Big Data Coursework

Imperial College London

Department of Mathematics

MSc in Statistics

Author:
Shane Mullins (CID: 01338546)

Date: April 22, 2022

Requested command outputs:

5fbcdaebabc758c70be2fcfd35644cbc prox-fixed.csv
e3f7dcd9fd02658395b4210faa94b07c prox-mobile.csv
786b99e77f5aa17349ca7d8719494c13 bldg-measurements.csv
745abb569b0e4f615d99dc95116b51b3 f2z2-haz.csv

Map Reduce

In this section, the following questions show the ANSWER, Shell commands in blue boxes and Python mapper/reducer codes in green boxes. In the following questions, the code for a combiner would be the same as that of a reducer but implemented after the mapper. In this case they were not implemented, however, when dealing with more intensive operations they would be useful to alleviate some of the information transfer burden.

1.

```
hadoop jar $HADOOP_STR/hadoop-streaming-2.7.0-mapr-1808.jar -libjars
    $HADOOP_STR/hadoop-streaming-2.7.0-mapr-1808.jar -input coursework/prox* -
    output q1ttttt-output -mapper "python mapper.py" -file /home/user404/bd-sp
    -2017/data/mapper.py -reducer "python reducer.py" -file /home/user404/bd-
    sp-2017/data/reducer.py
```

mapper.py

```
#!/usr/bin/env python
import sys

# input comes from STDIN
for line in sys.stdin:
        # remove leading and trailing whitespace
        line = line.strip()
        day,_,pid = line.split(",")[0:3] # id corresponds to 3rd elements
        key = '%s-%s' % (day[8:10],pid)
        print '%s\t%s' % (key, 1)
```

reducer.py

```
#!/usr/bin/env python
import sys

current_day = ""
current_id = ""
current_count = 0

# input comes from STDIN
for line in sys.stdin:
        # remove leading and trailing whitespace

        line = line.strip()
        if ' prox-id' in line: # remove header
                continue

        # parse the input we got from mapper.py
        key, count = line.split('\t')
        day, id = key.split('- ')

        # convert count (currently a string) to int
        try:
                count = int(count)
        except ValueError:
                # count was not a number, so silently
                # ignore/discard this line
                continue

        # this IF-switch only works because Hadoop sorts map output
        # by key (here: word) before it is passed to the reducer
        if current_day == day and current_id == id:
               # current_count = count
                continue
        elif current_day == day:
                current_id = id
                current_count += count

        else:
                if current_id:
                        # write result to STDOUT
                        print '%s\t%s' % (current_day, current_count)
                current_count = count
                current_day = day
                current_id = id

# do not forget to output the last word if needed!
if current_day == day and current_id == id:
        print '%s\t%s' % (current_day, current_count)
```

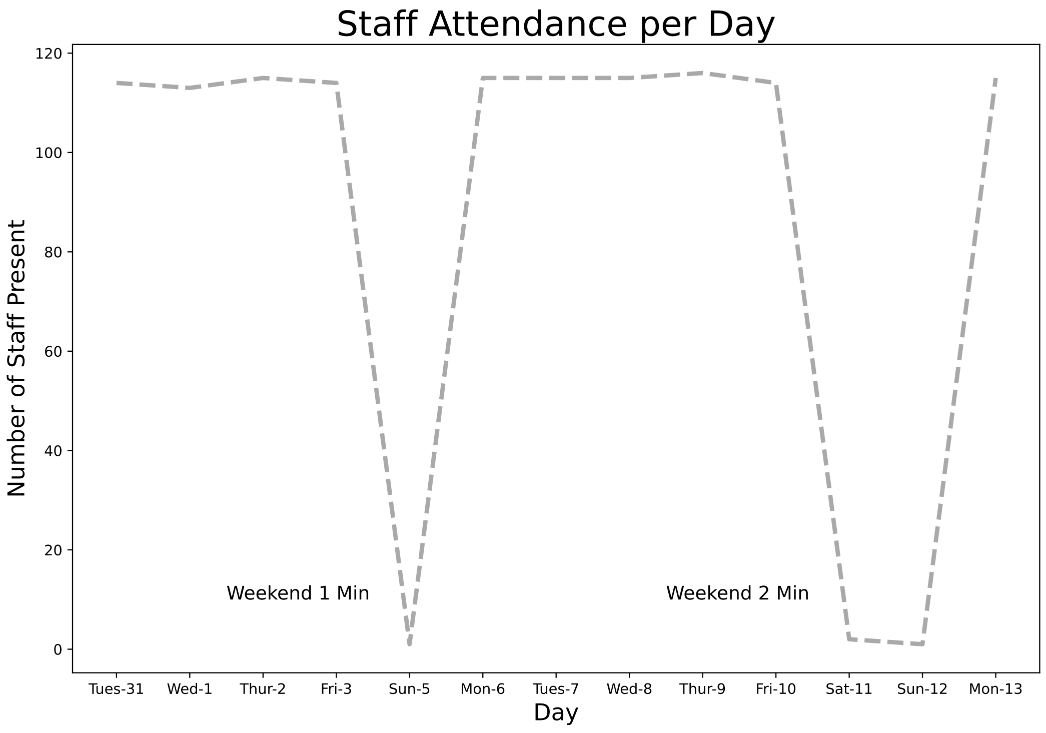

Figure 1: Number of staff members present in the building on each day. Weekend dips annotated. X axis labels refer to: day of week - date of month. Plotting on local computer is a feasible solution as the summary data is very unlikely to be large enough to qualify as Big Data.

