Data Intensive Computing (CSE – 587)
Assignment 2

Group Members :

1) Anuj Narayanaswamy – anujnara - 50341631

2) Sai Abhishek Koppu – saiabhis - 50317410

3) Soumyya Kanti Datta- soumyyak – 50336866

Report 1: Word Count

The code is divided into two parts:

1) **Mapper:** In the mapper , the code reads each line from the 3 input text files and then splits them into words . We are splitting the words separated by space, “--" and comma. Then it prints those words one by one as key, with 1 as their value .
2) **Reducer :** In the reducer, the code reads the output given by the mapper as input and splits it into key and value pair. There it aggregates the number of times a word is present and adds up its value. Then it prints the total number of times a word is present in the 3 input text files. This is done for all the words one by one to give the output.

   files : mapper1.py , reducer1.py

Report 2: N-Grams

1. **Mapper logic:** The mapper is given the Gutenberg dataset as input. The mapper function goes through the articles line by line. Each line is split into a list consisting of only the words, and all unwanted symbols, characters and spaces are removed. Then we generate all the tri-grams around given keywords (science, sea, fire) and replace the keyword with ‘$’. We print all these tri-grams with a count of 1.
2. **Reducer logic:** Reducer receives all the relevant tri-grams from the mapper. It aggregates the count for the tri-grams and stores the trigram along with the count in a dictionary. The dictionary is sorted in descending order of count. Then the top 10 tri-grams that have the highest counts are printed.

   files : mapper2.py , reducer2.py

Report 3: Inverted Index

The code is divided into two parts:

> **Mapper :** In mapper , the code reads each line from the 3 input text files and splits them into words. We are splitting the words separated by space, “--" and comma. Then it prints those words one by one as key, with the name of the text file it was present in, as their value.
> **Reducer :** In the reducer, the code reads the output given by the mapper as input and splits it into key value pairs. There it prints the word and the name of all files it is present in. This process is done for all the words one by one to produce the output.
>
> files : mapper3.py , reducer3.py

Report 4: Relational Join

Pre processing required for this step is to convert the existing xslx files into Tab space delimited text files using an online converter.

The code is divided into two parts:

> **Mapper :** In input to the mapper is each line from the text files which represent one row each. Since these rows are tab delimited we use the split and arrange the data into columns and parse them. The output to the mapper is the employee ID as the key and the rest of the columns as Value.
> **Reducer :** The input to each reducer is the key and the corresponding values. We store the key in a dictionary with the key and all the values are appended to this key. The output of the reducer is now simply the dictionary which contains the employee ID as the key and the rest of the columns as values.
>
> files : mapper4.py , reducer4.py

Report 5: KNN

We pass a hyperfile parameter. This file is the Test.csv to each and every mapper given the Test.csv is small with only 15 rows in it.

External libraries used : from scipy we use stats to find mode, numpy to handling data and pandas for reading csv files.

The code is divided into two parts:

> **Mapper :** In input to the mapper is each line from the Train.csv file. Each line is then split based on comma (,) which represents one row. We also read the Test.csv which is passed on as a hyper file parameter. We compare the given row's distance with each and every row of the test file. The output of the mapper is the test row as key and values as the distances and the label.
> **Reducer :** The input to each reducer is the key and the corresponding values. We store the key and append all the values to a numpy array. We now sort this numpy array based on distances and take the top k values and pick the mode of the labels for these top k values. We then output from the reducer as the Test row as key and the Label as the value.
>
> files : mapper5.py , reducer5.py