] iter 90: loss = 15.229367, delta_loss = 0.8025656, RMSE = 1.869473, delta_RMSE = -0.003391, learn_rate = 0.07322483
[DEBUG] 2018-04-23 15:01:21,305 -- BiasedMF fold [7] iter 91: loss = 14.499884, delta_loss = 0.72948384, RMSE = 1.872804, delta_RMSE = -0.003332, learn_rate = 0.076886065
[DEBUG] 2018-04-23 15:01:21,305 -- BiasedMF fold [7] iter 92: loss = 13.85078, delta_loss = 0.64910436, RMSE = 1.873922, delta_RMSE = -0.001118, learn_rate = 0.08073037
[DEBUG] 2018-04-23 15:01:21,309 -- BiasedMF fold [7] iter 93: loss = 13.286479, delta_loss = 0.5643007, RMSE = 1.874790, delta_RMSE = -0.000868, learn_rate = 0.08476689
[DEBUG] 2018-04-23 15:01:21,313 -- BiasedMF fold [7] iter 94: loss = 12.807611, delta_loss = 0.47886682, RMSE = 1.875331, delta_RMSE = -0.000541, learn_rate = 0.08900523
[DEBUG] 2018-04-23 15:01:21,315 -- BiasedMF fold [5] has writeen item recommendations to .\Results\BiasedMF-top-10-items fold [5].txt
[DEBUG] 2018-04-23 15:01:21,316 -- BiasedMF fold [7] iter 95: loss = 12.411287, delta_loss = 0.39632437, RMSE = 1.875722, delta_RMSE = -0.000391, learn_rate = 0.09345549
[DEBUG] 2018-04-23 15:01:21,316 -- BiasedMF fold [5]: 0.100667,0.080000,0.065611,0.148735,0.644122,0.080036,0.127641,0.214074   Time: 00:03, 00:00
[DEBUG] 2018-04-23 15:01:21,317 -- BiasedMF fold [7] iter 96: loss = 12.092506, delta_loss = 0.3187812, RMSE = 1.876059, delta_RMSE = -0.000337, learn_rate = 0.09812827
[DEBUG] 2018-04-23 15:01:21,317 -- BiasedMF fold [7] iter 97: loss = 11.846268, delta_loss = 0.24623895, RMSE = 1.876398, delta_RMSE = -0.000339, learn_rate = 0.10303468
[DEBUG] 2018-04-23 15:01:21,317 -- BiasedMF fold [7] iter 98: loss = 11.669772, delta_loss = 0.17649476, RMSE = 1.876801, delta_RMSE = -0.000403, learn_rate = 0.10818642
[DEBUG] 2018-04-23 15:01:21,317 -- BiasedMF fold [7] iter 99: loss = 11.564183, delta_loss = 0.10558939, RMSE = 1.877326, delta_RMSE = -0.000525, learn_rate = 0.11359574
[DEBUG] 2018-04-23 15:01:21,317 -- BiasedMF fold [7] iter 100: loss = 11.535479, delta_loss = 0.028704982, RMSE = 1.878023, delta_RMSE = -0.000697, learn_rate = 0.119275525
[DEBUG] 2018-04-23 15:01:21,318 -- BiasedMF fold [7] evaluate test data ...
[DEBUG] 2018-04-23 15:01:21,320 -- BiasedMF fold [7] has candidate items: 243
[DEBUG] 2018-04-23 15:01:21,323 -- BiasedMF fold [7] has writeen item recommendations to .\Results\BiasedMF-top-10-items fold [7].txt
[DEBUG] 2018-04-23 15:01:21,324 -- BiasedMF fold [7]: 0.066667,0.080000,0.050556,0.147566,0.730065,0.027947,0.087119,0.104524   Time: 00:03, 00:00
[INFO ] 2018-04-23 15:01:21,324 -- BiasedMF,0.069696,0.060311,0.044520,0.102690,0.658535,0.041853,0.087278,0.185212,,5, 0.001, -1.0, 0.001, 0.001, 0.001, 100, true,'00:03','00:00'

Program Practice1_RecSys by LibRec.BiasedMF [DESKTOP-J2CF07F@192.168.1.112] has completed!

C:\Users\user\Desktop\Data Mining\Project\Ques\Practice1_RecSys by LibRec>
```

Here Precision@10 is 0.060311

- **Change learn.rate=0.01**

```
################################################ Model-based Methods ########################################
num.factors=5
num.max.iter=100

# options: -bold-driver, -decay ratio, -moment value
learn.rate=0.01 -max -1 -bold-driver

reg.lambda=0.1 -u 0.001 -i 0.001 -b 0.001 -s 0.001

# probabilistic graphic models
pgm.setup=-alpha 2 -beta 0.5 -burn-in 300 -sample-lag 10 -interval 100
```

```
[DEBUG] 2018-04-23 15:02:24,863 -- BiasedMF fold [2] iter 97: loss = 2.4640846, delta_loss = 0.005361885, RMSE = 1.930562, delta_RMSE = 0.000261, learn_rate = 0.11125653
[DEBUG] 2018-04-23 15:02:24,874 -- BiasedMF fold [5] has writeen item recommendations to .\Results\BiasedMF-top-10-items fold [5].txt
[DEBUG] 2018-04-23 15:02:24,872 -- BiasedMF fold [10] iter 89: loss = 2.513944, delta_loss = 0.017838897, RMSE = 1.835303, delta_RMSE = 0.000188, learn_rate = 0.0753028
[DEBUG] 2018-04-23 15:02:24,877 -- BiasedMF fold [10] iter 90: loss = 2.4991431, delta_loss = 0.014800873, RMSE = 1.835263, delta_RMSE = 0.000040, learn_rate = 0.07906794
[DEBUG] 2018-04-23 15:02:24,868 -- BiasedMF fold [8] has candidate items: 246
[DEBUG] 2018-04-23 15:02:24,877 -- BiasedMF fold [10] iter 91: loss = 2.4864156, delta_loss = 0.01272749, RMSE = 1.835339, delta_RMSE = -0.000077, learn_rate = 0.083021335
[DEBUG] 2018-04-23 15:02:24,877 -- BiasedMF fold [10] iter 92: loss = 2.4752333, delta_loss = 0.0111822365, RMSE = 1.835502, delta_RMSE = -0.000163, learn_rate = 0.087172404
[DEBUG] 2018-04-23 15:02:24,877 -- BiasedMF fold [2] iter 98: loss = 2.4592469, delta_loss = 0.004837842, RMSE = 1.930264, delta_RMSE = 0.000297, learn_rate = 0.11681936
[DEBUG] 2018-04-23 15:02:24,877 -- BiasedMF fold [5]: 0.087500,0.062500,0.075253,0.206503,0.697446,0.059117,0.126017,0.207440   Time: 00:00, 00:00
[DEBUG] 2018-04-23 15:02:24,877 -- BiasedMF fold [2] iter 99: loss = 2.4548206, delta_loss = 0.0044262335, RMSE = 1.929929, delta_RMSE = 0.000336, learn_rate = 0.122660324
[DEBUG] 2018-04-23 15:02:24,877 -- BiasedMF fold [10] iter 93: loss = 2.4652815, delta_loss = 0.009951745, RMSE = 1.835722, delta_RMSE = -0.000220, learn_rate = 0.09153102
[DEBUG] 2018-04-23 15:02:24,877 -- BiasedMF fold [3] has candidate items: 246
[DEBUG] 2018-04-23 15:02:24,878 -- BiasedMF fold [10] iter 94: loss = 2.4563463, delta_loss = 0.008935347, RMSE = 1.835973, delta_RMSE = -0.000251, learn_rate = 0.09610757
[DEBUG] 2018-04-23 15:02:24,878 -- BiasedMF fold [2] iter 100: loss = 2.4507554, delta_loss = 0.0040652994, RMSE = 1.929557, delta_RMSE = 0.000372, learn_rate = 0.12879334
[DEBUG] 2018-04-23 15:02:24,880 -- BiasedMF fold [8] has writeen item recommendations to .\Results\BiasedMF-top-10-items fold [8].txt
[DEBUG] 2018-04-23 15:02:24,880 -- BiasedMF fold [2] evaluate test data ...