2.

ANSWER: Most visited location in the building: floor 2, zone 1, 6146 card readings.

```
hadoop jar $HADOOP_STR/hadoop-streaming-2.7.0-mapr-1808.jar -libjars
    $HADOOP_STR/hadoop-streaming-2.7.0-mapr-1808.jar -input coursework/prox-
    fixed.csv -output q2-output -mapper "python mapper2.py" -file /home/
    user404/bd-sp-2017/data/mapper2.py -reducer "python reducer2.py" -file /
    home/user404/bd-sp-2017/data/reducer2.py
```

mapper.py

```
import sys

for line in sys.stdin:

        line = line.strip()
        floor,zone = line.split(",")[3:5]
        key = '%s-%s' % (floor,zone)
        print '%s\t%s' % (key, 1)
```

reducer.py

```
import sys

current_key = ""
current_count = 0

for line in sys.stdin:

        line = line.strip()
        if 'floor' in line: # remove header
                continue

        key, count = line.split('\t')
        floor, zone = key.split('- ')

        try:
                count = int(count)
        except ValueError:
                continue

        if current_key == key:
               current_count += count

        else:
                if current_key:
                        print '%s\t%s' % (current_key, current_count)
                current_count = count
                current_key = key

if current_key == key:
        print '%s\t%s' % (current_key, current_count)
```

3.

ANSWER: Prox-ID of most active staff member June 2nd 2016: fresumir001, 64 card readings.

```
hadoop jar $HADOOP_STR/hadoop-streaming-2.7.0-mapr-1808.jar -libjars
   $HADOOP_STR/hadoop-streaming-2.7.0-mapr-1808.jar -input coursework/prox* -
   output q3t-output -mapper "python mapper3.py" -file /home/user404/bd-sp
   -2017/data/mapper3.py -reducer "python reducer3.py" -file /home/user404/bd
   -sp-2017/data/reducer3.py
```

mapper.py

```
import sys

for line in sys.stdin:

        line = line.strip()
        day,_,pid = line.split(",")[0:3]
        key = '%s-%s' % (day[8:10],pid)
        print '%s\t%s' % (key, 1)
```

reducer.py

```
import sys

current_id = ""
current_count = 0

for line in sys.stdin:

        line = line.strip()
        if ' prox-id' in line:
                continue

        key, count = line.split('\t')
        day, prox_id = key.split('- ')

        try:
                count = int(count)
        except ValueError:
                continue

        if day == '02':
                if current_id == prox_id:
                        current_count += count

                else:
                        if current_id:
                                print '%s\t%s' % (current_id, current_count)
                        current_count = count
                        current_id = prox_id

if current_id == prox_id:
        print '%s\t%s' % (current_id, current_count)
```

4.

ANSWER: Figure 2 shows an expected decrease in the power used by the building in the early hours of the morning and late hours of night. A typical workday may span from 8am-5pm which corresponds to the region of highest power usage in this case. The dips at 10am and 2pm may be related to company-specific power behaviour such as machine resets/lunch breaks etc.

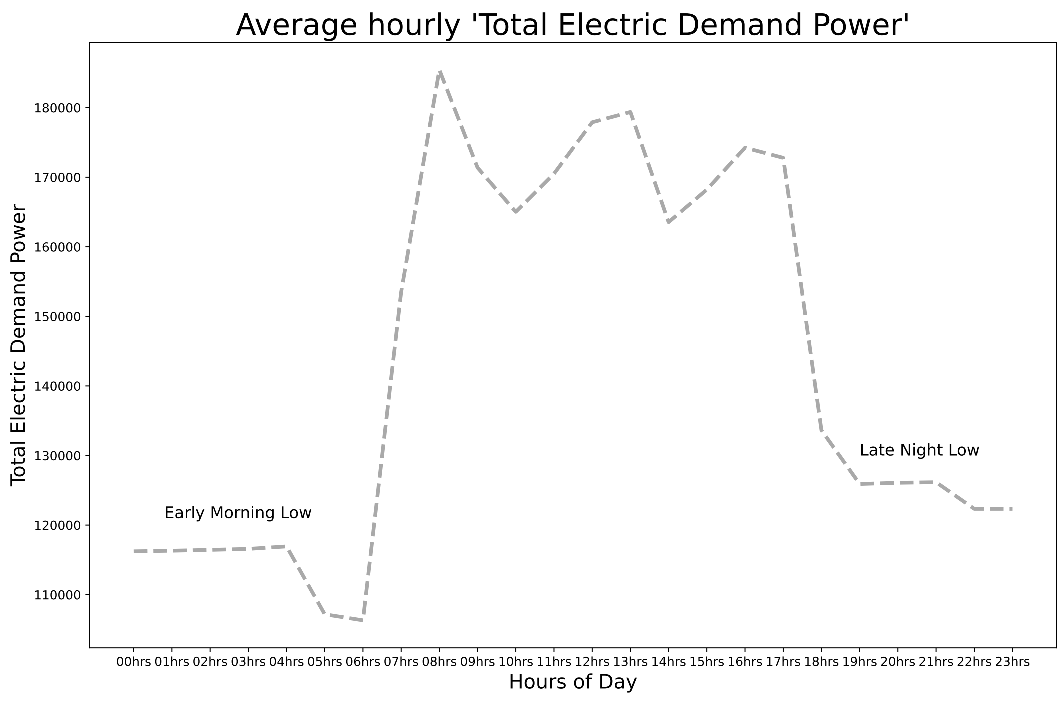

Figure 2: Average hourly "Total Electric Demand Power" shows lower power usage by building in early morning and late night.

```
hadoop jar $HADOOP_STR/hadoop-streaming-2.7.0-mapr-1808.jar -libjars
    $HADOOP_STR/hadoop-streaming-2.7.0-mapr-1808.jar -input coursework/bldg-
    measurements.csv -output q4-output -mapper "python mapper4.py" -file /home
    /user404/bd-sp-2017/data/mapper4.py -reducer "python reducer4.py" -file /
    home/user404/bd-sp-2017/data/reducer4.py
```

