CMP3749M

Big Data Module

Assignment 1 – Report

Name: Alexandre Barlaam (19695898@students.lincoln.ac.uk)

Student Number: 19695898

Contents

Task One – Narrative

Introduction

In this research review, I will consider four approaches to big data analysis. Each method will be explained, advantages and disadvantages of each will be considered as well as fundamental limitations all in the context of a problem scenario. The problem scenario chosen is the processing of vaccination and Covid-19 data during the ongoing pandemic done by the NHS in the UK.

Data Modelling

In big data, Data modelling is stated to be "methods of fast and cost-effective mathematical analysis with approximate relationships between variables" (Galetsi, Katsaliaki, & Kumar, 2020). In our problem scenarios' case, this can be finding links in our data set between the severity of covid cases, and the vaccinations. The data, once modelled, could indicate that people that have had both doses of a vaccine and a booster dose have a reduced chance of hospitalisation following catching covid, for example.

Data Modelling has the advantage of allowing for easy understanding of data and the relationship between it and other data entities. It can help companies understand their data better to establish how it should be used and for what purpose (Mullins, 2021). However, if the relationships that are made are overly approximate or broad, this can lead to conclusions that are inaccurate or vague which could impact the results taken from that data and the understanding gained from analysing it.

This method has the strength is that it can be used to easily gain a better understanding of the data that the health sector has in our case. This can be useful to put the necessary measures in place for Covis-19 for example. However, the links made within the variables in the data have to be accurate. Otherwise, findings from the data can influence potentially bad decisions being made which is one of the limitations.

Data Visualization

Another approach is Data visualisation. This can be simply defined as "Creating tables, images, diagrams, and other intuitive ways to understand data" (Galetsi, Katsaliaki, & Kumar, 2020). In context, this can be interpreted to be graphs to show the current daily positive cases for covid, for example or it also could be a diagram to illustrate the vaccine uptake in the population including first, second, and booster doses.

An advantage of this method is that it makes it much easier for the average person to understand. Other methods tend to be much less accessible to the public as the trends are less obvious often appearing in numbers rather than being illustrated as seen here. On the flipside, trends can be skewed due to inaccurate data. This is true for most big data techniques but especially so here as it is much more obvious in graphs and charts and can appear more skewed than in other approaches. Furthermore, some can only look at the important data without taking into consideration the bigger picture. This can lead to biased results in cases.

Overall, this method's strengths lie in the ease of illustrating information and sharing it with almost anyone which is important when thinking about Covid-19 data. However, its limitations are that it is sensitive to inaccurate data which must be removed before the data can be processed. It can also lead to biased results as discussed formerly.

Machine Learning

Thirdly, we have the machine learning approach. It is defined as "AI aimed to design algorithms that allow computers to evolve behaviours based on empirical data" (Galetsi, Katsaliaki, & Kumar, 2020). These algorithms "learn from data and provide data driven insights, decisions, and predictions" (L'Heureux, Grolinger, Elyamany, & Capretz, 2017). Such Algorithms could be used to predict how the pandemic will advance based on the trends of data that are present in the dataset.

An advantage of ML is the "presumption that algorithms can learn better with more data" (L'Heureux, Grolinger, Elyamany, & Capretz, 2017). This is perfect for big data as they have access to large datasets and therefore can make more accurate predictions and insights. This comes with drawbacks, however. Traditional ML algorithms "were designed for smaller datasets" (L'Heureux, Grolinger, Elyamany, & Capretz, 2017) assuming that "the entire dataset can fit in memory". In big data, this is not practical nor true as the datasets are simply too sizable.

This method's strengths lie in the fact that the more data it is supplied with. The better the predictions are. This is important when predicting data for any application including our own scenario. However, while the memory can cause a limitation as discussed previously, some of these have been alleviated with "paradigms" and "frameworks" such as MapReduce and Hadoop (L'Heureux, Grolinger, Elyamany, & Capretz, 2017)

Data Mining

Data mining can be defined as "A set of techniques that extract information about data" (Galetsi, Katsaliaki, & Kumar, 2020). The most popular techniques used in data mining are "decision trees, artificial neural networks and support vector machines". In our chosen scenario, this could be used to make a prediction model for the survivability of Covid-19 based on certain conditions on may be suffering from and vaccinations status.

Data Mining has the advantage that it is a very efficient way of extracting trends from data in the sample. This means that it can be scaled up by large amount without having the processing time go up significantly which is very important in big data. As well as that, like machine learning, it is a very good way of making predictions based on the trends extracted from the data and models based upon this such as prediction models. However, Data mining is very hardware demanding and requires a large amount of it. This means that it is not very accessible.