[DEBUG] 2018-04-23 15:02:24,880 -- BiasedMF fold [10] iter 95: loss = 2.4482563, delta_loss = 0.008090003, RMSE = 1.836229, delta_RMSE = -0.000256, learn_rate = 0.10091295
[DEBUG] 2018-04-23 15:02:24,880 -- BiasedMF fold [8]: 0.100000,0.083333,0.107085,0.210246,0.677440,0.117231,0.162521,0.216766   Time: 00:00, 00:00
[DEBUG] 2018-04-23 15:02:24,880 -- BiasedMF fold [10] iter 96: loss = 2.4408574, delta_loss = 0.007398931, RMSE = 1.836470, delta_RMSE = -0.000241, learn_rate = 0.1059586
[DEBUG] 2018-04-23 15:02:24,882 -- BiasedMF fold [10] iter 97: loss = 2.434004, delta_loss = 0.00685311, RMSE = 1.836679, delta_RMSE = -0.000209, learn_rate = 0.11125653
[DEBUG] 2018-04-23 15:02:24,882 -- BiasedMF fold [2] has candidate items: 242
[DEBUG] 2018-04-23 15:02:24,882 -- BiasedMF fold [10] iter 98: loss = 2.427563, delta_loss = 0.0064413515, RMSE = 1.836845, delta_RMSE = -0.000166, learn_rate = 0.11681936
[DEBUG] 2018-04-23 15:02:24,884 -- BiasedMF fold [10] iter 99: loss = 2.4214163, delta_loss = 0.0061464356, RMSE = 1.836960, delta_RMSE = -0.000115, learn_rate = 0.122660324
[DEBUG] 2018-04-23 15:02:24,884 -- BiasedMF fold [10] iter 100: loss = 2.415471, delta_loss = 0.0059453575, RMSE = 1.837022, delta_RMSE = -0.000062, learn_rate = 0.12879334
[DEBUG] 2018-04-23 15:02:24,889 -- BiasedMF fold [10] evaluate test data ...
[DEBUG] 2018-04-23 15:02:24,891 -- BiasedMF fold [3] has writeen item recommendations to .\Results\BiasedMF-top-10-items fold [3].txt
[DEBUG] 2018-04-23 15:02:24,891 -- BiasedMF fold [2] has writeen item recommendations to .\Results\BiasedMF-top-10-items fold [2].txt
[DEBUG] 2018-04-23 15:02:24,892 -- BiasedMF fold [3]: 0.082353,0.058824,0.113841,0.153057,0.629343,0.089017,0.123204,0.158824   Time: 00:00, 00:00
[DEBUG] 2018-04-23 15:02:24,892 -- BiasedMF fold [2]: 0.080000,0.060000,0.040135,0.118923,0.640819,0.040249,0.081402,0.130741   Time: 00:00, 00:00
[DEBUG] 2018-04-23 15:02:24,894 -- BiasedMF fold [10] has candidate items: 246
[DEBUG] 2018-04-23 15:02:24,898 -- BiasedMF fold [10] has writeen item recommendations to .\Results\BiasedMF-top-10-items fold [10].txt
[DEBUG] 2018-04-23 15:02:24,898 -- BiasedMF fold [10]: 0.061538,0.053846,0.042308,0.066026,0.632485,0.046561,0.080980,0.170940   Time: 00:00, 00:00
[INFO ] 2018-04-23 15:02:24,898 -- BiasedMF,0.098892,0.071300,0.079290,0.134295,0.665588,0.062552,0.112766,0.201997,,5, 0.01, -1.0, 0.001, 0.001, 0.001, 100, true,'00:00','00:00'

Program Practice1_RecSys by LibRec.BiasedMF [DESKTOP-J2CF07F@192.168.1.112] has completed!

