# CA4010

## Data Mining and Data Warehousing Report

Group 48
Sam Wood
and
David Craig

Student numbers
Samuel Wood: 16476066
David Craig: 14506617

## *Introduction*

A tsunami or a seismic wave is when a body of water is generated into a number of waves caused by the movement of a bigger volume of water, typically in an ocean and sometimes in a lake. These seismic waves can be caused by earthquakes, volcano eruptions, landslides and sometimes Meteorologically. Given the vast amounts of information and data on the internet on this topic we felt it was appropriate to acquire a dataset on past tsunamis and use it to make predictions for our Data Mining and Data Warehousing module.

## *Project idea and dataset description*

For this module our aim was to make predictions associated with past tsunami occurrences. It dawned on us that there was a number of things we could potentially predict. For example Location, Cause of the tsunami, how ferocious it was, number of deaths because of the incident, etc. With the use of kaggle we acquired a dataset that was of past tsunamis that went back as far as the year 2000 BC. It was obvious to some of us that some of these attributes would not be included in our prediction. To make these predictions we would have to study past trends for these tsunamis. The first steps of this would of involved Identifying central tendencies in our workshop and also classification and other analysis methods.

Initially from kaggle there were two data sets, waves.csv and sources.csv. In the waves.csv dataset there were many attributes that we felt would not play a part in our prediction for example houses damaged, fatalities and injuries. It was obvious to us that some of these attributes were not going to be useful for us when we were making our prediction. Out of all the attributes the ones that we felt would help us the most were Year, Location, Country, Cause, Magnitude, Validity and Intensity. The magnitude is the amount of energy released at the tsunamis cause. Validity is a numerical value ranging from -1 to 4, -1 being an almost erroneous entry and 4 being a definite tsunami. Intensity is defined I=log2((2**0.5)*h) where h is the maximum runup height of the wave.

The wave.csv file didn’t seem to carry a magnitude attribute and their were quite a lot of notable inconsistencies with the table, for this reason we decided to move forward with the project using the sources Dataset.

# *Data Preparation*

It was obvious to us from the beginning that the data set we acquired initially contained attributes that we would not find entirely useful. Also, given that the Location and Country classes both had over 100 different attributes, we noticed problems may show later in the project when predicting.

As previously stated, our dataset sources.csv had many classes that we were not interested in. Examples include ‘House Destruction Total’, ‘All Damage in Millions’, ‘House Damage Estimate’ and ‘Warning Status’. We knew that this data would of had to be cleaned. The dataset had approximately 45 classes but for the first stage of our project, which was our workshop. Only the use of 7 or 8 of these attributes interested us. The classes we had kept for our workshop were

- Year (From 2000 BC to 2017)
- The tsunami’s Cause (Earthquake, Landslide, Meteorological, Volcanic, Earthquake and Landslides together and Unknown)
- The Validity ( Values ranging on a scale from -1 to 4 where 3 would of been classed as a probable tsunami and 4 as an absolute etc.)
- Primary Magnitude if caused by an Earthquake ( A scale of values of 0.0 to 9.9 of the energy released at the source of the Earthquakes collision)
- Location (Country, State, Province or Island)
- Country
- Tsunami Intensity ( Values ranging on a scale from -5 to 10 defined by I = log2(2**0.5 * h) where h is the runup height of the wave.

We decided to include the Country and Location attributes as were not sure which would be used or what we would do with predicting one of them. To remove the unwanted attributes we used the python pandas library and we wrote the below python script. The “inplace” function does remove the data from the actual csv file but we had saved another copy of the dataset should we want to refer back to our missing attributes.

```
8  tsunami_data = pd.read_csv('sources.csv')
9
10 tsunami_data.drop(["STATE/PROVINCE","MONTH", "DAY", "HOUR", "MINUTE", "FOCAL_DEPTH", "REGION_CODE", "STATE/PROVINCE", "LATITUDE", "
       LONGITUDE", "MAXIMUM_HEIGHT", "MAGNITUDE_ABE", "MAGNITUDE_IIDA", "WARNING_STATUS","MISSING", "INJURIES" , "INJURY_ESTIMATE", "
       FATALITIES", "FATALITY_ESTIMATE", "DAMAGE_MILLIONS_DOLLARS", "DAMAGE_ESTIMATE", "HOUSES_DAMAGED", "HOUSE_DAMAGE_ESTIMATE", "
       HOUSES_DESTROYED", "HOUSE_DESTRUCTION_ESTIMATE", "ALL_MISSING", "MISSING_TOTAL", "ALL_INJURIES", "INJURY_TOTAL", "
       ALL_FATALITIES", "FATALITY_TOTAL", "ALL_DAMAGE_MILLIONS", "DAMAGE_TOTAL", "ALL_HOUSES_DAMAGED", "HOUSE_DAMAGE_TOTAL", "
       ALL_HOUSES_DESTROYED", "HOUSE_DESTRUCTION_TOTAL"], axis = 1, inplace=True)
11
```