mapper.py

```
import sys

for line in sys.stdin:
        line = line.strip()
        if 'Date/Time' in line:
                continue
        hour = line.split(",")[0]
        tot_elec = line.split(",")[8]
        print '%s\t%s' % (hour[11:13], tot_elec)
```

reducer.py

```
import sys

current_hour = 0
current_elec =float(0)
current_counter = 0

for line in sys.stdin:

        line = line.strip()
        hour, elec = line.split('\t')
        elec = float(elec)

        if current_hour == hour:
                current_elec += elec
                current_counter += 1

        else:
                if current_hour:
                        avg_elec = current_elec/current_counter
                        print '%s\t%s' % (current_hour, avg_elec)
                current_counter = 0
                current_hour = hour
                current_elec = elec

if current_hour == hour:
        print '%s\t%s' % (current_hour, avg_elec)
```

Spark

Similar to the previous section, the layout below follows ANSWER then Spark commands in yellow boxes.

5.

```
val proxfixFile = sc.textFile("file:///home/user404/bd-sp-2017/data/prox-fixed.
   csv")

import org.joda.time.format.{DateTimeFormat, DateTimeFormatter}
import org.joda.time.DateTime

case class ProxReading(timeStamp: org.joda.time.DateTime, id: String, floorNum:
   String, zone: String)

def parse(line: String) = {
val tform = DateTimeFormat.forPattern("yyyy-MM-dd HH:mm:ss")
val pieces = line.split(',')
val timeStamp = tform.parseDateTime(pieces(0))
val id = pieces(2).toString
val floor = pieces(3).toString
val zone = pieces(4).toString
ProxReading(timeStamp, id, floor, zone)
}

def isHeader(line: String): Boolean = {
line.contains("timestamp")
}

val proxfixData = proxfixFile.filter(x => !isHeader(x)).map(parse)
```

6.

ANSWER: Most visited location in the building: floor 2, zone 1, 6146 card readings.

```
val proxfixflzone = proxfixData.map(x => (x.floorNum,x.zone)).map((_,1))
val proxfixloccount = proxfixflzone.reduceByKey(_+_)
proxfixloccount.sortBy(_._2,false).take(5).foreach(println)
```

7.

ANSWER: Prox-ID of most active staff member June 7th 2016: fresumir001, 64 card readings.

```
val proxfixred = proxfixData.filter(x => x.timeStamp.getMonthOfYear==6).filter(x
    => x.timeStamp.getDayOfMonth==7)
val proxmobFile = sc.textFile("file:///home/user404/bd-sp-2017/data/prox-mobile.
    csv")
case class ProxReading1(timeStamp: org.joda.time.DateTime, id: String)
def parsemob(line: String) = {
val tform = DateTimeFormat.forPattern("yyyy-MM-dd HH:mm:ss")
val pieces = line.split(',')
val timeStamp = tform.parseDateTime(pieces(0))
val id = pieces(2).toString
ProxReading1(timeStamp, id)
}

def isHeader2(line: String): Boolean = {
line.contains("floor")
}
val proxmobData = proxmobFile.filter(x => !isHeader2(x)).map(parsemob)
val proxmobred = proxmobData.filter(x => x.timeStamp.getMonthOfYear==6).filter(x
    => x.timeStamp.getDayOfMonth==7)
val proxcomData = proxmobred.map(x => x.id).union(proxfixred.map(x => x.id))
val proxcomid = proxcomData.map((_,1))
val proxcomcount = proxcomid.reduceByKey(_+_)
proxcomcount.sortBy(_._2,false).take(5).foreach(println)
```

8

My experiences using both Map Reduce programmes and Spark commands can be summarised by the following evident differences between them, noticed when performing the previous tasks:

- Similar tasks involving the mapping and aggregation of many paired values were performed noticeably quicker using Spark. The time discrepancy between the two methods is likely to increase further with larger datasets due possibly to the lazy evaluation principles prevalent in Spark transformations. The use of caching would also speed up operations.
- In general, Spark commands were more concise than their Map Reduce equivalents. This is clear to see from the previous code outputs. This functional way of programming allowed for greater interpretability and, hence, easier construction and debugging.

9.

ANSWER: Date and time of the first occurrence of the 'F_2_Z_1 VAV REHEAT Damper Position' being fully open: 15:20 pm 2nd June 2016

```
val bldg = sc.textFile("file:///home/user404/bd-sp-2017/data/bldg-measurements.
   csv")
case class Measure(timeStamp: org.joda.time.DateTime, damper: String)
def isHeader1(line: String): Boolean = {
line.contains("Time")
}
def p1(line: String) = {
  val tform = DateTimeFormat.forPattern("yyyy-MM-dd HH:mm:ss")
  val pieces = line.split(',')
  val timeStamp = tform.parseDateTime(pieces(0))
  val damper = pieces(192).toString
   Measure(timeStamp, damper)
}
val bldgData = bldg.filter(x => !isHeader1(x)).map(p1)
def isMax(line: String): Boolean = {
line.contains(" 1.0000")
}
bldgData.filter(x=> isMax(x.damper)).take(5).foreach(println)
```

10.

