Assignment 2 – CMP3749M Big Data

Dylan Samuel Petty

19703357@students.lincoln.ac.uk

Deadline: 19/01/2023

Contents

Task 1 – Analysis of Nuclear Plants dataset (2500 words)

1.1 – Dataset groups:

Upon being given some datasets to analyse for this assignment, we must first consider the integrity and validity of the data. This process ensures that there are no missing values or duplicates from erroneous input before planning out data insights.

Below you can see the code implemented to irradicate any None, NULL or NaN values from the dataset so that we can delve deeper into this task.

Searching for Null values in the dataset:

```
smlNucplant2 = smlNucplant.select([count(when(col(c).contains('None') | \
                        col(c).contains('NULL') | \
                        (col(c) == '' ) | \
                        col(c).isNull() | \
                        isnan(c), c
                       )).alias(c)
                   for c in smlNucplant.columns])
smlNucplant2.show()
```

```
+------+--------------------+--------------------+---------------------+--------------------+------------------+------------------+------------------+------------------+------------------+------------------+------------------+------------------+
|Status|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|
+------+--------------------+--------------------+---------------------+--------------------+------------------+------------------+------------------+------------------+------------------+------------------+------------------+------------------+
|     0|                   0|                   0|                    0|                   0|                 0|                 0|                 0|                 0|                 0|                 0|                 0|                 0|
+------+--------------------+--------------------+---------------------+--------------------+------------------+------------------+------------------+------------------+------------------+------------------+------------------+------------------+
```

As shown underneath the code box, the results show the number of erroneous results per column; evidently, there is no missing data in the datasets given. Therefore, the methods to dealing with this issue will be discussed throughout.

What is missing data:

Missing data is when a value, for whatever reason, is not present in the dataset where it can be assumed that it should be. This can offer occur due to human error in cases such as:

-Data has been omitted by the user and has not been given.

-Data has been deleted or manipulated incorrectly which can lead to missing values or data corruption.

-Data cleaning method not covering all types of invalid datatypes; thus, they remain among justified data.

-Data entry is empty

-A data breach has occurred and unauthorised access to the data will result in unwanted effects; data has been tampered with. (Irwin, L. 2022)

All of these are prevalent issues faced in data science when creating insights and predictions into the future and use cases of the datasets, especially so when human error is involved as this is the cause of a good portion of mishaps in maintaining and securing large scale dataframes. Nonetype values are represented differently depending on the library that manipulates the dataframe. For example, in the Pandas dataframe, Nonetype values are represented as “NaN” which stands for “Not a Number”, or in a PySpark dataframe these values are shown as “Null”. These erroneous results can be filtered out using the loop shown in the figure above. The code above also filters out any other unwanted values like “None” or empty entries as in evaluation, these entries have no weight at all and could throw off a comparison. (Tamboli, N. 2021. 1-3)

As for why the data is missing; there are various patterns that have been studied of where and why data has been verified as invalid. These include topologies such as Missing At Random (MAR), Missing Not At Random (MNAR) and Missing Completely At Random (MCAR). (Tamboli, N. 2021, 4)

Beneath some case studies will be discussed and give some insight into how each pattern may occur.

MAR – Data that is Missing At Random is a pattern in which the data is conditionally missing when the probability of the data missing depends on the data that is observed. for example, it is not completely missing at random, but it is related to other variables in the given dataset. Another example of this pattern would be that we can analyse the where and why the data is missing, hypothetically in the context of the nuclear power plant dataset, if the data were more likely to be missing based on the Status being abnormal than normal, then we can classify this as Missing At Random based on this condition. (Tamboli, N. 2021, 6)

MNAR – Data which is classified as Missing Not At Random is often a sign that the data that is missing, has been made deliberately so. This could range from the user cleaning out the database based on a condition, to the data entry having omitted fields. An example of data cleaning would be that if there were too many missing data entries, then there are two methods which will determine the next steps. These being either deleting the columns or rows where there are many missing entries or using data imputation to predict a value to substitute for the empty field. In order to classify data as MNAR, we must use different methods to ensure that there is no underlying condition for the missing data. These methods include little's MCAR test, a sensitivity analysis and even a visual inspection in some cases. A sensitivity analysis runs and algorithm that tests the rate of missingness against conclusions of statistical analysis in order to detect any kind of bias between relationships of variables in correlation to given conclusions. In our context, the decision to remove or predict data is very dangerous because formulating data that could determine the coolant dispersion rate of a nuclear power plant could lead to a complete meltdown if a prediction is even slightly wrong, very real proof of this concept would be the Chernobyl nuclear power plant disaster; although not the direct cause, very likely to be a contributing factor. (Tamboli, N. 2021, 7)

