Munching Num-Num

Anmol Jaising	Jeet Thakur	Karan Makhija	Rishab Lalwani	Soumil Sarvaiya	Suchit Nandal
akj9051	jbt9370	km5337	rl9703	ss8165	sn1566

Introduction:

Munching Num-Num is a big data-oriented project which concentrates on the analysis of the data provided by the Yelp repository. The data from the Yelp users are extracted and eventually mined that defines patterns and trends thus enabling restaurants to solve a plethora of their problems.

In this report, we will cover what all has been done till now. It mentions the various technique and languages used to get to this point. It describes what changes have been covered. Apart from this, it also describes the data mining component in detail.

Data Set Selection:

We have chosen the Yelp Data Set for our data mining task.

Yelp helps users to search through millions of other user reviews to find the perfect place to dine. This dataset, being mammoth in size, gives us a wealth of data to manipulate and play with. More importantly, it has provided us a medium through which our concepts of Data Management and Data Mining has increased four-folds.

It provides a broader picture of the data management components such as data scrapping and data normalization as well as increases our knowledge about the various data mining concepts and techniques.

The Yelp dataset helps us to analyze the various problems faced while dealing with such sheer volumes of data.

Data Scrapping

Data Scrapping is used to import data from one form to another.

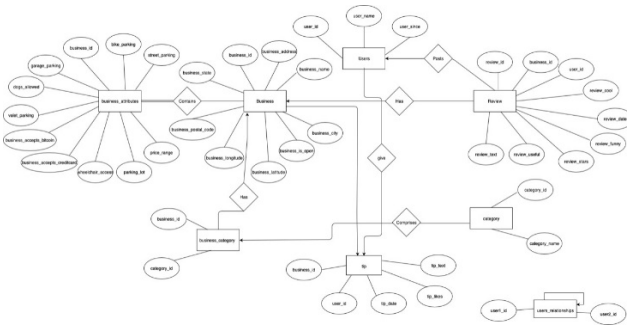
The dataset obtained from the Yelp repository being in the .JSON makes it not viable to be queried. Thus, we have created python scripts that extracts the data from these .JSON file and exports it into MySQL, thus eliminating the manual need of table creation.

Data Cleaning:

Data Cleaning is the process of the detecting and correcting the anomalies or incorrect data from a record set. By doing so, it improves data quality.

In our dataset, we have cleaned the data by removing null values, eliminating data duplicates, etc.

We graphically illustrate the entities and their relationship. This is done with the help of an Entity – Relationship (ER) Diagram.



--This Image has been attached as a separate file

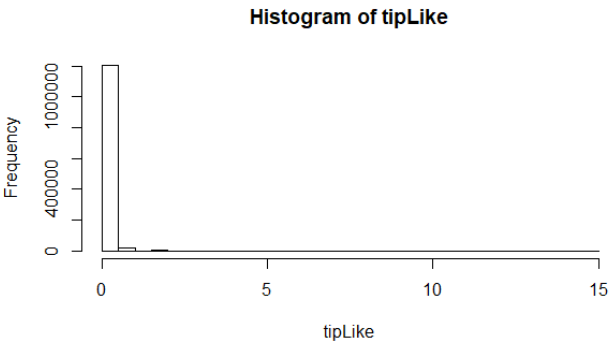
Data Exploration:

Data Exploration forms the base step for further data analytics. We explore our large data set to uncover initial patterns, characteristics and points of interest. Data exploration provides us the data which can be used for trends spotting, pattern spotting and can also be used to visualize the data.

We can perform data exploration by performing either Univariate or Bivariate analysis.

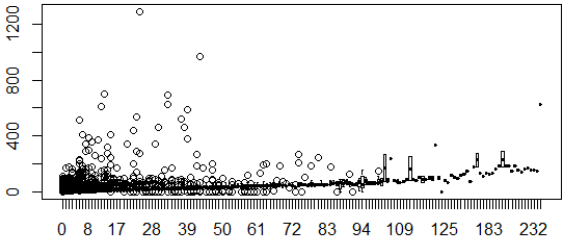
Univariate Analysis explores variables (attributes) one by one. This can be categorized either numerically or categorically.