Initially looking at the summary statistics through Spark's Mllib stat package tells us that both the damper position and Hazium concentration have similar means: ≈ 0.5704 and ≈ 0.6509 respectively. However, they lie on noticeably different ranges with the former between 0-1 and the latter 0-8.2330. One test to measure the statistical association between two variables is **Pearson's correlation**. Implementing this again through Mllib yields a low correlation of ≈ 0.1033. However, the **correlation between damper position and Hazium values above the mean is $\approx$ 0.3544**. There seems to be a stronger relationship between damper position and high levels of Hazium. 1343 of the total 3983 observations relate to damper values = 1 (i.e. fully open). The **mean Hazium concentration when the damper is fully open is $\approx$ 0.9312** which is significantly higher than the third quartile (0.7500). The **statistical findings above indicate a moderately strong relationship between damper position and Hazium levels**. The rogue employee's tactic to modify the damper position would likely achieve the desired Hazium concentration change.

```
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.regression.LinearRegressionModel
import org.apache.spark.mllib.regression.LinearRegressionWithSGD
import org.apache.spark.mllib.linalg.Vectors
val haz = sc.textFile("file:///home/user404/bd-sp-2017/data/f2z2-haz.csv")
def p2(line: String) = {
val tform = DateTimeFormat.forPattern("yyyy-MM-dd HH:mm:ss")
val pieces = line.split(',')
val timeStamp = tform.parseDateTime(pieces(0))
val damper = pieces(1).toString
Measure(timeStamp, damper)
}

val hazData = haz.filter(x => !isHeader1(x)).map(p2)

val hazpair = hazData.map(x=>(x.timeStamp,x.damper))
val bldgpair = bldgData.map(x=>(x.timeStamp,x.damper))
val combined_meas = bldgpair.join(hazpair)

import org.apache.spark.mllib.linalg.{Vector, Vectors}
import org.apache.spark.mllib.stat.{MultivariateStatisticalSummary, Statistics}
import org.apache.spark.mllib.linalg.{Matrix, Matrices}
val observations = combined_meas.map(_._2).map(x => Vectors.dense(x._1.toDouble,x
    ._2.toDouble))
val summary: MultivariateStatisticalSummary = Statistics.colStats(observations)
println(summary.mean)
println(summary.variance)
println(summary.numNonzeros)
val correlMatrix: Matrix = Statistics.corr(observations, "pearson")
println(correlMatrix.toString)


val combinedhigh =  combined_meas.filter(x => x._2._2>" 0.651")
val observations1 = combinedhigh.map(_._2).map(x => Vectors.dense(x._1.toDouble,x
    ._2.toDouble))
val correlMatrix: Matrix = Statistics.corr(observations1, "pearson")

val combinedones =  combined_meas.filter(x => x._2._1>" 1.000")
val observations2 = combinedones.map(_._2).map(x => Vectors.dense(x._1.toDouble,x
    ._2.toDouble))
```

11.

ANSWER: Employee IDs for those that entered the Server Room prior to the sudden increase in Hazium concentration: 'pyoung001', 'sflecha001', 'ncalixto001', 'lbennett001', 'csolos001'

```
val proxfixred1 = proxfixData.filter(x => x.timeStamp.getMonthOfYear==6).filter(x
    => x.timeStamp.getDayOfMonth==10).filter(x=>x.zone==" Server Room")
proxfixred1.take(10).foreach(println)
```

12

Throughout this module, we have focused on the successes of Big Data architectures, in particular distributed systems and their efficacy in dealing with faults of a systematic or random nature. However, in this era of widespread data analytics, data privacy is a topic of concern for many enterprises and academics alike. Big Data poses unique threats when it comes to information security and data confidentiality. How does the use of distributed systems 1) facilitate the continuous improvement of Big Data technology into the future and 2) affect the security of its associated data?

Notes on Solution:

- Question forces one to think about the, perhaps overlooked, consequences of the underlying structure of Big Data technology.
- Domain of possible answers lies within the scope of the course but requires some supplementary independent research.
- Balanced problem, drawing attention to both the opportunities and problems of Big Data technology.
- Big Data security is a relevant area of active research (Suri, n.d.), (Sun, Zhang, & Fang, 2021), (Ogbuke, Yusuf, Dharma, & Mercangoz, 2022).
- Some answers could focus on the scalability of distributed systems and the increased security risk of multiple systems containing copies of the same data.

13

In this section, the statistical relationship between crime and certain community factors, such as immigration, race and income, is investigated (Mullins, 2022). The UCI Machine Learning Repository 'Communities and Crime Data Set' (Repository, 2009) is used for this purpose[1]. The chosen subset of societally relevant attributes are: population size, black race proportion, white race proportion, immigrants proportion, median income, poverty level, years of education and unemployment levels[2]. The dataset documentation was consulted when making choices about this and the following statistical analysis.