MCAR – Data that is Missing Completely At Random is missing data that has no linkable relationship to other variables. A sure-fire way to determine this is through Little's MCAR test, only after all other variable relationships have been ruled out as this method will assume there are no connections or correlations. This method is especially good at confirming that the missing data is actually Missing Completely At Random, because it tests the null hypothesis, runs the data through a complex algorithm and returns a p-value; if the value is above 0.05 this usually means that the data has some underlying relationship that is not MCAR. To put this into context, if data is missing as such then it shows that there is a major fault somewhere in the system and immediate action must be taken or very real consequences may occur.(Tamboli, N. 2021, 5)

1.2 – Splitting Dataframe by Status:

To correctly prepare this dataset, it is important that we separate the values into the two groups of subjects: Normal and Abnormal. This is a necessary step before we can properly analyse and isolate relevant data to use in our evaluation. This is especially the case with a nuclear power plant as the representation of "Normal" values must show clear differences otherwise temperature readings are potentially random which are likely to cause real world consequences.

Status	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	Vibrat
Normal	4.5044	0.7443	6.34	1.9052	29.5315	0.8647	2.2044	6.048	14.4659	
Normal	4.4284	0.9073	5.6433	1.6232	27.5032	1.4704	1.9929	5.9856	20.8356	
Normal	4.5291	1.0199	6.113	1.0565	26.4271	1.9247	1.942	6.7162	5.3358	
Normal	5.1727	1.0007	7.8589	0.2765	25.1576	2.609	2.9234	6.7485	1.9017	
Normal	5.2258	0.6125	7.9504	0.1547	24.0765	3.2113	4.4563	5.8411	0.5077	

only showing top 5 rows

Status	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	Vibr
Abnormal	8.4467	5.8667	9.1736	11.6904	1.1803	1.4415	4.3256	6.4249	8.3453	
Abnormal	7.7426	5.7504	9.4666	11.3361	2.0382	1.2612	6.1856	6.8098	8.9823	
Abnormal	7.7426	5.6149	10.2393	11.2692	5.2864	1.5143	6.7682	6.3428	7.4222	
Abnormal	7.3059	5.7349	10.2228	11.4734	9.4818	1.8207	6.8006	5.5452	7.2652	
Abnormal	5.874	5.9592	9.1791	11.1758	10.2951	1.4392	8.0166	4.7337	3.9511	

only showing top 5 rows

Figure 1. dataset splits by status

As seen in the figure above, the data has been split by the status column before the evaluation has started.

	Power_range_sensor_1	Power_range_sensor_2	Power_range_sensor_3
min	0.085100	0.040300	4.382600
max	12.129800	11.928400	14.098200
mean	5.602453	6.844503	9.292054
50%	5.178650	6.717650	9.262850

	Power_range_sensor_4	Pressure _sensor_1	Pressure _sensor_2
min	0.154700	0.024800	0.010400
max	16.356800	56.856200	9.221200
mean	8.701398	13.797526	3.415646
50%	9.240850	10.634800	3.113000

	Pressure _sensor_3	Pressure _sensor_4	Vibration_sensor_1
min	0.077400	0.00580	0.009200
max	12.647500	15.10850	31.498100
mean	5.923353	5.58618	8.441437
50%	5.739400	4.25915	7.449900

	Vibration_sensor_2	Vibration_sensor_3	Vibration_sensor_4
min	0.027700	0.064600	0.083100
max	34.867600	53.238400	43.231400
mean	9.699616	19.437804	10.925098
50%	8.700750	16.464500	9.485450

Figure 2. Normal Status stats

	Power_range_sensor_1	Power_range_sensor_2	Power_range_sensor_3
min	0.008200	0.389100	2.583966
max	10.923078	10.154100	15.759900
mean	4.396695	5.914043	9.164170
50%	4.513550	5.932218	9.472050

	Power_range_sensor_4	Pressure _sensor_1	Pressure _sensor_2
min	0.062300	0.131478	0.008262
max	17.235858	67.979400	10.242738
mean	6.009146	14.600728	2.740270
50%	5.399300	12.596150	2.382689

	Pressure _sensor_3	Pressure _sensor_4	Vibration_sensor_1
min	0.001224	0.029478	0.000000
max	11.772400	16.555620	36.186438
mean	5.575115	4.407824	7.887689
50%	5.744257	3.322575	6.535950

	Vibration_sensor_2	Vibration_sensor_3	Vibration_sensor_4
min	0.018500	0.131784	0.009200
max	34.331466	36.911454	26.466900
mean	10.303570	10.938159	8.942085
50%	8.973100	8.987269	8.137600

Figure 3. Abnormal Status stats

As seen in Figures 2 and 3, the minimum, maximum, mean and median values have all been extracted from each column separately as a step of pre-processing the dataset for our analysis.