For example, in our dataset we are analyzing what kind of tips we can have. We use the single attribute value 'tip_like' from the table 'tip'



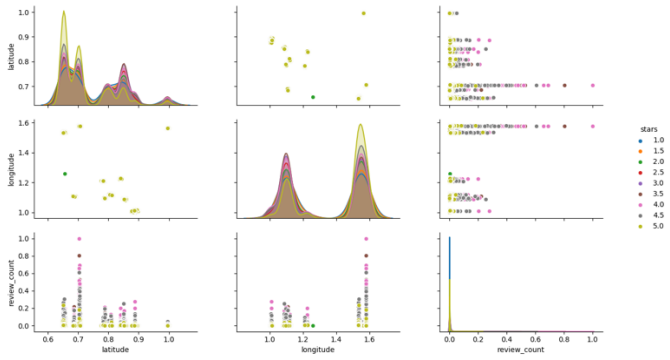
Bivariate Analysis explores variables two at a time. This helps us understand pairwise relationships between variables.

For example, in our dataset, we show the relationship between Review Funny and Review Cool from the Review table.



Data Visualization:

To show data visualization, we give provide the correlation on the business tables:



We show the correlation between different attributes such as review_count, latitude and longitude in order to show the business stars.

We provide scatter plots for those having different attributes.

Data Mining:

For the data mining part, we are using K- Nearest Neighbor (kNN) as the classifier.

kNN is an algorithm which is used to store all the available cases. It uses these cases to classify any new case based on a similar feature.

The value of k will decide how well the data will be utilized in determining the efficiency of the model.

A) kNN for predicting stars received

In our dataset, we choose 3 values of k (100, 10 and 5) to predict the stars a restaurant receives based on the review count, checking count and city it is located in.

We split our data in 75:25 ratio to form the training and testing data. We train our data using to create a model that consists of x_train (review count, checking count and stars), and y_train (stars)

On the testing data which is the remaining 25% data, we create a model consisting a x_test and y_test.

```
Split = sample.split(Data,SplitRatio = 0.75)
SplitIndex = createDataPartition(Data$review_count,p=.7,list=FALSE,times=1)
training_set = Data[SplitIndex,]
testing_set = Data[-SplitIndex,]
X_train = training_set[,c("review_count","checkin_count","city")]
Y_train = training_set[,c("stars")]
X_test =  testing_set[,c("review_count","checkin_count","city")]
Y_test =  testing_set[,c("stars")]
```

Using the nearest 100 (k=100) neighbors we create a model using x_train, x_test and y_train and give this to a confusion matrix table.

```
prc_test_pred <- knn(train = X_train, test = X_test,cl = Y_train, k=100)
ConfuseMatrix = table(Y_test,prc_test_pred)
ConfuseMatrix

##       prc_test_pred
## Y_test    1    2    3    4    5
##      1    0    5    0   85    0
##      2    0  127    5 2368    0
##      3    0   80    9 2656    0
##      4    0  149    8 8991    0
##      5    0    9    1  326    0
```

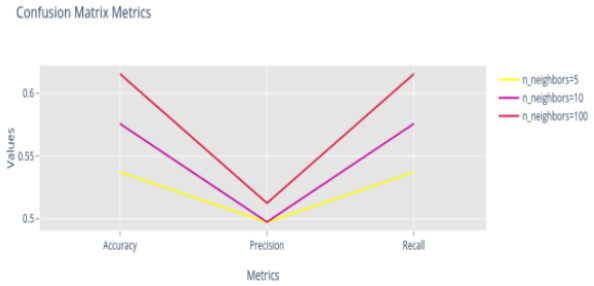
Now we use k as 10, we get

```
prc_test_pred1 <- knn(train = X_train, test = X_test,cl = Y_train, k=10)
ConfuseMatrix1 = table(Y_test,prc_test_pred1)
ConfuseMatrix1

##       prc_test_pred1
## Y_test    1    2    3    4    5
##      1    0   17    7   66    0
##      2    0  430  205 1862    3
##      3    0  268  209 2265    3
##      4    0  593  433 8119    3
##      5    0   37   25  274    0
```

Using k = 5, we get:

```
prc_test_pred2 <- knn(train = X_train, test = X_test,cl = Y_train, k=5)
ConfuseMatrix2 = table(Y_test,prc_test_pred2)
ConfuseMatrix2

##       prc_test_pred2
## Y_test    1    2    3    4    5
##      1    0   25   17   45    3
##      2    1  537  296 1650   16
##      3    1  374  325 2036    9
##      4    4  804  735 7566   39
##      5    0   50   36  246    4
```

Putting these 3 on a graph, we get:



From this we can see for the nearest 100 neighbors, we get the most accurate prediction.

B) Decision Tree for predicting price range

We also can analyze the business attribute data by creating a Decision Tree.

In this model, we analyze and mine the data to predict the price range when particular attributes (business accepts bitcoins, bike parking, garage parking, street parking, dogs allowed, wheelchair access, business accepts credit card, parking_lot and valet_parking) are present.