*Fig 1.1 Drop function for removing classes*

In relation to the prediction attribute problem, we thought to narrow the number of attributes in the class we were predicting. To do this we constructed a new class for Continents. The attributes in this class were Antarctica, Australia (commonly referred to as Oceania), North America, South America, Asia, Africa, Europe. Given that we were told to be predicting from a pool of roughly ten attributes, we were content with this change to our dataset. The below was our dataset up to the point of our workshop. We had made a copy of the original dataset and also the waves dataset in case we had decided to use them at a later date for further study.

| YEAR | CAUSE | VALIDITY | PRIMARY_ | COUNTRY | CONTINEI | LOCATION | INTENSITY_SOLOVIEV |
|---|---|---|---|---|---|---|---|
| -58 | 1 | 2 | 6.6 | ALBANIA | EUROPE | SYRIAN CO | 6 |
| 346 | 1 | 1 | 6.8 | ALBANIA | EUROPE | THERA ISL | 6 |
| 1273 | 1 | 2 | 6.5 | ALBANIA | EUROPE | SYRIAN CO | 6 |
| 1833 | 1 | 2 | 6.4 | ALBANIA | EUROPE | IONIAN CO | 5 |
| 1851 | 1 | 2 | 6.6 | ALBANIA | EUROPE | ISRAEL AN | 3 |
| 1866 | 1 | 2 | 6.5 | ALBANIA | EUROPE | LEBANON | 6 |
| 1866 | 0 | 3 | | ALBANIA | EUROPE | LEBANON | 3 |

*Fig 1.2*

## *Identifying Central Tendencies*

The workshop we were allocated into for this project was on identifying central tendencies in our dataset ie. mean, median and mode values for the attributes. Given that identifying trends was included in our approach this workshop was well suited for us. Firstly we wanted to examine which location had tsunami occurrences most frequently. The top five countries for tsunamis were Japan, USA, Indonesia, Russia and Chile. Of course this means that Asia was the mode continent for tsunami occurrences.

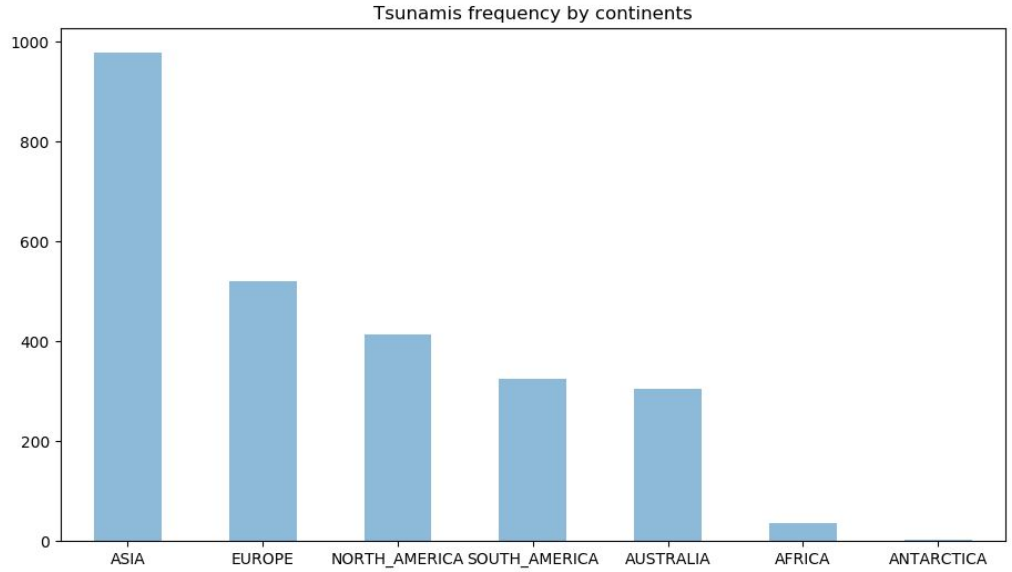

*Fig 2.1 bar chart for Tsunami occurrences by Continent*

We wanted to be sure if this meant that it was automatically more likely to get a devastating tsunami.To do this we wrote a python code to calculate the mean Validity for each continent. The two highest averages were Australia (also colloquially referred to as Oceania) and Asia.

```
europe validty mean:
2.0828516377649327

africa validty mean:
2.4285714285714284

north america validty mean:
1.9758454106280192

south america validty mean:
2.009259259259259

asia validty mean:
2.5991820040899793

australia validity mean:
2.7434210526315788

antarctica validty mean:
2.6666666666666665
```

*Fig 2.2 Validity mean per continent*

As you can see in the diagram above, even though Asia had the most tsunami occurrences, on average its incidents were not as devastating as Australia. We could conclude that even though Asia had the most incidents it does not make it the most dangerous region. This was an interesting find for our project.