These values are necessary in order to create boxplots for each data split as infographics help the viewer to better understand the correlations between datasets better than just visually scanning the numbers individually.

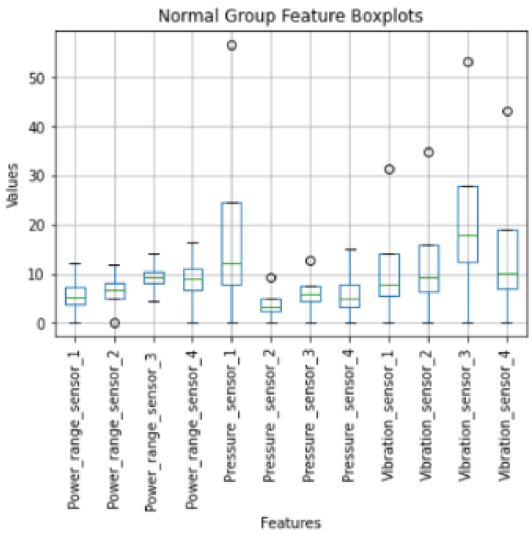

Figure 4. Normal Status Boxplots

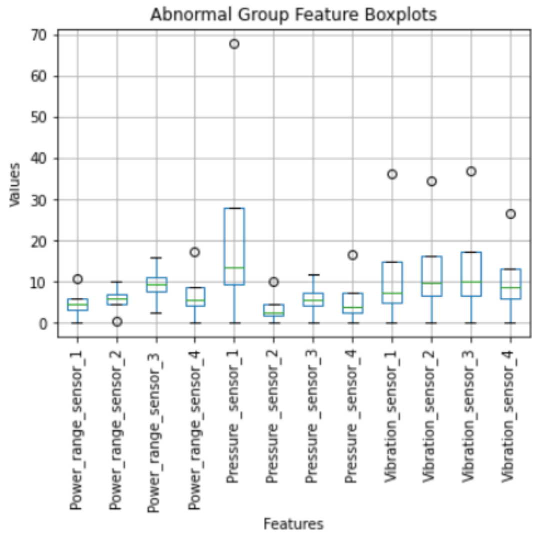

Figure 5. Abnormal Status Boxplots

In both figures 4 and 5, the split datasets have been formatted into boxplots so that the stakeholder may better understand the niche differences between the statuses. Something to note is that although the graphs look similar, the scales very slightly differ; the abnormal group has as maximum value difference of roughly 10 for Pressure_sensor_1. Even though the pattern of the results looks similar, the abnormal results display slight differences in value whether above the norm or beneath, hence the Abnormal status.

1.3 – Correlation Matrix:

A correlation matrix is an effective way to measure the relationship or likelihood of two different variables, this is done by comparing a dataset's fields to one another and returning a value between 1 and -1 based on the similarities. This is calculated by plotting all the points of the dataset on a graph and drawing a line of best fit through the points; the closer the points to the line the stronger the correlation, also the direction of the line is important; upward facing means positive, the inverse is also true for both factors. Although Correlation does not imply causation, the linear dependence of each variable must be further studied before drawing any conclusions about their relationships. A good example of this conundrum is "women who are more educated tend to have lesser children. Women who are less educated tend to have more children, it's a general observation. If you look at the population of developed and under-developed countries and look at their national education index, the two seem to be correlated but we can't say education makes you produce lesser babies." Thus correlation is a better suggestion than an ultimatum. (M, Krishnan, website)