C:\Users\user\Desktop\Data Mining\Project\Ques\Practice1_RecSys by LibRec>
```

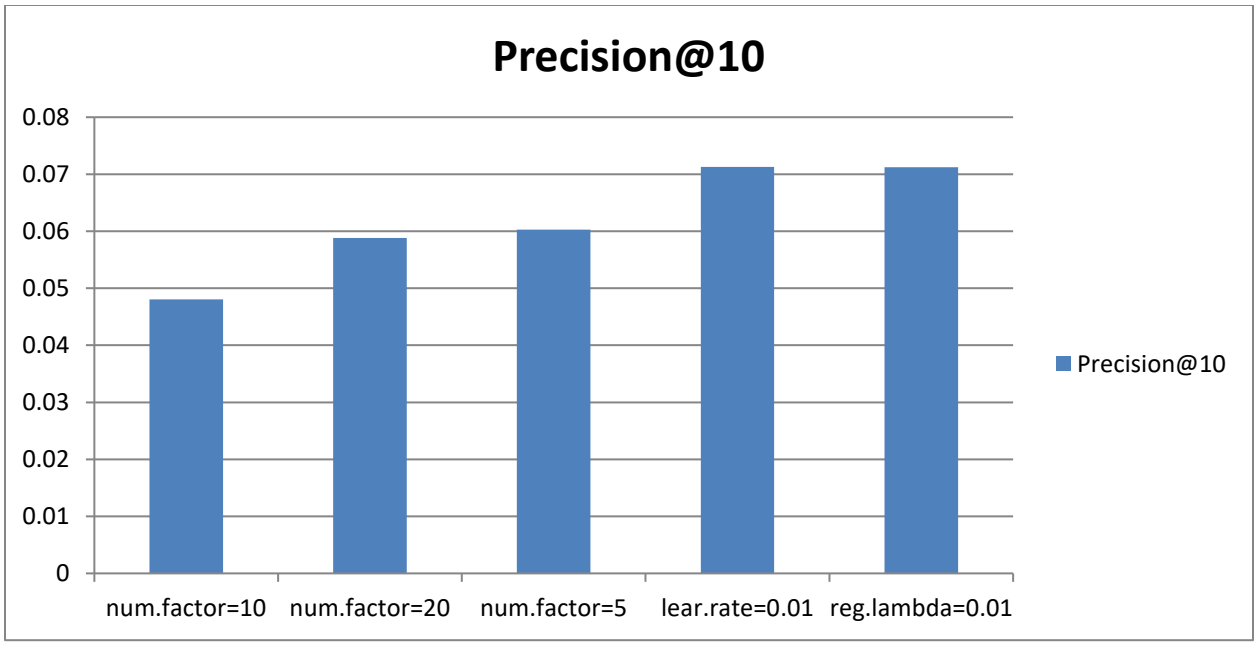
Here precision@10 is 0.071300

- **Change reg.lambda to 0.01**

```
35 ################################################## Model-based Methods ##########################################
36 num.factors=5
37 num.max.iter=100
38
39 # options: -bold-driver, -decay ratio, -moment value
40 learn.rate=0.001 -max -1 -bold-driver
41
42 reg.lambda=0.01 -u 0.001 -i 0.001 -b 0.001 -s 0.001
43
44 # probabilistic graphic models
45 pgm.setup=-alpha 2 -beta 0.5 -burn-in 300 -sample-lag 10 -interval 100
46
```

```
[DEBUG] 2018-04-23 15:03:49,088 -- BiasedMF fold [10] iter 100: loss = 13.881585, delta_loss = 0.4104674, RMSE = 1.724990, delta_RMSE = 0.000248, learn_rate = 0.119275525
[DEBUG] 2018-04-23 15:03:49,088 -- BiasedMF fold [10] evaluate test data ...
[DEBUG] 2018-04-23 15:03:49,088 -- BiasedMF fold [4] iter 87: loss = 16.673773, delta_loss = 1.1034253, RMSE = 1.834143, delta_RMSE = -0.001803, learn_rate = 0.06325436
[DEBUG] 2018-04-23 15:03:49,088 -- BiasedMF fold [4] iter 88: loss = 15.679947, delta_loss = 0.99382675, RMSE = 1.835928, delta_RMSE = -0.001785, learn_rate = 0.066417076
[DEBUG] 2018-04-23 15:03:49,088 -- BiasedMF fold [3] has writeen item recommendations to .\Results\BiasedMF-top-10-items fold [3].txt
[DEBUG] 2018-04-23 15:03:49,089 -- BiasedMF fold [1] evaluate test data ...
[DEBUG] 2018-04-23 15:03:49,089 -- BiasedMF fold [3]: 0.047059,0.029412,0.038839,0.068250,0.605550,0.028708,0.059479,0.143137   Time: 00:00, 00:00
[DEBUG] 2018-04-23 15:03:49,089 -- BiasedMF fold [10] has candidate items: 242
[DEBUG] 2018-04-23 15:03:49,091 -- BiasedMF fold [1] has candidate items: 237
[DEBUG] 2018-04-23 15:03:49,096 -- BiasedMF fold [10] has writeen item recommendations to .\Results\BiasedMF-top-10-items fold [10].txt
[DEBUG] 2018-04-23 15:03:49,094 -- BiasedMF fold [4] iter 89: loss = 14.788971, delta_loss = 0.8909762, RMSE = 1.837349, delta_RMSE = -0.001421, learn_rate = 0.069737926
[DEBUG] 2018-04-23 15:03:49,100 -- BiasedMF fold [10]: 0.100000,0.071429,0.053855,0.144671,0.688304,0.049518,0.108887,0.220238   Time: 00:00, 00:00