This method's strengths rely on the fact that it is very efficient and accurate and that it is a good way of making predictions which can be relied upon to make decisions. This is very important in our scenario. However, this method isn't very accessible which can be a limitation. This could, on the other hand, be solved by outsourcing.

Summative Table

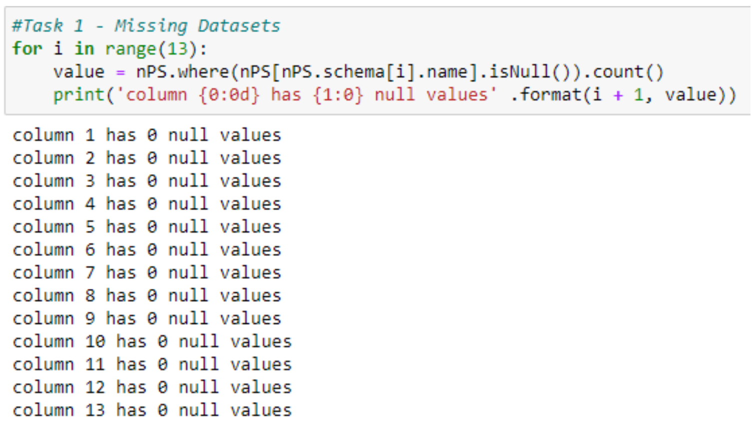
BIG DATA APPROACHES	ADVANTAGES	DISADVANTAGES
DATA MODELLING	Easy and accessible understanding of data	Conclusions can be vague or inaccurate if the trends are overly approximate
DATA VISUALIZATION	Easy to spot trends for the average person	Can be badly affected by inaccurate data fields
MACHINE LEARNING	More data means more accurate predictions made	Traditionally, ML algorithms were designed with "the assumption that entire dataset can fit in memory".
DATA MINING	Efficiency at extracting data trends Good at making predictions from data	Demanding. Requires large amounts of hardware

Task Two – Analysis

Section I: Data Summary, Understanding and Visualisation

Task 1; Missing Values:

In this dataset there was found to be no missing values from any fields in the dataset as shown in Fig 1 below. To achieve this, each column had a statement as shown in Fig 1 to count the null values in each column and display the results.



```
#Task 1 - Missing Datasets
for i in range(13):
    value = nPS.where(nPS[nPS.schema[i].name].isNull()).count()
    print('column {0:0d} has {1:0} null values' .format(i + 1, value))
```

```
column 1 has 0 null values
column 2 has 0 null values
column 3 has 0 null values
column 4 has 0 null values
column 5 has 0 null values
column 6 has 0 null values
column 7 has 0 null values
column 8 has 0 null values
column 9 has 0 null values
column 10 has 0 null values
column 11 has 0 null values
column 12 has 0 null values
column 13 has 0 null values
```

Figure 1: (Opposite)
This figure illustrates the code used to missing values within the dataset. The for-loop loops over each column and checks whether there are any null values. In the output it can be observed that there are no missing values.

As no missing data was found, this means that the dataset doesn't have to be altered before we can start processing it. In the event that there was missing data, there are a few ways to remedy this. The first way would be to insert data into the missing fields to complete them. However, this means that the extra new data in the dataset has to be accurate which is difficult in this case as, unless we take new readings which would mean a new dataset, we cannot do this accurately enough. The second approach would be to just simply delete the rows which have missing data on them. This allows us to have a complete dataset minus the empty rows. While this means that we will have less data to work with, a complete dataset is required to extract trends and make accurate predictions. We also have the approach of just leaving the empty rows as they are however this isn't a particularly good approach. They could mean that the program could run into errors as pyspark has to know what to do with these empty rows and if it doesn't, it could cause us some issues when processing the data. It could also affect the results because the classification algorithm could take account of those values as a number and incorporate them within the analysis. This could skew the analysis and lessen its accuracy for example. If I was to choose a method. I would just remove the rows that have missing values as I think that this is the most sensible approach and would give the most accurate predictions from the data.