Like before, we create a training data on 75% of the data and use the remaining part of the data for testing.

Initially, we train our data by providing the business accepts bitcoins, bike parking, garage parking, street parking, dogs allowed, wheelchair access, business accepts credit card, parking_lot and valet_parking, with the deciding attribute to be prince_range.

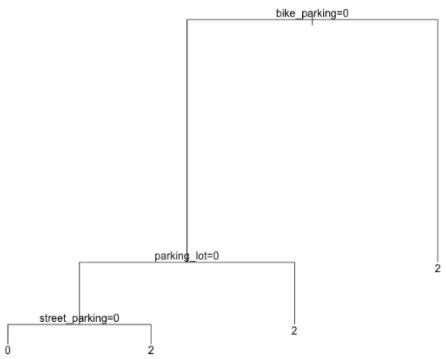
```
library("rpart")
ba <- read.csv("business_attributes.csv", sep = ";")
cols <- c('business_accepts_bitcoin', 'street_parking','bike_parkir
ba[cols] <- lapply(ba[cols], as.factor)

set.seed(1)
train <- sample(1:nrow(ba), 0.75 * nrow(ba))
target = price_range ~ business_accepts_bitcoin + street_parking +
baTree <- rpart(target, data = ba[train, ], method = 'class')
plot(baTree)
```

Now, we apply the model we have just learned and give it to the test data which is the remaining data.

```
text(baTree,pretty=0)
baPred <- predict(baTree, ba[-train, ], type = 'class')
x=table(baPred, ba[-train, ]$price_range)
x=sum(diag(x))*100/sum(x)

x
```

After plotting, we get a decission tree that predicts what should be the price range based on the given attributes.

This becomes beneficial when we have constraints on the our attributes.



C) kNN for predicting stars

In our dataset, we use a 3 values of k (25,50,75) to predict the stars a restaurant receives based on the attributes review_cool, review_funny and review_useful.

```
review = read.csv(file = "D:\\CSV\\review.csv", sep = ";",nrows = 10000)
data_var <- review[,
c("review_stars","review_cool","review_funny","review_useful")]
table(data_var$review_stars)

##
##    1    2    3    4    5
## 1427  789 1137 2269 4378

round(prop.table(table(data_var$review_stars))*100, digits = 1)

##
##    1    2    3    4    5
## 14.3  7.9 11.4 22.7 43.8
```

We then normalize our data so that the value is in the range 0 to 1.

```
normalize <- function(x){
  return ((x - min(x))/(max(x)- min(x)))
}
data_new <- as.data.frame(lapply(data_var[2:4], normalize))

summary(data_new)

##   review_cool       review_funny      review_useful
##  Min.   :0.00000   Min.   :0.000000   Min.   :0.000000
##  1st Qu.:0.00000   1st Qu.:0.000000   1st Qu.:0.000000
##  Median :0.00000   Median :0.000000   Median :0.000000
##  Mean   :0.00513   Mean   :0.005917   Mean   :0.012824
##  3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu.:0.009346
##  Max.   :1.00000   Max.   :1.000000   Max.   :1.000000
```

We split our data to 60:40 ratio to form a training and testing data.

When we give the value of k as 25, we get the following confusion matrix:

```
data_train_label, k = 25)

confuse_matrix = table(data_test_label, prc_test_pred)
confuse_matrix

##                prc_test_pred
## data_test_label   1   2   3   4   5
##               1   0   0   1   1  51
##               2   0   0   0   3  36
##               3   1   0   4   1  41
##               4   0   0   2   3  68
##               5   2   0   3   8 175
```

When we give the value of k as 50, we get the following confusion matrix:

```
prc_test_pred <- knn(train =data_train, test=data_test, cl =
data_train_label, k = 50)

confuse_matrix = table(data_test_label, prc_test_pred)
confuse_matrix

##                prc_test_pred
## data_test_label   1   2   3   4   5
##               1   0   0   0   1  52
##               2   0   0   0   1  38
##               3   1   0   0   1  45
##               4   0   0   0   1  72
##               5   0   0   1   3 184
```

When we give the value of k as 75, we get the following confusion matrix:

```
prc_test_pred <- knn(train =data_train, test=data_test, cl =
data_train_label, k = 75)

confuse_matrix = table(data_test_label, prc_test_pred)
confuse_matrix

##                prc_test_pred
## data_test_label   1   2   3   4   5
##               1   0   0   0   0  53
##               2   0   0   0   0  39
##               3   0   0   0   1  46
##               4   0   0   0   0  73
##               5   0   0   0   2 186
```