[DEBUG] 2018-04-23 15:03:49,101 -- BiasedMF fold [9] has writeen item recommendations to .\Results\BiasedMF-top-10-items fold [9].txt
[DEBUG] 2018-04-23 15:03:49,100 -- BiasedMF fold [4] iter 90: loss = 13.993886, delta_loss = 0.7950852, RMSE = 1.837992, delta_RMSE = -0.000643, learn_rate = 0.07322483
[DEBUG] 2018-04-23 15:03:49,101 -- BiasedMF fold [9]: 0.076923,0.053846,0.057448,0.076129,0.671043,0.034371,0.081738,0.217949   Time: 00:00, 00:00
[DEBUG] 2018-04-23 15:03:49,102 -- BiasedMF fold [4] iter 91: loss = 13.286862, delta_loss = 0.7070231, RMSE = 1.838472, delta_RMSE = -0.000480, learn_rate = 0.076886065
[DEBUG] 2018-04-23 15:03:49,102 -- BiasedMF fold [4] iter 92: loss = 12.659047, delta_loss = 0.6278153, RMSE = 1.838600, delta_RMSE = -0.000128, learn_rate = 0.08073037
[DEBUG] 2018-04-23 15:03:49,102 -- BiasedMF fold [4] iter 93: loss = 12.100834, delta_loss = 0.55821306, RMSE = 1.838844, delta_RMSE = -0.000244, learn_rate = 0.08476689
[DEBUG] 2018-04-23 15:03:49,102 -- BiasedMF fold [4] iter 94: loss = 11.602467, delta_loss = 0.49836785, RMSE = 1.839241, delta_RMSE = -0.000397, learn_rate = 0.08900523
[DEBUG] 2018-04-23 15:03:49,102 -- BiasedMF fold [4] iter 95: loss = 11.154875, delta_loss = 0.44759145, RMSE = 1.839813, delta_RMSE = -0.000572, learn_rate = 0.09345549
[DEBUG] 2018-04-23 15:03:49,102 -- BiasedMF fold [4] iter 96: loss = 10.750721, delta_loss = 0.40415362, RMSE = 1.840600, delta_RMSE = -0.000786, learn_rate = 0.09812827
[DEBUG] 2018-04-23 15:03:49,103 -- BiasedMF fold [1] has writeen item recommendations to .\Results\BiasedMF-top-10-items fold [1].txt
[DEBUG] 2018-04-23 15:03:49,104 -- BiasedMF fold [4] iter 97: loss = 10.385622, delta_loss = 0.36509886, RMSE = 1.841634, delta_RMSE = -0.001035, learn_rate = 0.10303468
[DEBUG] 2018-04-23 15:03:49,104 -- BiasedMF fold [1]: 0.028571,0.050000,0.023214,0.141071,0.598770,0.036480,0.076472,0.071429   Time: 00:00, 00:00
[DEBUG] 2018-04-23 15:03:49,104 -- BiasedMF fold [4] iter 98: loss = 10.05948, delta_loss = 0.3261427, RMSE = 1.842943, delta_RMSE = -0.001309, learn_rate = 0.10818642