The dataset yields a low mean community violent crime rate ≈ 0.2380 with a relatively large standard deviation of ≈ 0.2330. Expectedly, the correlation matrix in Figure 3 shows strong relationships between black/white proportions and number of immigrants/population, for example. **Black and white proportions have strong correlations with violent crime ≈ 0.63 and** ≈ -0.68 respectively. Using **Map Reduce in Hadoop**, the average violent crime rates for the top five states were found to be: 0.5045 (Louisiana), 0.4868 (S Carolina), 0.48 (Maryland), 0.4583 (Florida) and 0.4020 (N Carolina). The majority of the **most violent states were located in the South of the US**.

A host of machine learning models have been adapted for use within Big Data environments. This is due in part to the associative/commutative properties of their underlying mathematical construction. A **linear regression model**, using stochastic gradient descent, is implemented on the training data of the aforementioned attributes to determine whether individually they are good predictors of crime, as well as whether crime rates can be predicted based on community factors as a whole. The process of caching is used throughout to improve the efficiency of operations. Through relatively large coefficients (≈ 0.0136 and ≈ 0.0129 respectively[3]), the results support our previous claims that **racial proportions have a noticeable effect on crime**[4]. Additionally, unemployment levels seem to somewhat influence crime. These conclusions are subject to the assumptions of the linear model. An RMSE value of ≈ 0.4085 indicates that perhaps the attributes as a whole are not particularly accurate predictors.

To augment these findings, a **SVM binary classifier** is fitted to the data. The target variable crime is categorised into 0s and 1s indicating values below and above the mean. The resulting **AUC (Area Under Curve) values of ≈ 0.8932 and ≈ 0.8202 for the ROC and precision-recall** curves show strong community factor prediction qualities when it comes to classification. This may be somewhat subject to bias as the majority of data points lie below the mean.

The results above show some relatively strong links between crime and race, employment and geographical location. Given the limit on this question, the above analysis lacks some statistical sophistication, however, the ultimate goal of this discussion is to demonstrate the use of statistical reasoning applicable to Big Data within the chosen environment. Performing even simple statistical tests would be extremely computationally expensive with Big Data. Therefore, the power of Hadoop/Spark and, more specifically, Mllib is very impressive in the context of data larger than that mentioned here.

[1]More about the data and its sources is discussed in section 14 b.

[2]A state identifier is included preliminarily but not within any models. The interpretation of the attributes has changed slightly due to prior normalisation between 0 and 1. To avoid unnecessary processing steps, only balanced columns with no missing data are chosen. The Mllib library is used for the following analysis

[3]All results subject to randomness were averaged over multiple iterations with varying step sizes.

[4]Coefficient values can be used as a measure of variable importance here as all attributes have been normalised to the same scale.

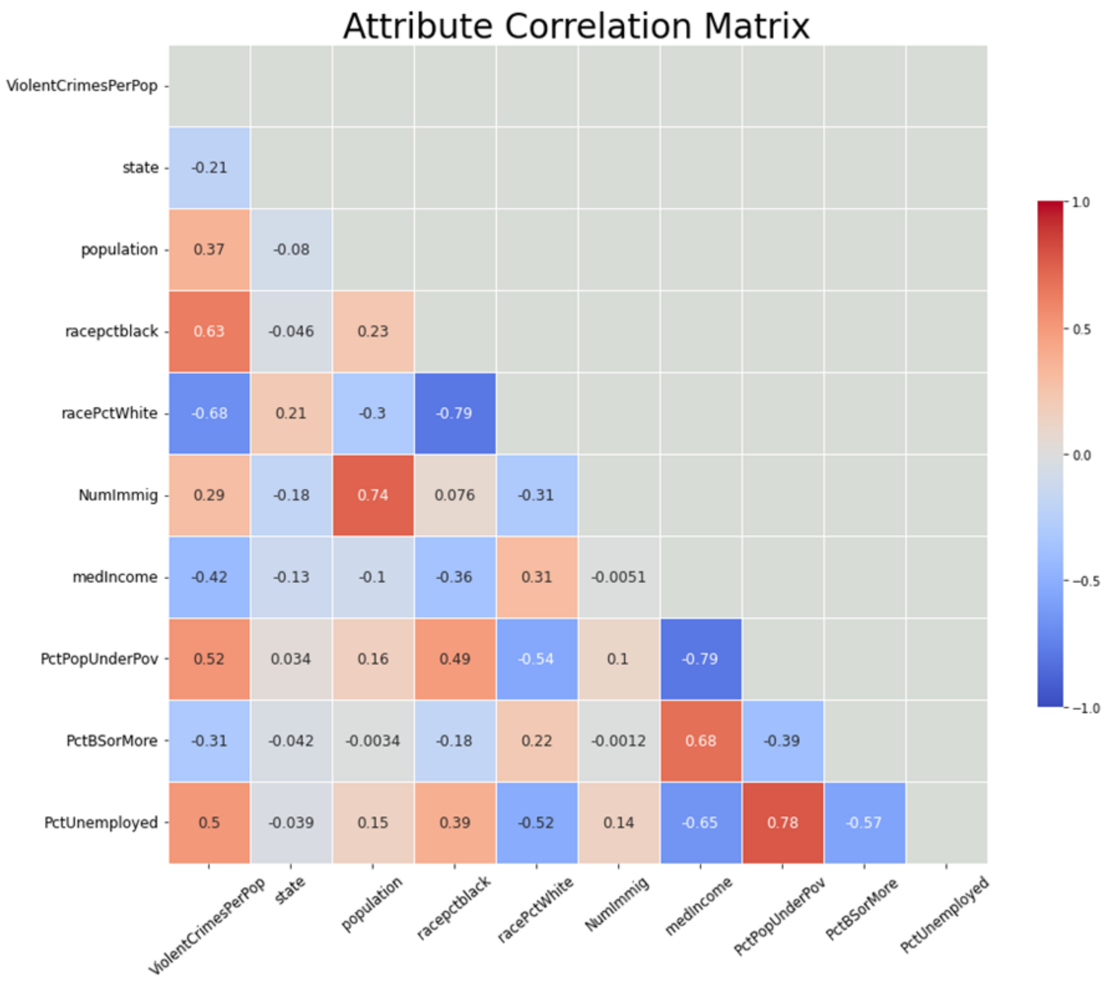

Figure 3: Heatmap showing correlations between attributes. Darker red and blue values indicate higher positive and negative correlations respectively.