D) kNN for predicting if a review is useful

In our dataset, we use 3 values of k (50,75,100) to predict if a particular review is useful or no based on the attributes review_cool, review_funny and review_stars.

```
review = read.csv(file = "D:\\CSV\\review.csv", sep = ";",nrows = 10000)
data_var <- review[,
c("review_stars","review_cool","review_funny","review_useful")]
table(data_var$review_useful)

##
##    0    1    2    3    4    5    6    7    8    9   10   11   12   13   14
## 5329 2180 1014  489  312  165  126   78   51   41   38   29   21   26   11
##   15   16   17   18   19   20   21   22   23   24   25   26   27   28   29
##   12    8    7    9    5    6    3    2    2    2    4    1    2    1    2
##   30   31   32   33   34   35   36   39   42   46   52   62   69   74   90
##    3    2    1    2    1    2    1    2    1    1    1    1    2    1    1
##  102  107
##    1    1
```

We again normalize the data so it is in the range between 0 to 1.

```
normalize <- function(x){
  return ((x - min(x))/(max(x)- min(x)))
}
data_new <- as.data.frame(lapply(data_var[1:3], normalize))

summary(data_new)

##   review_stars     review_cool       review_funny
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.000000
##  1st Qu.:0.5000   1st Qu.:0.00000   1st Qu.:0.000000
##  Median :0.7500   Median :0.00000   Median :0.000000
##  Mean   :0.6845   Mean   :0.00513   Mean   :0.005917
##  3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.:0.000000
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.000000
```

We split our data to 80:20 ratio to form a training and testing data.

When we give the value of k as 50, we get the following confusion matrix:

```
prc_test_pred <- knn(train =data_train, test=data_test, cl =
data_train_label, k = 50)

confuse_matrix = table(data_test_label, prc_test_pred)
confuse_matrix

##                prc_test_pred
## data_test_label   0   1   2   3   4   5   6   7   8   9  10  11  12  13
##               0 103   0   0   0   0   0   0   0   0   0   0   0   0   0
##               1  43   7   1   0   0   0   0   0   0   0   0   0   0   0
##               2  12   4   0   0   0   0   0   0   0   0   0   0   0   0
##               3  10   2   1   0   0   0   0   0   0   0   0   0   0   0
##               4   4   1   0   0   0   0   0   0   0   0   0   0   0   0
##               5   1   1   2   0   0   0   0   0   0   0   0   0   0   0
##               6   0   1   1   0   0   0   0   0   0   0   0   0   0   0
##               7   2   0   0   0   0   0   0   0   0   0   0   0   0   0
##               9   0   1   0   0   0   0   0   0   0   0   0   0   0   0
##               13  0   0   1   0   0   0   0   0   0   0   0   0   0   0
##               18  0   0   1   0   0   0   0   0   0   0   0   0   0   0
##               20  0   0   1   0   0   0   0   0   0   0   0   0   0   0
##                 prc_test_pred
## data_test_label  14  16  17  19  27  29  74
##               0   0   0   0   0   0   0   0
##               1   0   0   0   0   0   0   0
##               2   0   0   0   0   0   0   0
##               3   0   0   0   0   0   0   0
##               4   0   0   0   0   0   0   0
##               5   0   0   0   0   0   0   0
##               6   0   0   0   0   0   0   0
##               7   0   0   0   0   0   0   0
##               9   0   0   0   0   0   0   0
##               13  0   0   0   0   0   0   0
##               18  0   0   0   0   0   0   0
##               20  0   0   0   0   0   0   0
```