Task 2; Summary Statistics:

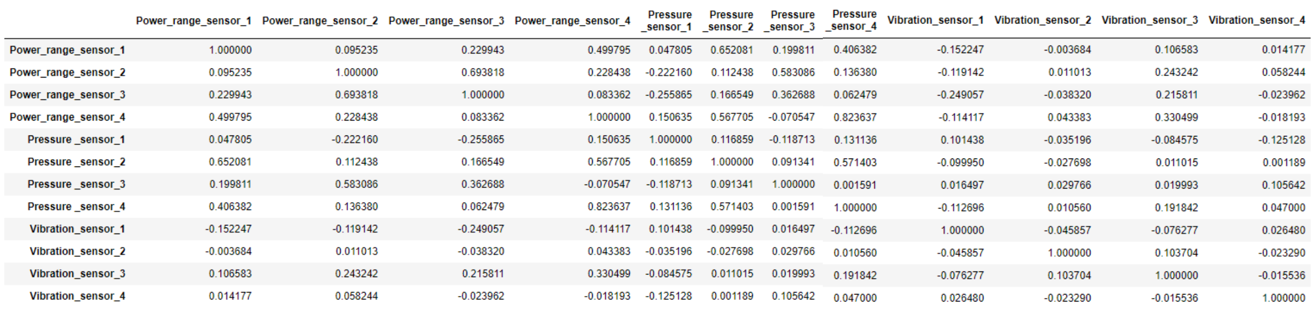
In this section, we calculated the summary statistics for each column in the dataset, this included the minimum value, maximum value, mean, median, and variance values. This was approached in a similar way to the missing value. Each column was looped over using a for loop and the statistics were calculated using a groupBy(<column>).agg(<function>). The functions that were used were from the pyspark SQL functions library and were min, max, mean, percentile_approx, and variance. The results were displayed with each column in the dataset having an individual table. In this table, each summary statistic was displayed in a column with the rows being the groups of the reactors "Normal" or "Abnormal". I chose to display it this way as this means that it is easy to compare and difference between the statistics can be easily spotted this way.

While the approach of using a for loop for looping over the columns works, it is not the most scalable method available. The much more scalable approach would be to just use one statement on the whole dataset such as "df.describe()" available in pandas. This is much more scalable and also more flexible as the loop won't need to be expanded if more columns are added or a different dataset is input. However, in our case, it only gives us some of the values that we are after meaning that it cannot be used by itself to calculate our summary statistics.

For the box plots, I decided to convert the sparky dataframe into a pandas dataframe, this is simply done with the ".toPandas" method. This allowed me to use the pandas.boxplot() functions to plot the boxplots. Each feature had an individual statement. The columns were grouped by status to have a separate boxplot for each reactor group (Normal and Abnormal). This resulted in having two boxplots for each group of a feature on the same graph allowing us to compare the distribution of data easily between the two groups and getting a feel for the trends in each group's data.

Task 3; Correlation Matrix:

The correlation matrix was produced using the pandas dataframe produces in the last task. This was done using the "df.corr()" function from the pandas library. This function produces a correlation matrix for the whole dataframe requiring only one line of code with each field representing the correlation between 2 features. This is the most scalable and efficient method as the correlation matrix would change accordingly if any changes were to be made on the dataframe it is based on or if a different dataframe altogether was to be processed. Figure 2 below shows the correlation matrix for our data.



	Power_range_sensor_1	Power_range_sensor_2	Power_range_sensor_3	Power_range_sensor_4	Pressure _sensor_1	Pressure _sensor_2	Pressure _sensor_3	Pressure _sensor_4	Vibration_sensor_1	Vibration_sensor_2	Vibration_sensor_3	Vibration_sensor_4
Power_range_sensor_1	1.000000	0.095235	0.229943	0.499795	0.047805	0.652081	0.199811	0.406382	-0.152247	-0.003684	0.106583	0.014177
Power_range_sensor_2	0.095235	1.000000	0.693818	0.228438	-0.222160	0.112438	0.583086	0.136380	-0.119142	0.011013	0.243242	0.058244
Power_range_sensor_3	0.229943	0.693818	1.000000	0.083362	-0.255865	0.166549	0.362688	0.062479	-0.249057	-0.038320	0.215811	-0.023962
Power_range_sensor_4	0.499795	0.228438	0.083362	1.000000	0.150635	0.567705	-0.070547	0.823637	-0.114117	0.043383	0.330499	-0.018193
Pressure _sensor_1	0.047805	-0.222160	-0.255865	0.150635	1.000000	0.116859	-0.118713	0.131136	0.101438	-0.035196	-0.084575	-0.125128
Pressure _sensor_2	0.652081	0.112438	0.166549	0.567705	0.116859	1.000000	0.091341	0.571403	-0.099950	-0.027698	0.011015	0.001189
Pressure _sensor_3	0.199811	0.583086	0.362688	-0.070547	-0.118713	0.091341	1.000000	0.001591	0.016497	0.029766	0.019993	0.105642
Pressure _sensor_4	0.406382	0.136380	0.062479	0.823637	0.131136	0.571403	0.001591	1.000000	-0.112696	0.010560	0.191842	0.047000
Vibration_sensor_1	-0.152247	-0.119142	-0.249057	-0.114117	0.101438	-0.099950	0.016497	-0.112696	1.000000	-0.045857	-0.076277	0.026480
Vibration_sensor_2	-0.003684	0.011013	-0.038320	0.043383	-0.035196	-0.027698	0.029766	0.010560	-0.045857	1.000000	0.103704	-0.023290
Vibration_sensor_3	0.106583	0.243242	0.215811	0.330499	-0.084575	0.011015	0.019993	0.191842	-0.076277	0.103704	1.000000	-0.015536
Vibration_sensor_4	0.014177	0.058244	-0.023962	-0.018193	-0.125128	0.001189	0.105642	0.047000	0.026480	-0.023290	-0.015536	1.000000