14 a)

Professor Xiao-Li Meng of Harvard University highlight's the flaws associated with relying heavily on data quantity when it comes to statistical analysis in the context of Big Data (Meng, 2018). Many fundamental statistical ideas, such as the Law of Large Numbers and the Central Limit Theorem, are made possible through the asymptotic properties of large samples. However, the power of these, and other foundational ideas, diminishes when **non-probabilistic sampling procedures** are used to gather data, especially in the context of very large datasets. This is hugely relevant given that Big Data is often collected without inference in mind. As a potential solution, the author emphasises the need to include the **true population size** when performing statistical reasoning.

Using relatively straightforward statistics, Meng rearranges a sample average estimation error expression into three constituent terms representing **data quality, data quantity and problem difficulty**. He calls this the 'universal trio identity for estimation error'[5].

$$\bar{G}_n - \bar{G}_N = \underbrace{\rho_{R,G}}_{\text{Data Quality}} \times \underbrace{\sqrt{\frac{1-f}{f}}}_{\text{Data Quantity}} \times \underbrace{\sigma_G}_{\text{Problem Difficulty}}$$

[5]G = any function to be estimated, N = population size, n = sample size, R = sampling response rate, f = sampling rate and σ = variance.

The extent of the non-probabilistic nature of Big Data sampling dictates data quality and is represented as the **data defect correlation** $\rho_{R,G}$. Subsequently, it is shown that $\rho_{R,G}$ plays a significant role in estimation accuracy. The Law of Large Populations is used to convince us that **population size N, not sample size n, is the driver for sample mean error**. In order to maintain the traditional n^{-1} relationship with prediction MSE, the data defect index $E_R[\rho^2_{R,G}]$ must be maintained at an N^{-1} rate. Meng explores this in the context of the US 2016 presidential election through the concept of effective sample size. Despite using conservative estimates for sampling trustworthiness and removing certain complications such as response bias, a somewhat **non-probabilistic dataset of n = 2,315,570 people is equivalent to a probabilistic one of n = 400!** This is due to the nature of how the data is typically collected for these campaigns. Furthermore, Meng explains how a **'Big Data paradox'** results in highly inaccurate estimator confidence intervals that ironically disimprove with increased sample size. Due to ignorance of the true population size, these confidence intervals are centred far from the correct location.

Meng exemplifies the above concepts, namely data defect, the Big Data paradox and the effect of unreliable sampling in the form of underreporting, through the 2016 election data. The motivation for doing so is to aid prediction accuracy for the 2020 election. **The statistical contributions of this paper also extend to (Quasi) Monte Carlo integration and data confidentiality.** Meng builds on a general Monte Carlo estimation equation separated into three terms analogous to the aforementioned estimation error expression (hickernell2016trio). He draws similarities between the underappreciation of the ρ-equivalent term in relation to result errors. Through weighting techniques and the inclusion of zero-mean noise, Meng also discusses the link between data quality and data confidentiality, highlighting the proportional relationship between the two. Finally, the author establishes some preconditions for the exploration of the paper's main ideas in the context of individualised predictions rather than population inference.

The ultimate aim of this paper is to convince statisticians of the importance of data quality, primarily when it comes to Big Data. Although the results may speak to the inefficacy of common sampling techniques, Meng is really trying to convey to us the extraordinary power of probabilistic sampling.

b)

Statistics relies upon making inferences about a population using a representative sample of data. But what happens when this sample is no longer an accurate representation of the chosen population? In general, it is **unrealistic to assume data sources were accumulated through deliberate probabilistic sampling techniques**. In the context of Big Data, sampling procedures are typically conducted without consideration of possible future statistical inference tasks. This makes them highly non-probabilistic and, as such, some traditional statistical approaches are no longer appropriate. These ideas will be explored in the context of the crime dataset discussed in section 13. Although the data is not large enough the qualify as Big Data, we can still apply some of Meng's findings and extrapolate results to datasets of a larger magnitude.

Data Collection

From the statistical analysis in section 13, the hope is to study the relatively small sample of 1994 communities to gain insights into crime in all US communities. According to the documentation: 'The data combines socio-economic data from the 1990 US Census, law enforcement data from the 1990 US LEMAS survey, and crime data from the 1995 FBI UCR' (Repository, 2009). As Meng pointed out, one must account for the statistical differences brought about due to combining datasets of different quantity and quality. Each of these data sources potentially exhibit some form of (non) response bias leading to a non-probabilistic sampling scenario. The LEMAS survey in particular leaves room for poorer quality data. Even before looking at how specific questions were answered, 2945 out of 3118 law enforcement agencies chose to respond to the survey. Furthermore, **only agencies with more than 100 officers were selected for reporting**. Questions involved salaries, officer delegation, weapon use, drug enforcement activities and other related features. Reliable answers to these questions depend heavily on the honesty of the officer filling out the form. The above aspects of the survey leave a lot of room for both non-response and response bias.