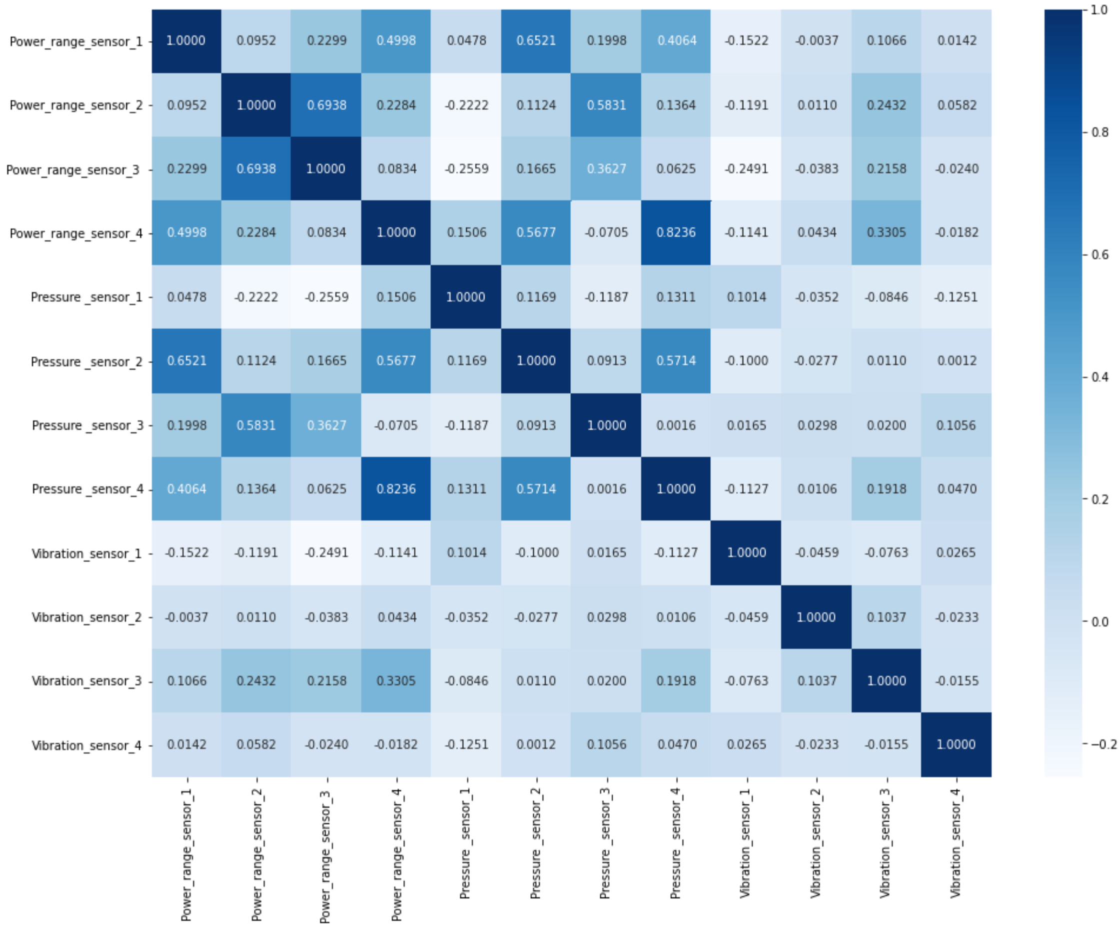

Figure 6. Correlation Matrix with Seaborn heatmap applied

1.4 – Train and test data splits

In order to train machine learning models, the data must be split into training and testing datasets so that the algorithm can be trained to predict or measure something, then the test dataset can be used for the evaluation of the model; accuracy of the measurement and various other metrics can be tested as a result of this.

While the 70:30 split is often a viable choice for a lot of machine learning models, there will be cases where the datasets will need to be split differently as discrepancies in dataset sizes might cause overfitting or underfitting, causing huge miscalculations in model predictions and evaluation. Typically, as the dataset gets bigger, the split of testing data gets smaller as the amount of testing and validation data doesn't scale directly with the volume of the dataset. The most common split in practice is a 80/20 split as this follows the Pareto principle which states that "for many outcomes, roughly 80% of consequences come from 20% of causes" (Chappelow, J. (n.d.)), meaning that 80% of the dataset can be applied to testing and the other 20% for testing for a generally good result with low variance; this leaves a decent segment of the data still to test the model on unseen data to truly evaluate the models metrics. (Asana. 2021)

```
#train test split 70%:30%
randOrder = dataIndexed.orderBy(rand())

##assign train and test sets by shuffled order
[randSplitTraining, randSplitTesting] = randOrder.randomSplit([0.7,0.3], seed=1234)

print(randSplitTraining.count())
print(randSplitTesting.count())

701
295
```

Figure 7. Train and Test data split count

As seen in figure 7 above, a 70% training to 30% testing data split has been employed, this is a safe bet as the dataset that's being analysed is relatively small in terms of big data scale, otherwise it is safe to assume that the 80% to 20% split would suffice as explained above.

1.5 – Model training and evaluation

In the evaluation of this dataset, 3 separate machine learning models were trained and evaluated against 3 performance metrics; A decision tree, Support Vector Machine and an Artificial Neural Network were tested for error rate, sensitivity and specificity.