Figure 2 (Above):
Shows the correlation matrix for the dataframe "Nuclear Plant Small Dataset". Each field is the correlation between two features

Upon looking at the correlation of the features in the matrix, most of them seem to be fairly low, the highest correlations seem to be at the power range end of the chart. Between Power Range Sensor 2 & 3 for example, the correlation between those stands at 0.694. This could perhaps mean that we could only use one of those for our investigation perhaps without too much negative effect on the results. Pressure Sensors 2 & 4 also had a fairly high correlation at 0.571. While still high this is only just above 55% which is a bit more than half. This is too low to be able to merge the two features and treat them as one. These would be the features that make the most sense to merge as they both measure the same thing: power range in the former and pressure in the latter. The highest correlation in the table stands at 0.823 between Pressure Sensor 4 and Power Range sensor 4. These however cannot be merged and treated as just one column as they measure different things within the reactor: one measuring the power range and the other the vibrations. This however could point out the fact that more power would equate to more vibrations. Most of the other correlation tend to sit below 0.20 which isn't particularly high. Some even go into the negatives notably at the vibration end of the table showing that the vibration sensors showing that as the values of one increase, the values of the other decrease to an extent which is quite interesting. This, as said earlier could point to trends within rectors.

Section 2: Classification

Task 4; Data Split:

The dataset was split into two sets from the training and testing of the classifiers. This was done using the randomSplit() function with the coefficients 0.7 and 0.3. This leads to us having a 70% training set and a 30% test set. The total number of rows in the whole dataset was 996, this was split equally between the two groups which gives us 498 rows for each group. When the program was run and the figures were collected, we had 682 rows of data in the training set. Of those rows, we had 351 rows of the normal group and 331 rows of the abnormal group. The test set had 314 rows of data of those rows, 147 were form the normal group and 167 were of the abnormal group. These figures will change each time the algorithm is run, however. Sometimes one of the sets will have a bit more data than another time and the split of the groups will be different as well.

Task 5; Classifiers Training, Testing & Evaluation:

When training a classifier, the data that is fed into the classifier cannot be in the format that it appears in the dataframe as the classifiers will not be able to work with this format of data. The data must therefore be in a vector. As well as this, we have to consider that the status of the reactor must be indexed as the classifier will not accept a string as data it can classify as it is a string and not numerical data. Therefore, the first thing that we have to do in order to work towards training the classifiers is preparing the data. This was split into three stages

The first stage was to index the status. Each status, Normal and Abnormal was indexed with a number, 1.0 and 0.0 respectively, this was achieved by using a string indexer. In our case, this was the StringIndexer function from the pyspark machine learning feature library. The string indexer had the status column of type string as input and output a new column named statusIndex with the numbers previously stated. The second stage was to vectorize the data so that the classifier would accept it and be able to be train and tested with the data, this was achieved using the VectorAssembler function from the same library as used in the first stage. All of the column's names were put into a array and given to the vector assembler as input. This gave it all the data from the table to put into a vector. The output column was named "features" and contained one vector of data per row. These vectors of data also needed to be indexed for the classifiers. The last step was therefore to index the vectors of data so that the classifier could take it as input. This was done using the Vector Indexer function. Its input was the vector of features compiled in the previous stage. The output was in the column "featuresIndex". This would give an index to the features which is what it going to be input into the classifiers.

Now that the data has is in a format that can be used by the classifiers. We need to define the three classifiers that we will be using. The first one is a Decision tree. This is the DecisionTreeClassifier function from the pyspark.ml classification library. This only requires that data that it will be working on and will only need two inputs: the status indexes and the vector indexes. The second classifier we will be using is a support vector machine. This is the LinearSVC functions from the same library. This also requires the two inputs previously used: the status indexes and the vector indexes. As well as these two inputs, I chose to set the maximum iterations to a value of 10 and the regParam to 0.1. The last classifier we will be using is an artificial neural network. This is the MultilayerPerceptronClassifier function. As with the others this will requires the status indexes and the vector indexes. I also chose to set the layers myself. The first layer was set to the number of features present in the vector. This is important as it will not work without this. The next layers were set to 20, 10 and 2 leading us to a four-layer artificial neural network. The seed parameter was also set to 123.