The Uniform Crime Reporting (UCR) Program overseen by the FBI is also susceptible to reporting bias. The collection strategy is of a voluntary nature allowing for the **cherry-picking of data** submission. Nevertheless, even in the 1990 report, significant efforts were made to ensure high data quality. A whole redesign of the program was conducted prior to 1990 including multiple levels of quality assurance. This highlights the acceptance of the need for high quality data in the statistics community. The US Census boasts advantages over other data sources due to its sheer size. In this case, however, these advantages are somewhat lost as only a small portion of the entire dataset is analysed. This leaves it vulnerable to the aforementioned biases associated with self-reporting and the administrative recording of data.

Data Quality

It is also important to look at selection bias associated with this specific dataset's creation (Repository, 2009) . In 2009, Michael Redmond of La Salle University, Philadelphia gathered and modified the relevant crime data for statistical modelling purposes. Various attributes were hand-picked due to their apparent relevance and potential predictive quality. In **construction of the violent crime attribute, data associated with particular states were deliberately removed**. This was due to slight discrepancies in the definitions of assault and rape. This highlights the dangers associated with combining different datasets with varying intended uses. Although this differs somewhat from response bias, it contributes to the non-probabilistic nature of the data sampling.

From the preceding discussion, it is clear that the key points of Meng's paper are relevant to the analysis performed on the crime dataset. The non-probabilistic community data underlies the regression and classification models implemented, as well as the presented summary statistics and correlations. In keeping with the paper, it would be wise to estimate the **effective sample size** associated with truly probabilistic data. According to Statista.com, in 2019 there were $\approx 19,500$ cities, towns and villages in the US. Assuming that the number has not changed dramatically since 1990, the population size for this analysis is N = 19,500. Using a conservative estimate of $E_R[\rho^2_{R,G}] = 0.05$ (as was the case in the paper) the following formula is used to calculate effective sample size:

$$n_{\text{eff}} \leq 400 \frac{f}{1-f}$$

where f refers to the sample size as a proportion of N. Given that n = 1994, f $\approx$ 0.1. Therefore, $\mathbf{n_{\text{eff}} \approx 44}$. Using a less conservative data defect correlation figure of $E_R[\rho^2_{R,G}] = 0.1$, the effective sample size is reduced to 11 samples. This is a huge sample size reduction of 99.45%. These figures illustrate the importance of data quality in the context of this dataset.

Potential Solutions

There exists extensive research on the topic of inference from non-random samples, notably (Little, 1982) and (Smith, 1983). In recent years, this, and similar work, has been extrapolated to Big Data environments. We can look to these theoretical advancements for potential solutions to our problem. The majority of the following methods involve the use of external data. In order to improve the validity of inferences made on non-probabilistic data, (Kim & Wang, 2019) proposed the use of

inverse sampling and data integration. The first of these involves creating a simple random sample from the existing large sample. The second requires the inclusion of external auxiliary data of known probabilistic nature. A similar **sample matching approach** is discussed by (Chen, Li, & Wu, 2020) given the existence of a related probabilistic survey sample. More complex quasi-randomisation techniques have also been researched in this context (Elliott & Valliant, 2017). Due to the nature of crime data and its reliance on accurate post-event recollection, it may be difficult to procure a sample that statistically qualifies as probabilistic. With this in mind, perhaps inverse sampling may be an appropriate solution. Regardless of which approach is chosen, one must keep in mind the dangers of inference on potentially non-random samples. This is particularly true when dealing with Big Data where the **Law of Large Populations** plays a significant role. As Big Data technology and the awareness of its importance spreads amongst the scientific community and the public alike, one hopes that new methods of dealing with this problem will come about. Whether at the collection phase, through improved widespread sampling techniques perhaps involving IoT (Internet of Things), or at the inference phase building upon the methods discussed above, the importance of this endeavour should not be taken lightly.