It is important to understand how these models work so these will be briefly discussed beneath:

A Decision Tree is a computationally heavy model as it will search through the dataset, figure out the best values to split by based on information gain, create a node and split the data directly between the found value. The model will store all the leaf nodes and logic for each, traversing the entire tree for even the smallest of decisions. Despite this, the model is especially good for finding and working with datasets with exceedingly complex relationships or large volume datasets. (Quinlan, J.R. 1996. 71 – 72)

A Support Vector Machine is a supervised learning algorithm that maps the input data into a "feature space" where patterns of the data, such as linear separability are tested the SVM will calculate the maximally effective boundary to split the data by a support vector that shares value with the boundary. Once a boundary has been established then data classification can begin by filtering data based on the side of the boundary it lies. SVMs are especially good at dealing with lots of features or variables, even when the dataset's lengths are imbalanced. (Pradhan, A. 2012. 83-84)

An Artificial Neural Network is comprised of layers of interconnected nodes, referred to as neurons, transmit data to one another from an input layer, through a pre-determined number of hidden layers. When each layer receives the input data, it is transformed and tested through some mathematical activation functions, which are influenced by the weighting of given parameters. Finally, these weights are used to optimise the models performance using an algorithm called gradient descent which works to minimise the difference between predicted and ground truth values. (Jerez et al. 2010 3.3.1)

```
Confusion Matrix:
 [[132  27]
 [ 15 121]]

pred accuracy:  0.8576271186440678
-------------------------------

 Error Rate (to 5d.p.):  0.14237

 Sensitivity (to 5d.p.):  0.83019

 Specificity (to 5d.p.):  0.88971
```

Figure 8. Decision Tree metrics

```
Confusion Matrix:
 [[125  34]
 [ 28 108]]

pred accuracy:  0.7898305084745763
--------------------------------

 Error Rate (to 5d.p.):  0.21017

 Sensitivity (to 5d.p.):  0.78616

 Specificity (to 5d.p.):  0.79412
```

Figure 9. SVM metrics

```
Confusion Matrix:
 [[140  19]
 [ 36 100]]

pred accuracy:  0.8135593220338984
--------------------------------

 Error Rate (to 5d.p.):  0.18644

 Sensitivity (to 5d.p.):  0.8805

 Specificity (to 5d.p.):  0.73529
```

Figure 10. ANN metrics

As seen in Figures 8-10, the metrics tested have varying results, especially the Confusion Matrix seen atop each figure. These are structured to show the True positive, False Positive, False Negative and True Negative values from the testing data split on the trained classification model. (Visa, S. 2011. 121) Error rate is the comparison of all false values against all true values (Visa,S. 2011. 121), sensitivity is the comparison of true positive over actual positive values and finally specificity is the comparison of true negative results against actual negative results. (Marino et al. 2013)

Ultimately, the decision tree has the better prediciton accuracy, specificity and lowest error rate, making this the most accurate model for this context. The dataset has a correct prediction rate of over 75% making it valid for machine learning classifying of the statuses Normal or Adnormal entries from the power plant.

Task 2 – MapReduce for Margie Travel dataset

The analysis of the datasets provided by Margie Travel has been undertaken with the use of Hadoop's MapReduce programming model, which excels in manipulating large datasets in parallel using distributed processing systems. MapReduce works by splitting the dataset into smaller chunks based on a pair of decisive values; the map function is called upon every record to generate a key value pair prepare the data in the correct format for the Reduce function in the next step. The Mapping stage also shuffles and sorts the mapped pairs based on the similarity between keys (Mastering Hadoop page 31). for an example see Figure 11's heading "Sort" and "Merge" sections for a visual representation.