[DEBUG] 2018-04-23 15:03:49,104 -- BiasedMF fold [4] iter 99: loss = 9.77767, delta_loss = 0.2818093, RMSE = 1.844534, delta_RMSE = -0.001591, learn_rate = 0.11359574
[DEBUG] 2018-04-23 15:03:49,104 -- BiasedMF fold [4] iter 100: loss = 9.551677, delta_loss = 0.22599383, RMSE = 1.846390, delta_RMSE = -0.001855, learn_rate = 0.119275525
[DEBUG] 2018-04-23 15:03:49,104 -- BiasedMF fold [4] evaluate test data ...
[DEBUG] 2018-04-23 15:03:49,105 -- BiasedMF fold [4] has candidate items: 237
[DEBUG] 2018-04-23 15:03:49,110 -- BiasedMF fold [4] has writeen item recommendations to .\Results\BiasedMF-top-10-items fold [4].txt
[DEBUG] 2018-04-23 15:03:49,114 -- BiasedMF fold [4]: 0.093333,0.080000,0.052862,0.097811,0.644294,0.048586,0.102258,0.227778   Time: 00:00, 00:00
[INFO ] 2018-04-23 15:03:49,115 -- BiasedMF,0.096691,0.071225,0.064157,0.118979,0.664702,0.048321,0.101335,0.199522,,5, 0.001, -1.0, 0.001, 0.001, 0.001, 100, true,'00:00','00:00'

Program Practice1_RecSys by LibRec.BiasedMF [DESKTOP-J2CF07F@192.168.1.112] has completed!
```

Here precision@10 0.071225

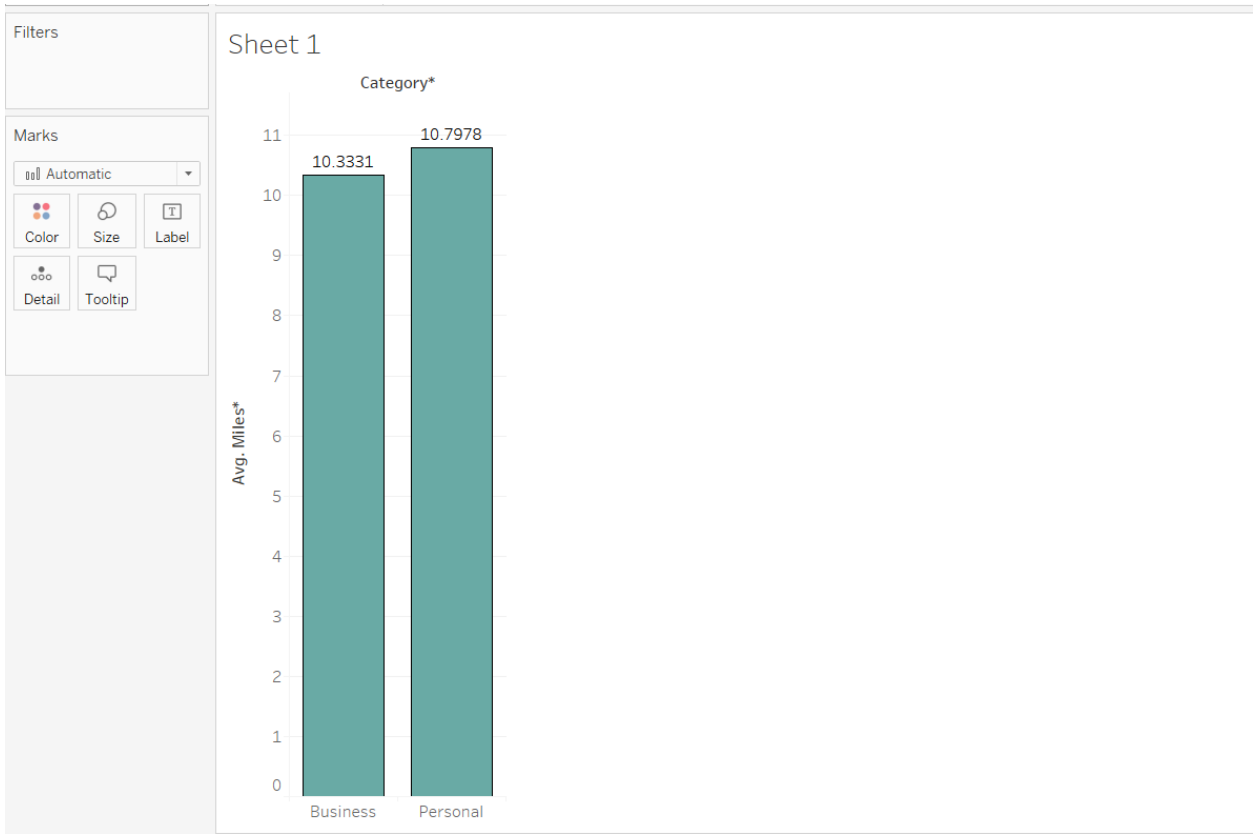


We will now compare all values and see that its highest for reg.lambda=0.01

- Comparing MAE value result of both BMF and USERKNN algorithms, we see that MAE value is MAE value for Biased Matrix Factorization is lowest for **num.factor=20** and value is **1.388591** whereas MAE value for USERKNN is lowest for PCC at K=15 and value is **1.530779. On comparing these two we see that Biased Matrix Factorization has lowest value so it is considered to be better than USERKNN.**
- Comparing Precision@10 value result of both BMF and USERKNN algorithms, we see that Precision@10for BMF is highest for **reg.lambda=0.01** and values is **0.071225** whereas Precision@10 for USERKNN for PCC similarity is highest for PCC at k=10 and value is **0.054631. On comparing these two we see that Biased Matrix Factorization has highest value so it is considered to be better than USERKNN.**

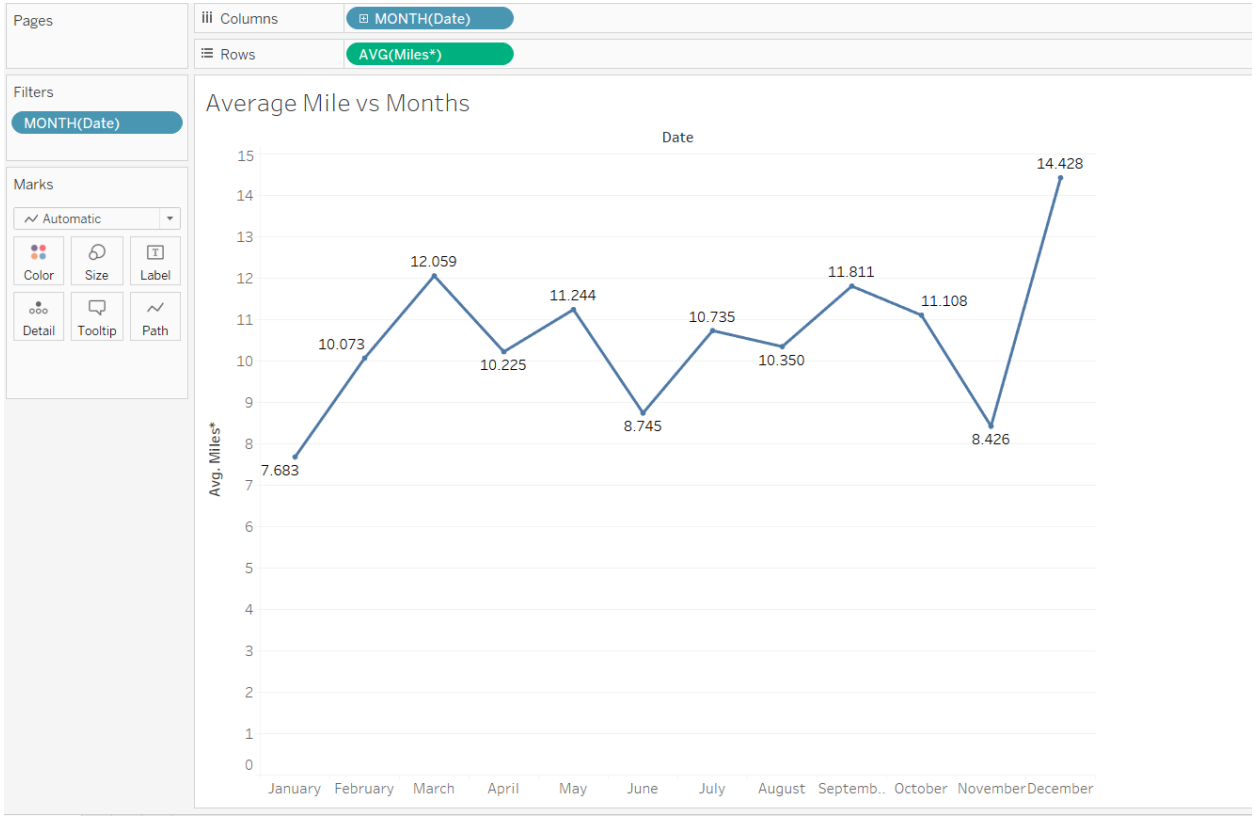
Solution 10) {Visualization}

Average Miles vs Category:



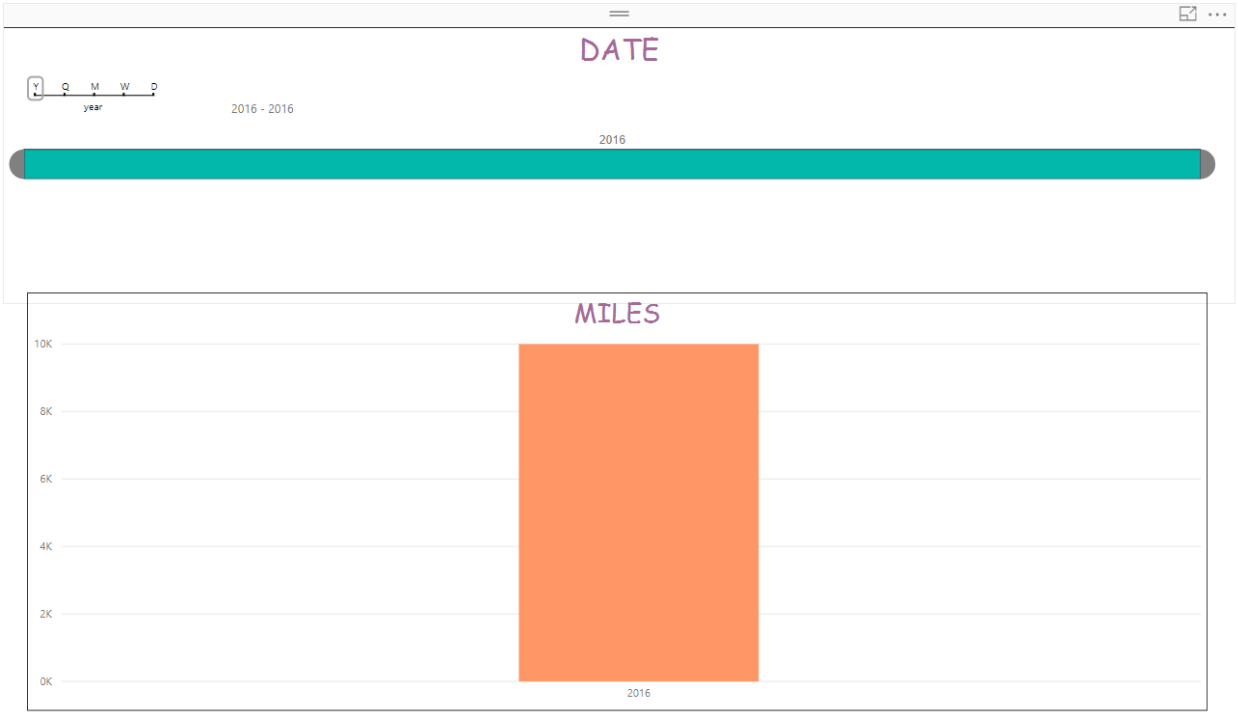
We can observe that personal category customers have the highest average miles traveled.