When we use k as 50, we get accuracy as 55%. When we give the value of k as 75, we get the following confusion matrix:

```
prc_test_pred <- knn(train =data_train, test=data_test, cl =
data_train_label, k = 75)

confuse_matrix = table(data_test_label, prc_test_pred)
confuse_matrix

##                prc_test_pred
## data_test_label   0   1   2   3   4   5   6   7   8   9  10  11  12  13
##               0 103   0   0   0   0   0   0   0   0   0   0   0   0   0
##               1  46   5   0   0   0   0   0   0   0   0   0   0   0   0
##               2  13   3   0   0   0   0   0   0   0   0   0   0   0   0
##               3  10   3   0   0   0   0   0   0   0   0   0   0   0   0
##               4   4   1   0   0   0   0   0   0   0   0   0   0   0   0
##               5   3   1   0   0   0   0   0   0   0   0   0   0   0   0
##               6   1   1   0   0   0   0   0   0   0   0   0   0   0   0
##               7   2   0   0   0   0   0   0   0   0   0   0   0   0   0
##               9   1   0   0   0   0   0   0   0   0   0   0   0   0   0
##               13  0   1   0   0   0   0   0   0   0   0   0   0   0   0
##               18  0   1   0   0   0   0   0   0   0   0   0   0   0   0
##               20  0   1   0   0   0   0   0   0   0   0   0   0   0   0
##                 prc_test_pred
## data_test_label  14  16  17  19  27  29  74
##               0   0   0   0   0   0   0   0
##               1   0   0   0   0   0   0   0
##               2   0   0   0   0   0   0   0
##               3   0   0   0   0   0   0   0
##               4   0   0   0   0   0   0   0
##               5   0   0   0   0   0   0   0
##               6   0   0   0   0   0   0   0
##               7   0   0   0   0   0   0   0
##               9   0   0   0   0   0   0   0
##               13  0   0   0   0   0   0   0
##               18  0   0   0   0   0   0   0
##               20  0   0   0   0   0   0   0

accurracy = sum(diag(confuse_matrix))*100/sum(confuse_matrix)
accurracy

## [1] 54
```

When we use k as 75, we get accuracy as 54%. When we give the value of k as 100, we get the following confusion matrix:

```
prc_test_pred <- knn(train =data_train, test=data_test, cl =
data_train_label, k = 100)

confuse_matrix = table(data_test_label, prc_test_pred)
confuse_matrix

##                prc_test_pred
## data_test_label   0   1   2   3   4   5   6   7   8   9  10  11  12  13
##              0  103   0   0   0   0   0   0   0   0   0   0   0   0   0
##              1   51   0   0   0   0   0   0   0   0   0   0   0   0   0
##              2   15   1   0   0   0   0   0   0   0   0   0   0   0   0
##              3   13   0   0   0   0   0   0   0   0   0   0   0   0   0
##              4    5   0   0   0   0   0   0   0   0   0   0   0   0   0
##              5    4   0   0   0   0   0   0   0   0   0   0   0   0   0
##              6    2   0   0   0   0   0   0   0   0   0   0   0   0   0
##              7    2   0   0   0   0   0   0   0   0   0   0   0   0   0
##              9    1   0   0   0   0   0   0   0   0   0   0   0   0   0
##              13   0   1   0   0   0   0   0   0   0   0   0   0   0   0
##              18   0   1   0   0   0   0   0   0   0   0   0   0   0   0
##              20   0   1   0   0   0   0   0   0   0   0   0   0   0   0
##                prc_test_pred
## data_test_label  14  16  17  19  27  29  74
##              0    0   0   0   0   0   0   0
##              1    0   0   0   0   0   0   0
##              2    0   0   0   0   0   0   0
##              3    0   0   0   0   0   0   0
##              4    0   0   0   0   0   0   0
##              5    0   0   0   0   0   0   0
##              6    0   0   0   0   0   0   0
##              7    0   0   0   0   0   0   0
##              9    0   0   0   0   0   0   0
##              13   0   0   0   0   0   0   0
##              18   0   0   0   0   0   0   0
##              20   0   0   0   0   0   0   0

accuracy = sum(diag(confuse_matrix))*100/sum(confuse_matrix)
accurracy

## [1] 54
```

When we use k as 100, we get accuracy as 54%.

E) Predicting stars using the text in the review:

In this we first read all the rows of the review_text and generate a dtm(Document Term Matrix) which gives a count of each word in the review_text.

```
review = read.csv(file = "D:\\CSV\\review.csv", sep = ";")
library(text2vec)
txt=review$review_text
txt<-as.character(txt)
prep = tolower
tofn = word_tokenizer
it_train = itoken(txt,preprocessor = prep, tokenizer = tofn,progressbar = TRUE)
vocab = create_vocabulary(it_train)
Vectorizer = vocab_vectorizer(vocab)
```

Above is the script for creating the Document Term Matrix.

We use a library glmnet which takes the Document Term Matrix as an input and tries to learn and predict the number of stars on the based of the count of the words.

```
Vectorizer = vocab_vectorizer(vocab)
dtm = create_dtm(it_train,Vectorizer)
dim(dtm)
vocab
library(glmnet)
glm = cv.glmnet(x=dtm,y=review$review_stars, nfolds = 4)
plot(glm)
ccc = cv.glmnet(x=dtm,y=review$review_stars, alpha=1 , type.measure = "deviance", nfolds = 4)
plot(ccc)
```