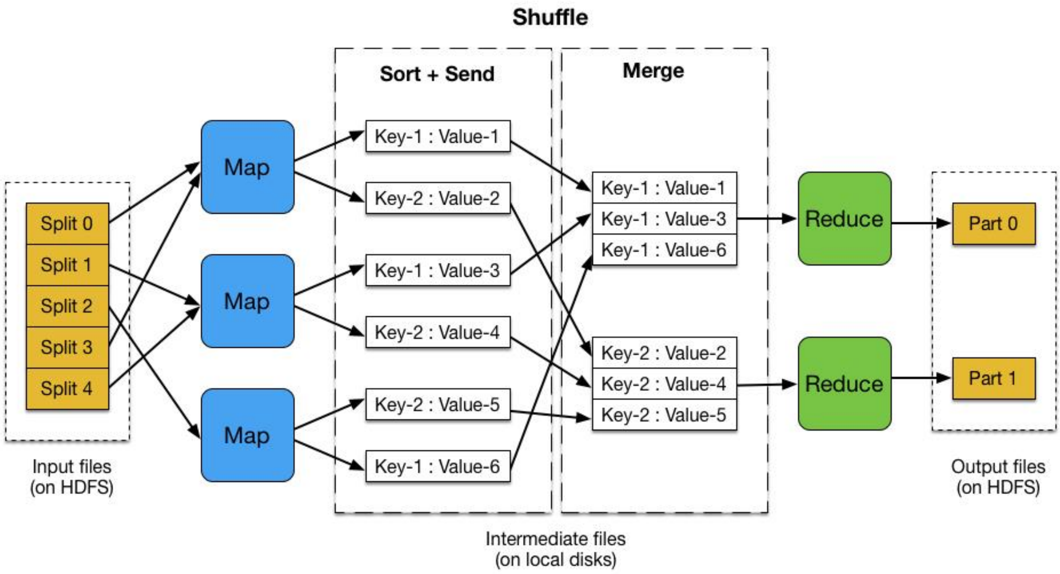

Figure 11. MapReduce example from Sunlab (www.sunlab.org)

The Reduce stage of the MapReduce methodology is where all the sorted and shuffled chunks are merged into one singular map for output, to summarise all the results into a reusable RDD variable. The number of nodes required for a for a reduce task is determined by a parameter fed into the function, which by default is 1 if not specified, which runs a great risk of overloading a particular node and causing data spill (mastering Hadoop page 50).

It is also important to note that we are using the AComp_Passenger_data_no_error dataset rather than the original file, AComp_Passenger_data. Although data cleaning is not out of the scope, we will are going to use the two files independently to make the comparison of filtered against unfiltered datasets for the analyses. In the uncleaned dataset there are various erroneous entries containing either Null values or entirely empty entries, the invalid entries can be either removed entirely or we could potentially use data imputation to predict the missing values if only a couple fields are empty. Another issue that should be noted is that a portion of the fields have either incorrect values or the data hasn't been formatted correctly for the field. For example, there are some airport codes that have symbols in, entry number 204's airport departure code is "A;S", wherein symbols are invalid or certainly unexpected for the formatting of an airport code. In this case, the data must be verified by the airport before evaluating further as running any correlation algorithms will yield incorrect results; erroneous entries will throw off prediction weighting and decrease true accuracy greatly.

The Datetime datasets results match up exactly with the results I gained from the conversion and calculation of datetime as seen in 2.2, the only difference in the results is that the formatting of the values is different; my results are formatted to include the date.

2.1 – Airport Usage.

As seen in Figure 12, It has been discovered that the Denver airport, code DEN, has accumulated the highest number of flights departing from it, the next highest being 5 different airports at 2 flights each. It is important that we can conclude these things because after data cleaning this dataset can be effectively used to make predictions and potentially map out insights for future usage.

```
Airport DEN has appeared 3 times
Airport JFK has appeared 1 times
Airport ORD has appeared 2 times
Airport KUL has appeared 2 times
Airport MAD has appeared 1 times
Airport LHR has appeared 1 times
Airport CGK has appeared 2 times
Airport MUC has appeared 1 times
Airport AMS has appeared 1 times
Airport DFW has appeared 1 times
Airport MIA has appeared 1 times
Airport CDG has appeared 1 times
Airport CAN has appeared 2 times
Airport IAH has appeared 2 times
Airport LAS has appeared 1 times
Airport CLT has appeared 1 times
Airport ATL has appeared 2 times
Airport PVG has appeared 1 times
Airport FCO has appeared 1 times
Airport BKK has appeared 1 times
Airport PEK has appeared 1 times
Airport HND has appeared 1 times
Below is a list of airports that have no flights:
['LAX', 'FRA', 'HKG', 'DXB', 'SIN', 'SFO', 'PHX', 'IST']
```

Figure 12. Airport usage.

Another segment to this investigation was to collect a list of all airports that have no departures in this dataset, at the bottom of Figure 12 you can see the list of the codes not used.