Analyze most number of rides taken in a month.

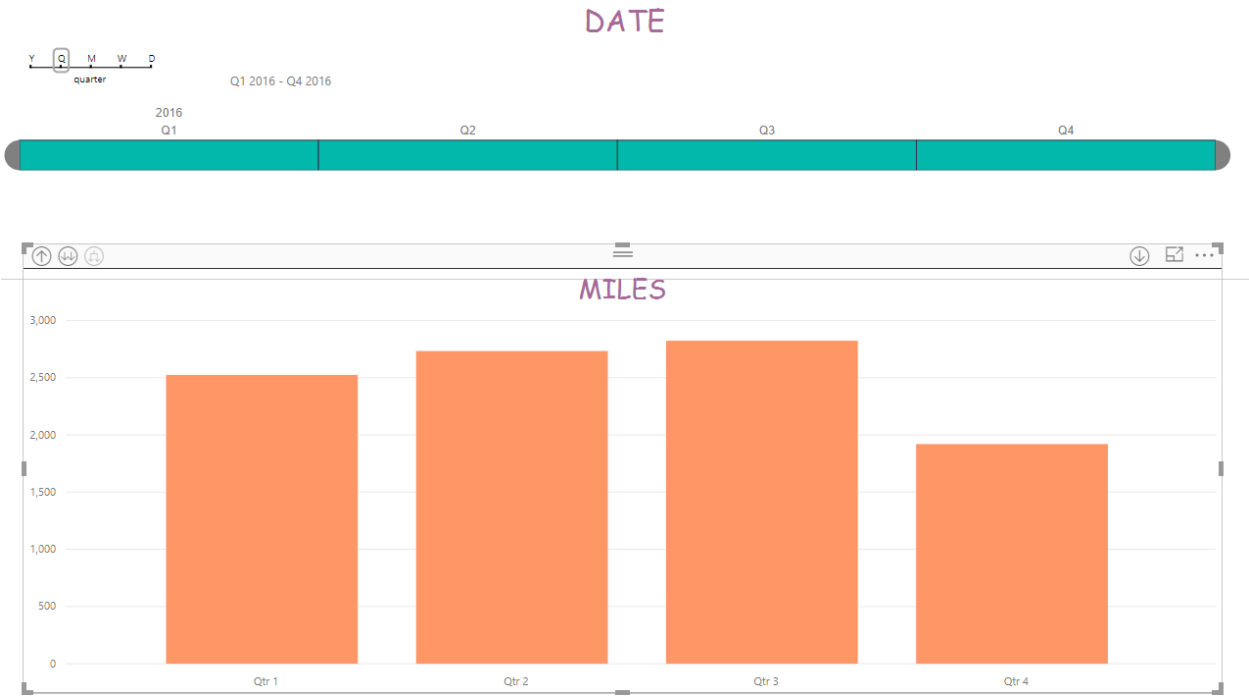


We can observe here that the average miles traveled was maximum in the month of December and it was least in the month of January. It can be possible that December is the month of festival due to which the miles traveled is highest.

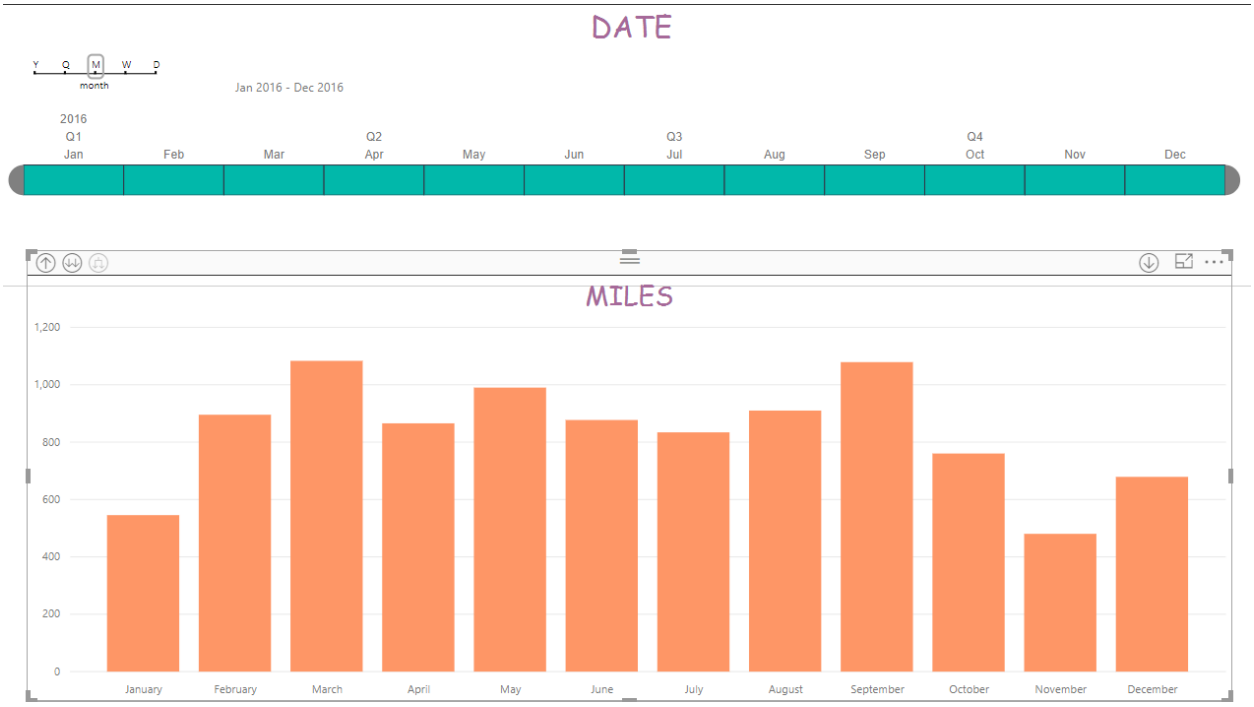
Date VS Miles:



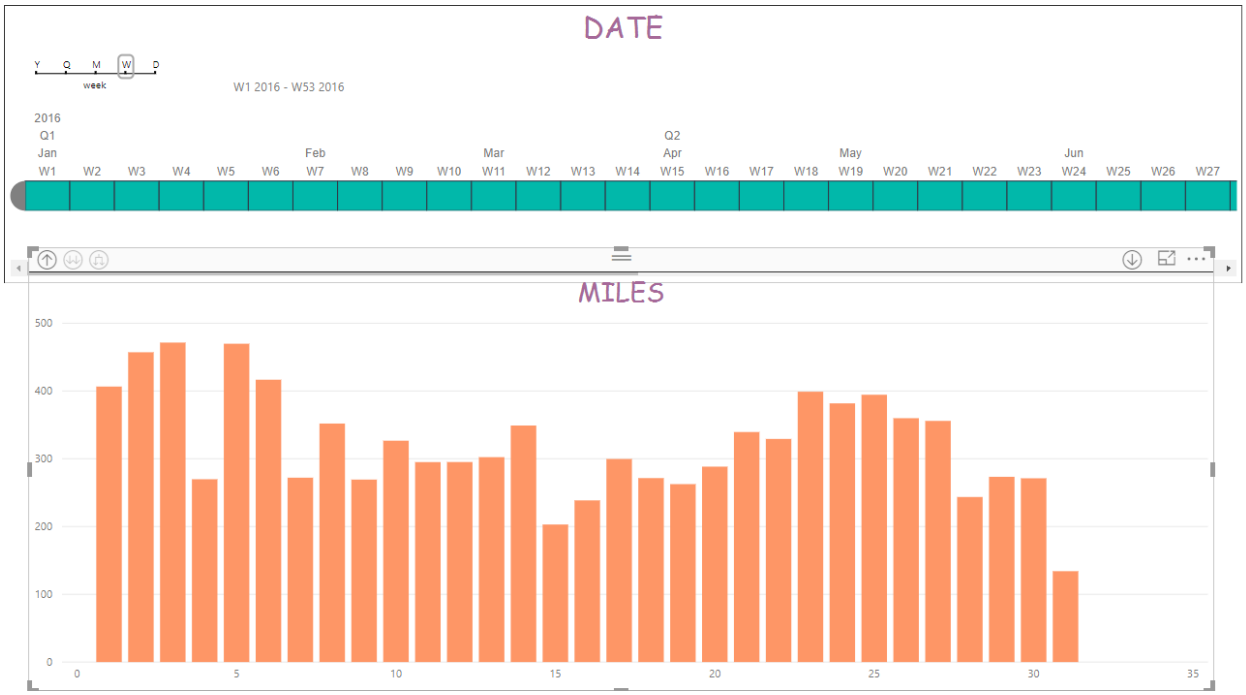
Quarter-Miles (Q1,Q2,Q3,Q4)



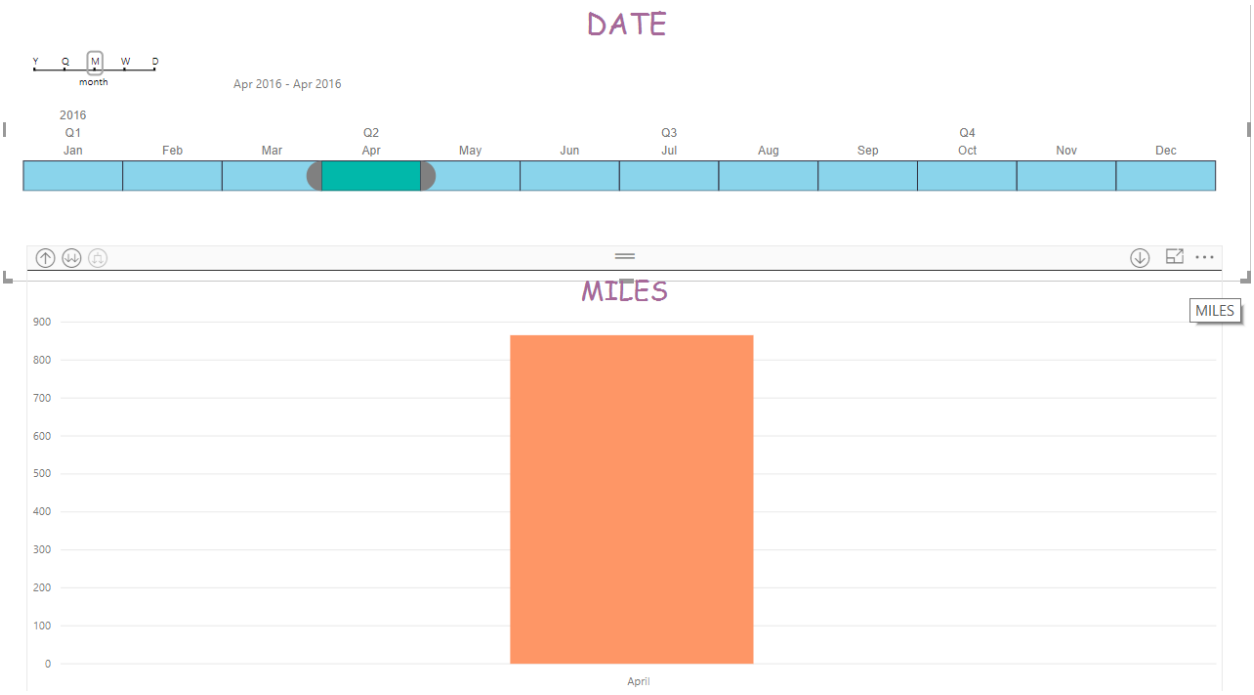
Months vs Miles:



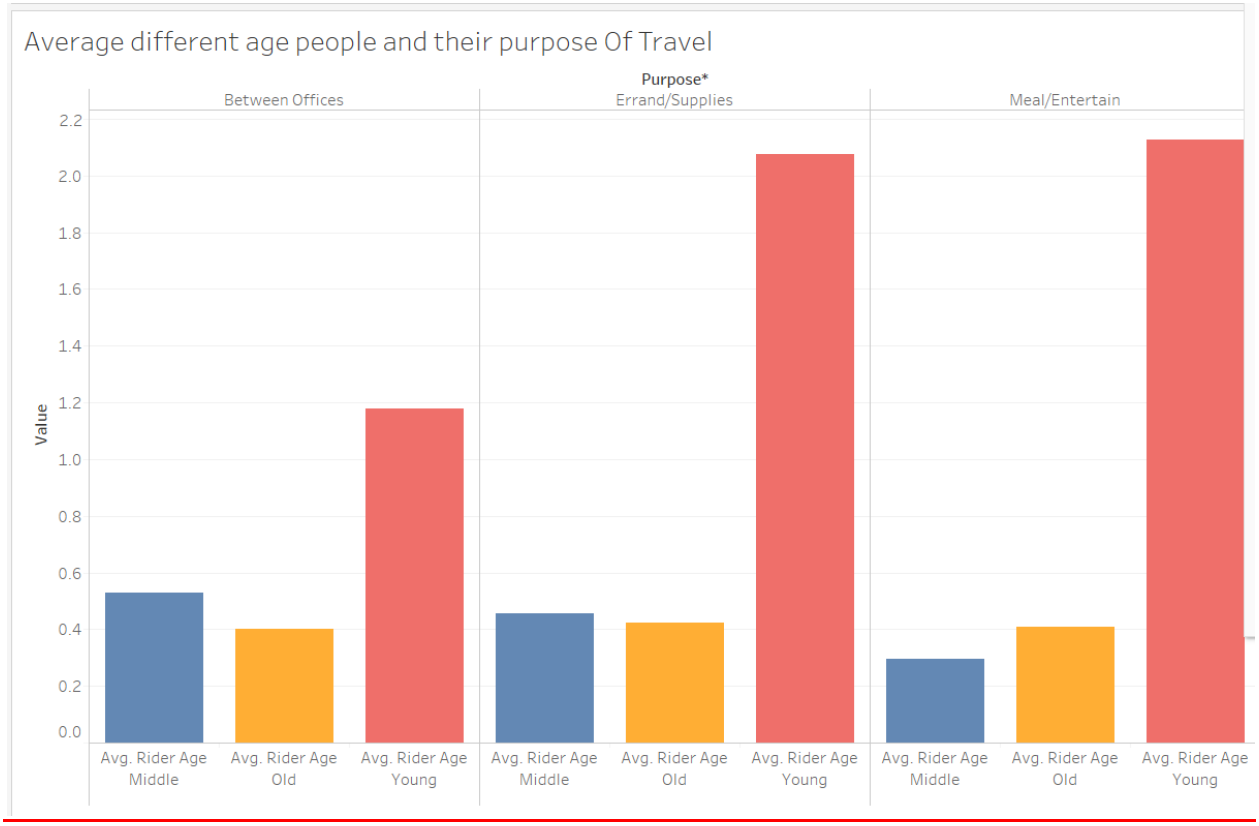
Weeks vs Miles:



Miles traveled in the month of April 2016:



People Age vs Purpose of travel:



It depicts that young people prefer more cabs for all 3 purposes, but for office purpose and errand/Supplies, middle age people book more cabs as compared to old age people.

Ride Duration vs Month:

We can observe above that the maximum ride duration was in the month of January and the minimum ride duration was in the month of October.

Most number of cabs delayed vs area:

We can observe that the most number cabs delayed were in the North area and least delay was in the West area.

6. Evaluations and Results

6.1. Evaluation Methods

As we have huge data set we split our data using hold out Evaluation method. We followed the standard learning process for splitting our data into training and testing data where we learned model using training data, validated the data using test data to access accuracy, and at the end applied the selected model to unseen data. We have split the data into 75% training the data and 25 % in testing data by using following command in python,

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.25, random_state=0)

