02nd Feb, 2020 (Pune, India)

Organizers: Mayank Mishra, Disha Patil, Ruturaj Nalawade, Saranjeet Kaur, Ravindra Pawse, Shreyas Kalmegh

What is PyData?

- Educational program of NUMFOCUS (a non-profit organization)
- Share ideas about data analysis tools
- Community-driven conference (novice to advance)

Frequent Topics

- Machine Learning
- Artificial Intelligence
- Predictive Modelling
- Data Mining
- Natural Language Processing
- Probabilistic Programming

What is R-Ladies?

- Registered as California-based non-profit organization
- Mission is to promote gender diversity in the R community
- Support R enthusiasts (of all proficiency levels)

Frequent Topics

- All things R!
- Immediate way ahead would include meetups on usage of tidyverse and ggplot2

K-Nearest Neighbours (Pseudocode)

- (X_i, Y_i); $i = 1, \ldots, n$. Known values.
- Y is the class label of X.
- **Objective**: The class label of x is unknown, to be found using k-NN.
- Compute $d(x, X_i)$; $i = 1, \ldots, n$; where, d is, say, the Euclidean distance.
- Sort the n Euclidean distances in non-decreasing order.
- Take first k distances from this sorted list, where, k is a pre-specified positive integer).
- Find the k corresponding points to these k-distances.
- k_i is the number of points belonging to the ith class among the k points.
- If $k_i > k_j$; $\forall\ i \neq j$, assign x to class i.

K-Means Clustering (Pseudocode)

Working:
1)Initialize cluster centers using some method
2) Assign each point to one cluster
3) Recalculate cluster centers
4) Repeat 2 and 3 until either process converges by stabilized cluster centers with given tolerance, or it reaches to maximum no of iterations
5) Perform 1 to 4 for n times with new distinct initiation of cluster
6) Pick the best result out of n times based on average intra cluster spread

Implementation Guidelines:

1) Class based implementations
2) Sk-learn like interfaces .fit, .predict .labels methods
3) Only for numeric data
4) Class signature : n_cluster, Init : only random initiation is fine, n_init, max_iter, Tol, Random state, no_of_jobs, algorithm

Python Cheat Sheet

- import pandas as pd; import numpy as np
- Read .csv file: pd.read_csv("filename") Write .csv file: pd.to_csv("filename")
- Some important methods:
  - df.head() ##by default print first 5 rows. Enter integer to print specific number of rows
  - df.info() ## information like column names, data types, index etc
  - df.describe() ## descriptive stats for numerical columns
- Subsetting:

Integer based: df.iloc[row_num, col_num] #0 based indexing

```
df.iloc[0,1]              ## extract single value i.e 1st row and 2nd column
df.iloc[:3, :4]       ## extract values from 1st three rows and 1st 4 columns
df.iloc[[2,3], [4,5]] ## extract values from 3rd and 4th rows and 5th and 6th columns
```

Label based: df.loc[index_label, column_label]

```
df.loc[123, "weight"] ## extract single value with index as 123 and column weight
df.loc[:, "petal length":"sepal length"] ## extract all rows with columns from petal lentgh to sepal length
df.loc[:, ["height", "weight"]] ## extract all rows with columns height and weight
```

Boolean based:

```
df.loc[df["weight"] > 70] ## select all rows with weight values greater than 70
df.loc[np.isin(df["weight"], [50, 70])] ## select all rows with weight as 50 or 70
df.loc[
(df["height"] > 160]) &
np.isin(df["weight"], [50, 70])
]  ## select all rows with height > 160 and weight as 50 or 70
```

Mathematical operations:

```
df["area"] = df["length"] * df["breadth"]
```

Apply/map:

```
df["area"].map(lambda x: f"area: {x}")
df.apply(function)
```

Groupby:

```
df["count of song by genre and year"] = df.groupby(["genre", "year"])["song name"].count()
```

R Cheat Sheet

- Read .csv file: read.csv("filename") and Write .csv file: write.csv("filename")
- Named Lists
  X<-c() ## vector intialization
- Subsetting:
  X[2] ## extracts the second element of X
  df[1, 1], df[2,],df[1, 'column_name'],df[, 'column_name']
- Named Lists
  Alist=list() ## initialise list
  Alist[['element name']]=vector or matrix ## store vectors to list
- Subset lists
  Alist[[name]][2] ## extract second element from Alist sublevel 'name'
- Loops
  for(i in 1:10){ …. } or while(condition){ …. }
- Functions: function_name <- function(arguments) { definition }
- The sample() function is used to randomly sample from a vector
- The apply() function family : sapply() and lapply()
- Help in R is ?topic or ??topic
- Bools : TRUE FALSE
- Error handling : tryCatch()
- AIM :
  From scratch implementation
  plotting functionality to the results ## easy target
  R package or Shiny app ## ambitious target

Java Cheat Sheet

- Variables: {public | private} [static] type name [= expression | value];
- Methods: {public | private} [static] {type | void} name(arg1, ..., argN){statements}
- BufferReader: BufferedReader reader = new BufferedReader(new InputStreamReader(System.in)); String name = reader.readLine();
- Class: public class Demo{ public static void main(String[] args)
- { System.out.println("Hello!");}}
- Iteration: for (int ctr = 1; ctr <= n; ++ctr) {System.out.print(ctr);
  for each loop: for (int val: someCollection) {}
- Selection: if (condition) {expression} else {expression}
  switch (var) { case 1: expression; break; default: expression; break; }
- Arrays: dataType[] varName= new dataType[size];
  dataType[][] varName = new dataType[row][col];
- Compile a Java Program :
  - Save your Java Program by the name of the class containing main() method along with .java extension: className.java
  - Call the compiler using javac command: javac className
  - Execute: java className
- Build tools: Maven
  Creating a Project: mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.4 -DinteractiveMode=false
  Build the Project: mvn compile; mvn clean package; mvn test
- Gradle: Creating a Project: gradle init --type java-application --test-framework spock
  Build the Project: ./gradlew build

Thanks to all attendees!
and sponsors!