2.2 – Passengers per flight and flight times.

```
Flight SQU6245R departs from DEN at 17:14, arriving at FRA at 10:43, with 21 passengers.
Flight XXQ4064B departs from JFK at 17:05, arriving at FRA at 06:27, with 25 passengers.
Flight SOH3431A departs from ORD at 17:00, arriving at MIA at 21:10, with 18 passengers.
Flight PME8178S departs from DEN at 17:13, arriving at PEK at 15:15, with 18 passengers.
Flight MBA8071P departs from KUL at 17:04, arriving at PEK at 02:36, with 16 passengers.
Flight MOO1786A departs from MAD at 16:56, arriving at FRA at 20:00, with 13 passengers.
Flight HUR0974O departs from DEN at 17:15, arriving at PVG at 16:33, with 7 passengers.
Flight GMO5938W departs from LHR at 17:11, arriving at PEK at 10:48, with 25 passengers.
Flight DAU2617A departs from CGK at 17:23, arriving at SFO at 23:34, with 12 passengers.
Flight RUM0422W departs from MUC at 16:58, arriving at MAD at 20:12, with 14 passengers.
Flight ATT7791R departs from AMS at 17:13, arriving at DEN at 09:54, with 15 passengers.
Flight WPW9201U departs from DFW at 17:21, arriving at PEK at 17:33, with 11 passengers.
Flight DKZ3042O departs from MIA at 17:05, arriving at SFO at 02:03, with 11 passengers.
Flight QHU1140O departs from CDG at 17:14, arriving at LAS at 12:07, with 21 passengers.
Flight ULZ8130D departs from CAN at 17:23, arriving at DFW at 21:26, with 27 passengers.
Flight VYU9214I departs from ORD at 17:18, arriving at DXB at 18:28, with 15 passengers.
Flight HZT2506M departs from IAH at 17:12, arriving at AMS at 10:36, with 14 passengers.
Flight EWH6301Y departs from CAN at 17:22, arriving at DFW at 21:25, with 10 passengers.
Flight VYW5940P departs from LAS at 17:26, arriving at SIN at 00:09, with 17 passengers.
Flight WSK1289Z departs from CLT at 16:59, arriving at DEN at 21:37, with 21 passengers.
Flight TMV7633W departs from CGK at 17:05, arriving at DXB at 07:14, with 15 passengers.
Flight FYL5866L departs from ATL at 17:25, arriving at HKG at 22:36, with 20 passengers.
Flight BER7172M departs from KUL at 17:26, arriving at LAS at 00:14, with 17 passengers.
Flight JVY9791G departs from PVG at 17:16, arriving at FCO at 13:05, with 20 passengers.
Flight VDC9164W departs from FCO at 17:18, arriving at LAS at 14:34, with 15 passengers.
Flight KJR6646J departs from IAH at 17:26, arriving at BKK at 01:34, with 23 passengers.
Flight YZO4444S departs from BKK at 17:28, arriving at MIA at 03:15, with 17 passengers.
Flight XIL3623J departs from PEK at 17:13, arriving at LAX at 14:55, with 13 passengers.
Flight RPG3351U departs from HND at 16:59, arriving at CAN at 23:13, with 13 passengers.
Flight XOY7948U departs from ATL at 17:07, arriving at LHR at 07:44, with 16 passengers.
```

Figure 13. Full flight information

In order of the flights first occurring in the dataset, Figure 13 shows a variety of information found from this investigation, including the departure and arrival times of each flight and the number of passengers on each flight. This was achieved by using map to collate all entries in the dataset by taking all columns necessary and reducing by the key of each key value pair to count all instances of the same flight IDs in parallel with one another. To get the timestamps for each flight the Unix epoch code and unformatted integer were passed through custom functions to calculate both instances of time shown in this investigation.