6.2 Results and Findings

1) We conclude that at 95% confidence level, there is sufficient evidence that average miles travelled is not equal to 10.4 miles.

2)We conclude that at 95% confidence level, there is sufficient evidence that average fare for both business users and personal users is same.

3) We have predicted the Age of customer traveling by Uber considering factors like Gender, Purpose of travel and Category of travel and predicted the Age of customer based on these features using KNN, Naïve Bayes, SVM, Decision Trees classification algorithms. Also, we found out that highest accuracy was given by decision tree algorithm.

4) We conclude that that using feature selection features like date, category, stop, status-delay and status-on time are features that are important to estimate cab fare amongst all other.

5) **a)** Females prefer Uber maximum for using it between offices.

b) Males prefer Uber maximum for using it In Errand/Supplies.

c) Females prefer Uber maximum for using It for Meal/Entertainment.

6) We conclude that we will do smoothing by bin means method on fare attribute. We will divide the date in the bin of 4 and then replace by the mean value and also depth of bin is 9 which means there are 9 values in each bin.

7) We found that AIC value is less for backward model compared to forward model, hence model having lowest AIC is better model, in our case backward model is better than forward model.

8) Using K-Means clustering and hierarchical clustering we analyzed the relation between Age and Gender. We found in some cases that males between the age group 10-30 preferred cabs most.