Now that we have all the elements that we require we can set the pipelines. A pipeline is essentially the machine learning workflow that the program will follow with the data when training. In our case, we need to follow this workflow: index the reactor status, assemble the data vector, index the data vectors and train the classifier. This is how our pipeline will be defined for each of the classifiers. We can now train our classifiers creating a model for each using the ".fit" function on the 70% training data set from the last task. This gives us a model on which we can test the classifier using the 30% test set. This should give use some predictions made on the reactor status from the data in the form of a status index in the prediction column in the resulting new table.

We can now evaluate our classifiers with three measures. The error rate which is self-explanatory, the sensitivity also know as the "True positive rate" which informs us about "the proportions of actual positive cases predicted as positive for our model" (Malik, 2020), in our case Normal reactors and, the specificity also known as the "True Negative Rate" which informs us about "the proportions of actual negative cases predicted as negative for our model" (Malik, 2020), in our case Abnormal Reactors. The results of the evaluation can be observed in the table below.

	Error Rate	*Sensitivity*	*Specificity*
Decision Tree	0.20	0.85	0.25
Support Vector M.	0.32	0.60	0.26
Artificial Neural N.	0.28	0.71	0.28

We notice that the error rate for the classifiers sit quite low at about 20% to 30%. This means that our classifiers have performed well at classifying the data that they have been given as a test. While the classifiers have predicted well when the reactors are functioning normally, they have a very low specificity meaning that they were not doing well at predicting when the reactors were functioning abnormally.

Task 6; Classifier Comparison:

The three classifiers didn't have error rates that were very different from one another. The highest error rate was 32% from the Support Vector Machine while the lowest was the decision tree at 20%, the artificial neural network did only slightly better at 28% error rate. We can deduce that the decision tree did the best in this respect. Sensitivity wise, the clear winner is once again the decision tree at 85%. The two other methods fell quite a way behind here at 71% for the artificial neural network and 60% for the support vector machine. Lastly, on specificity, all of the methods were not great here, the artificial neural network was the best at 28% which is a very low figure while the support vector machine and the artificial neural network were close behind at 26% and 25% respectively. Ironically, the artificial neural network doing the best in the other sections did the worse here.

To conclude, the artificial neural network would be the preferred method for classification of this dataset. While it did the worse on specificity, it did the best by quite a good margin in both the error rate and sensitivity. All methods were also quite close on specificity, and all did equally terribly. Looking at the data, it is clear why the artificial neural network was chosen as the best.

Task 7; Abnormality Detection Evaluation:

Based on the analysis of this data, I think it is quite clear that the classification techniques used in this problem are not adequate to detect the status of the reactor. While they are very good at predicting a reactor that is operating normally as shows by the very high sensitivity, they are not good at detecting when a reactor is operating abnormally. All methods scored less than 30% in this respect. Given the dangerous nature of a nuclear reactor operating abnormally and the environmental concerns that it could have, they would have to have much higher scores to be used to detect abnormalities such as 70% or 80%

Task 8; MapReduce:

In this section, the pyspark dataframe was converted into a pyspark RDD (Resilient Distributed Dataset). The reason for this is because the map and reduce functions cannot be applied to dataframes and only to an RDD. The RDD was then mapped using the ".map" command.

Bibliography

Galetsi, P., Katsaliaki, K., & Kumar, S. (2020). Big Data analytics in health sector: Theorical Framework, Techniques and Prospects . *International Journal of Information Management* , 206-216.

L'Heureux, A., Grolinger, K., Elyamany, H. F., & Capretz, M. A. (2017, April 20). *Machine Learning with Big Data: Challenges and Approaches.* Retrieved from IEEE Explore: https://ieeexplore.ieee.org/abstract/document/7906512

Malik, F. (2020, February 20). *Sensitivity Vs Specificity In Data Science*. Retrieved from Medium: https://medium.com/fintechexplained/sensitivity-vs-specificity-in-data-science-2f673039dbd1

Mullins, C. S. (2021, May 3). *The Importance of Data Modeling in a Big Data World.* Retrieved from Database, Trends and Applications: https://www.dbta.com/Editorial/Think-About-It/The-Importance-of-Data-Modeling-in-a-Big-Data-World-145915.aspx