2.3 – Total Nautical Miles travelled

```
Passenger UES9151GS5 accumulated 132026.8516084164 nautical miles
Passenger BWI0520BG6 accumulated 124877.43077719095 nautical miles
Passenger DAZ3029XA0 accumulated 123221.24192152942 nautical miles
Passenger SPR4484HA6 accumulated 122395.49033551376 nautical miles
Passenger PUD8209OG3 accumulated 115942.80142856004 nautical miles
Passenger WBE6935NU3 accumulated 99270.6931500715 nautical miles
Passenger HCA3158QA6 accumulated 97102.59986851209 nautical miles
Passenger WYU2010YH8 accumulated 96844.57147002101 nautical miles
Passenger JJM4724RF7 accumulated 93147.21911000309 nautical miles
Passenger CKZ3132BR4 accumulated 92832.79793950269 nautical miles
Passenger EZC9678QI6 accumulated 89419.94124146449 nautical miles
Passenger LLZ3798PE3 accumulated 84190.57755153331 nautical miles
Passenger HGO4350KK1 accumulated 81887.9219352002 nautical miles
Passenger POP2875LH3 accumulated 81122.66332327103 nautical miles
Passenger CXN7304ER2 accumulated 78832.6331209729 nautical miles
Passenger YMH6360YP0 accumulated 76343.53544752272 nautical miles
Passenger VZY2993ME1 accumulated 73773.0084819783 nautical miles
Passenger EDV2089LK5 accumulated 70509.91998052782 nautical miles
Passenger JBE2302VO4 accumulated 69079.99332739634 nautical miles
Passenger SJD8775RZ4 accumulated 67526.64993984473 nautical miles
Passenger XFG5747ZT9 accumulated 66495.08874440922 nautical miles
Passenger CDC0302NN5 accumulated 63183.804415471124 nautical miles
Passenger MXU9187YC7 accumulated 61108.84127858806 nautical miles
Passenger WTC9125IE5 accumulated 59677.48335596712 nautical miles
Passenger KKP5277HZ7 accumulated 58621.720671994444 nautical miles
Passenger ONL0812DH1 accumulated 54287.14124382211 nautical miles
Passenger CYJ0225CH1 accumulated 54253.08085093294 nautical miles
Passenger IEG9308EA5 accumulated 42062.85490750052 nautical miles
Passenger PIT2755XC1 accumulated 36117.04632723889 nautical miles
Passenger PAJ3974RK1 accumulated 34267.70881103935 nautical miles
Passenger UMH6360YP0 accumulated 3344.994467200869 nautical miles
```

Figure 14. User Distance ranking in descending order.

For the final evaluation of the Margie Travel dataset, we were tasked with calculating the total nautical miles travelled by each passenger and rank them in descending order so the top passenger can be identified. This is a useful metric to evaluate as in the real world it could be used to offer loyalty discounts to the most loyal customers or compare this to other metrics and figure out what makes someone want to travel more or scale up their ticket.

References:

- Asana (2021). Understanding the pareto principle (the 80/20 rule) • asana. [online] Asana. Available at: https://asana.com/resources/pareto-principle-80-20-rule.

- Chappelow, J. (n.d.). Pareto Principle Definition. [online] Investopedia. Available at: https://www.investopedia.com/terms/p/paretoprinciple.asp#:~:text=The%20Pareto%20Principle%2C%20named%20after.

-Irwin, L. (2022). Human Error is Responsible for 82% of Data Breaches. [online] GRC eLearning Blog. Available at: https://www.grcelearning.com/blog/human-error-is-responsible-for-85-of-data-breaches#:~:text=The%20employee%20might%20miss%20a [Accessed 18 Jan. 2023].

-Jerez, J.M., Molina, I., García-Laencina, P.J., Alba, E., Ribelles, N., Martín, M. and Franco, L. (2010). Missing data imputation using statistical and machine learning methods in a real breast cancer problem. Artificial Intelligence in Medicine, [online] 50(2), pp.105–115. [Accessed 18 Jan. 2023].

- Marino, M., Li, Y., Rueschman, M.N., Winkelman, J.W., Ellenbogen, J.M., Solet, J.M., Dulin, H., Berkman, L.F. and Buxton, O.M., 2013. Measuring sleep: accuracy, sensitivity, and specificity of wrist actigraphy compared to polysomnography. Sleep, 36(11), pp.1747-1755.

- Muthu Krishnan T, (n.d.). Understanding Correlations and Correlation Matrix – Muthukrishnan. [online] Available at: https://muthu.co/understanding-correlations-and-correlation-matrix/ (Accessed: 18 January 2023).

- Pradhan, A., 2012. Support vector machine-a survey. International Journal of Emerging Technology and Advanced Engineering, 2(8), pp.82-85.

-Sandeep Karanth (2014) Mastering Hadoop. Birmingham, England: Packt Publishing (Community Experience Distilled). Available at: https://search-ebscohost-com.proxy.library.lincoln.ac.uk/login.aspx?direct=true&db=nlebk&AN=934162&site=ehost-live (Accessed: 18 January 2023).

-Tamboli, N. (2021). Tackling Missing Value in Dataset. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/10/handling-missing-value/[Accessed 18 Jan. 2023].

- Visa, S., Ramsay, B., Ralescu, A.L. and Van Der Knaap, E., 2011. Confusion matrix-based feature selection. MAICS, 710(1), pp.120-127.

-www.sunlab.org. (n.d.). CSE 6250 Big Data for Healthcare | MapReduce Basics. [online] Available at: https://www.sunlab.org/teaching/cse6250/fall2019/hadoop/mapreduce-basic.html#mapreduce [Accessed 18 Jan. 2023].