Next of all we wanted to analyze central tendencies for the causes of the incidents in our dataset. We ran a python script for our dataset on these causes and obtained the following pie chart.

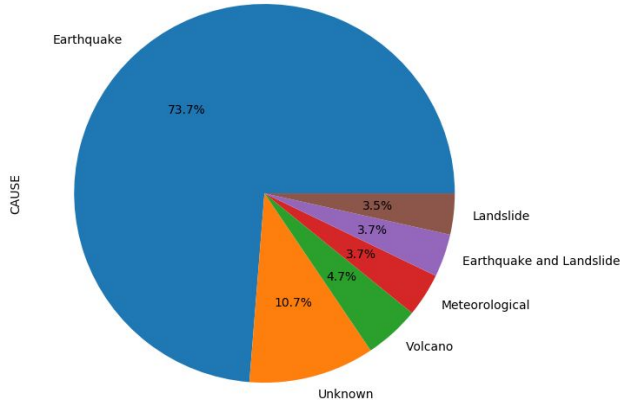


*Fig 2.3 Causes of tsunami's represented as a pie chart*

Clearly The earthquake is the mode cause of the tsunami's at 73.7%. Following this we wanted to find out if the mode cause remained true for all of the tsunami's in the dataset. To do this we ran the same code but only for tsunami's that had a validity

attribute of 4, in other words, absolute tsunami's. When we ran this there was nearly no change in the distribution.

Each of these earthquakes had a magnitude attribute. Following our last finding e then wanted to analyze the mode mode magnitude for these tsunami's with a validity rating of 4.

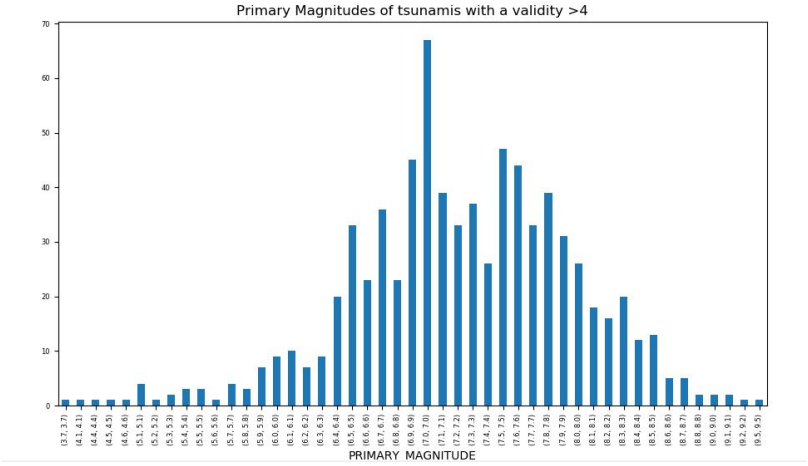


*Fig 2.4 Earthquake tsunami magnitudes*

We found that the mode magnitude was 7.0 for all of these significant incidences. The Primary magnitude graphed as a bar chart showed us a normal distribution that was skewed in a way that showed there were more outliers on the right side of the skew. Following this graph we then wanted to compare it to the magnitude for all earthquake related tsunamis. To do this we ran another python script for the mean and median of these figures. Interestingly enough the median was exactly the same as the mode for the absolute tsunami's, and the mean was also very close. See the below figure for what was returned from our script.

```
C:\Users\Sam\Desktop\CA4010>python magnitude_validity.py
Median Magnitude of all earthquake related tsunami's:
7.0
Mean Magnitude of all earthquake related tsunami's:
7.038725154215216
```

*Fig 2.5 Mean and median magnitudes*

## *Naive Bayes classifiers*

One of the techniques that had stood out to us the most was The Naive Bayes classifiers technique. After moving on from our workshop we noticed that there were a number of classes from our original dataset that we thought we could actually employ in our dataset for this part of the project. These were the 'All fatalities' and

‘Damage Total’. This left us with the classes as follows; ‘Year’, ‘Cause’, ‘Validity’, ‘Primary Magnitude’, ‘Intensity’, ‘All Fatalities’, ‘Damage Total’ and ‘House Destruction Total’. Given that we were dealing with predetermined classes, this form of supervised learning seemed suitable for our project. Given that Primary Magnitude class was blank for any of the tsunami occurrences that were not caused by an earthquake, we decided not to include this class in our prediction after all.