Appendix: Q13 Code

```
val comFile = sc.textFile("file:///home/user404/bd-sp-2017/data/communities.csv")
def isHeader(line: String): Boolean = {
  line.contains("state")
}
def isMissing(line: String): Boolean = {
  line.contains("?")
}
case class CrimeRecord(crime: Float, state: Float, population: Float, black: Float,
white: Float, immigrants: Float, income: Float, poverty: Float, education: Float,
 unemployment: Float)
def parse(line: String) = {
  val pieces = line.split(',')
  val crime = pieces(128).toFloat
  val state = pieces(1).toFloat
  val population = pieces(6).toFloat
  val black = pieces(8).toFloat
  val white = pieces(9).toFloat
  val immigrants = pieces(57).toFloat
  val income = pieces(57).toFloat
  val poverty = pieces(34).toFloat
  val education = pieces(37).toFloat
  val unemployment = pieces(38).toFloat
  CrimeRecord(crime, state, population, black, white, immigrants, income,
poverty, education, unemployment)
}

val comData = comFile.filter(x => !isHeader(x)).map(parse)
comData.take(3).foreach(println)

import org.apache.spark.mllib.linalg.{Vector, Vectors}
import org.apache.spark.mllib.stat.{MultivariateStatisticalSummary, Statistics}
import org.apache.spark.mllib.linalg.{Matrix, Matrices}
val observations = comData.map(x => Vectors.dense(x.crime.toDouble,x.state.to
Double,x.population.toDouble,x.black.
toDouble,x.white.toDouble,x.immigrants.toDouble,x.income.toDouble,x.poverty
.toDouble,x.education.toDouble,x.unemployment.toDouble))
val summary: MultivariateStatisticalSummary = Statistics.colStats(observations)
println(summary.mean)
println(summary.variance)
println(summary.numNonzeros)
val correlMatrix: Matrix = Statistics.corr(observations, "pearson")
println(correlMatrix.toString)

import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.regression.LinearRegressionModel
import org.apache.spark.mllib.regression.LinearRegressionWithSGD
import org.apache.spark.mllib.linalg.Vectors

def parse2(line: String) = {
  val pieces = line.split(',')
  val crime = pieces(128)
  val population = pieces(6)
  val black = pieces(8)
  val white = pieces(9)
  val immigrants = pieces(57)
  val income = pieces(57)
  val poverty = pieces(34)
  val education = pieces(37)
  val unemployment = pieces(38)
  LabeledPoint(crime.toDouble, Vectors.dense(population.toDouble,black.
toDouble,white.toDouble,immigrants.toDouble,income.toDouble,poverty.
toDouble,education.toDouble,unemployment.toDouble))
}

val splits = comFile.randomSplit(Array(0.6, 0.4), seed = 11L)
val modData1 = splits(0).filter(x => !isHeader(x)).filter(x => !isMissing(x)).
map(parse2).cache
val numIterations = 100
val stepSize = 0.01
val model = LinearRegressionWithSGD.train(modData1, numIterations,
stepSize)
println(model.intercept)
println(model.weights)

val input = splits(1).filter(x => !isHeader(x)).filter(x => !isMissing(x)).
map(parse2)
val scoreAndLabels = input.map { point =>
  val preds = model.predict(point.features)
  (preds, point.label)
}
import org.apache.spark.mllib.evaluation.{RankingMetrics,
RegressionMetrics}
import org.apache.spark.mllib.recommendation.{ALS, Rating}
val regressionMetrics = new RegressionMetrics(scoreAndLabels)
println(s"RMSE = ${regressionMetrics.rootMeanSquaredError}")
```

```
val comData1 = splits(0).filter(x => !isHeader(x)).map(parse)
val bincomData = comData1.map(x => if (x.crime > 0.2318) (1,
x.population, x.black, x.white, x.immigrants, x.income, x.poverty,
x.education, x.unemployment) else (0, x.population, x.black,
x.white, x.immigrants, x.income, x.poverty, x.education,
x.unemployment)).cache
val bincomData1 = bincomData.map(x => LabeledPoint(x._1, Vectors.dense
(x._2,x._3,x._4,x._5,x._6,x._7,x._8,x._9)))

import org.apache.spark.mllib.classification.{SVMModel, SVMWithSGD}
import org.apache.spark.mllib.evaluation.BinaryClassificationMetrics
import org.apache.spark.mllib.util.MLUtils
val numIterations = 100
val binmodel = SVMWithSGD.train(bincomData1, numIterations)
binmodel.clearThreshold()
val comData2 = splits(1).filter(x => !isHeader(x)).map(parse)
val bincomData2 = comData2.map(x => if (x.crime > 0.2318) (1, x.population
, x.black, x.white, x.immigrants, x.income, x.poverty, x.education,
x.unemployment) else (0, x.population, x.black, x.white, x.immigrants,
x.income, x.poverty, x.education, x.unemployment)).cache
val bincomData3 = bincomData2.map(x => LabeledPoint(x._1, Vectors.dense
(x._2,x._3,x._4,x._5,x._6,x._7,x._8,x._9)))
val scoreAndLabels = bincomData3.map { point =>
  val score = binmodel.predict(point.features)
  (score, point.label)
}
val metrics = new BinaryClassificationMetrics(scoreAndLabels)
val auROC = metrics.areaUnderROC()
val auPR = metrics.areaUnderPR()
println(s"Area under ROC = $auROC")
```

References

Chen, Y., Li, P., & Wu, C. (2020). Doubly robust inference with nonprobability survey samples. *Journal of the American Statistical Association*, *115*(532), 2011–2021.

Elliott, M. R., & Valliant, R. (2017). Inference for nonprobability samples. *Statistical Science*, *32*(2), 249–264.

Kim, J. K., & Wang, Z. (2019). Sampling techniques for big data analysis. *International Statistical Review*, *87*, S177–S191.

Little, R. J. (1982). Models for nonresponse in sample surveys. *Journal of the American statistical Association*, *77*(378), 237–250.

Meng, X.-L. (2018). Statistical paradises and paradoxes in big data (i) law of large populations, big data paradox, and the 2016 us presidential election. *The Annals of Applied Statistics*, *12*(2), 685–726.

Mullins, S. (2022). *Analysis of crime rates in us communities.*

Ogbuke, N. J., Yusuf, Y. Y., Dharma, K., & Mercangoz, B. A. (2022). Big data supply chain analytics: ethical, privacy and security challenges posed to business, industries and society. *Production Planning & Control*, *33*(2-3), 123–137.

Repository, U. M. L. (2009). *Communities and crime data set.*

Smith, T. (1983). On the validity of inferences from non-random samples. *Journal of the Royal Statistical Society: Series A (General)*, *146*(4), 394–403.

Sun, L., Zhang, H., & Fang, C. (2021). Data security governance in the era of big data: status, challenges, and prospects. *Data Science and Management*, *2*, 41–44.

Suri, N. (n.d.). Distributed systems security knowledge area version..