9) Using Recommender systems and on comparing two algorithms i.e. USERKNN and Biased Matrix factorization, we see that Biased Matrix Factorization has lowest value, so it is considered to be better than USERKNN and we will use BMF to recommend cabs on basis of driver's rating.

10) Using Tableau and Power BI visualizations we have visualized different cases which can be useful for Uber company to look and identify different cases by which they can improve their profit.

7. Conclusions and Future Work

7.1 CONCLUSIONS:

In this fast-paced world, analyzing the past data and predicting the future outcomes has become essential for businesses to perform better. Similarly, our project should perform analysis on the past data and provided the results. We tried to solve every research problem and performed analysis on it. We implemented classification and clustering algorithms and found out accuracy for classification algorithms and identified the algorithm which gives the best accuracy. We also implemented Recommender systems using Librec and identified which algorithm is better.

7.2 LIMITATIONS:

There could be many limitations in data set which we found online. There is no proof that the data sample we are working on is true. There could be wrong sample data could have led to wrong conclusions and hypothesis testing.

7.3 Potential inprovements or Future Work:

In future, we would try to get more accurate data. We shall try to connect with actual people who would have worked with Uber and were taking care of all real data so that we work on more accurate and refined data set. Also we would like to build a report using Power BI or Tableau Report. Apart from this, we would also like to implement more classification and clustering algorithms.