## *Algorithm Description*

After researching the implementation of Bayes classifiers online on various websites and youtube tutorials we implemented an algorithm consisting of a series of functions. The first function is called ‘loadCSV’ which takes the csv file and converts every element of the data set to a floating point number. This is essential given that there is division done in the algorithm (mean and standard deviation). The next function is called ‘DataSplit’, this splits the dataset randomly into data for training and data for testing.

```
def loadCSV():
    Data = csv.reader(open("sources_4.csv"))
    dataset = list(Data)
    for j in range (len(dataset)):
        if dataset[j] != '':
            dataset[j] = [float(i) for i in dataset[j]]
    return dataset


def DataSplit(dataset, Ratio):
    Trainer = int(len(dataset)*Ratio)
    tSet = []
    copy = list(dataset)
    while len(tSet) <Trainer:
        index = random.randrange(len(copy))
        tSet.append(copy.pop(index))
    return [tSet, copy]
```

Next we need to collect a summary of the training data. To do this we needed to separate each data by class value to calculate statistics for each class. This is done using the ‘ClassSeperator’ function. To calculate the mean of an attribute for a class value we have the mean function. This is used as the center point of our Gaussian distribution. We also require a standard deviation function to use on each attribute.

```
def ClassSeperator(dataset):
    classes = {}
    for i in range(len(dataset)):
        v  = dataset[i]
        if (v[-1] not in classes):
            classes[v[-1]] = []
        classes[v[-1]].append(v)
    return classes

def mean(nums):
    return sum(nums)/float(len(nums))

def standardDeviation(numbers):
    avg = mean(numbers)
    variance  = sum([pow(x-avg,2) for x in numbers])/float(len(numbers)-1)
    return math.sqrt(variance)
```

At this point we have what we need to summarize what we have for a given list of variables. To calculate the summaries for each attribute we have the ‘summarize’ and ‘summarize2’ functions to summarize by class. Using these functions we can estimate a given attribute value, and then estimate the accuracy. In the next function, ‘Probability’, the exponent is calculated with the mean and standard deviation and then we find the main division. We are now able to calculate the probability of an attribute belonging to a class. Now we want to combine the probabilities of all the values. This is done in the ‘ClassProbabilities’ function.

```
def summarize(dataset):
    summaries = [(mean(attribute), standardDeviation(attribute)) for attribute in zip(*dataset)]
    del summaries[-1]
    return summaries
    print('summaries: ' + summaries)

def summarize2(dataset):
    separated = ClassSeperator(dataset)
    summaries = {}
    for classValue, instances in separated.items():
        summaries[classValue] = summarize(instances)
    return summaries

def Probability(x, mean, standardDeviation):
    exponent = math.exp(-(math.pow(x-mean,2)/(2*math.pow(standardDeviation,2))))
    return (1/(math.sqrt(2*math.pi)* standardDeviation))*exponent

def ClassProbabilities(summaries, inputVector):
    probabilities = {}
    for classValue, classSummaries in summaries.items():
        probabilities[classValue] = 1
        for i in range(len(classSummaries[1])):
            mean, standardDeviation = classSummaries[i]
            x = inputVector[i]
            probabilities[classValue] *= Probability(x, mean, standardDeviation)
        return probabilities
```

We now have what we need to make a prediction. The ‘Predict’ function takes the summaries and the ‘inputVector’ which is basically all the probabilities which are being inputted. The function after this is called ‘getPredictions’ which gets predictions based on the training data and its summary.

```
def predict(summaries, inputVector):
    probabilities = ClassProbabilities(summaries, inputVector)
    bestLabel, bestProb = None, -1
    for classValue, probability in probabilities.items():
        if bestLabel is None or probability >bestProb:
            bestProb = probability
            bestLabel = classValue
    return bestLabel

def getPredictions(summaries, testSet):
    predictions = []
    for i in range(len(testSet)):
        result = predict(summaries, testSet[i])
        predictions.append(result)
    return predictions
```

Now the predictions can be compared to the class values in the test dataset and thus accuracy can be calculated . The ‘acc’ function calculates this ratio. See ‘acc()’ and ‘main()’ in the figure below.

```
def acc(testSet, predictions):
    correct = 0
    for x in range(len(testSet)):
        if testSet[x][-1] == predictions[x]:
            correct += 1
    return (correct/float(len(testSet)))*100.00

def main():
    filename= "sources_clean_BAYES.csv"
    splitRatio = 0.9
    dataset = loadCSV()
    trainingSet, testSet = DataSplit(dataset, splitRatio)
    print('Split {0} rows into train = {1} and test {2} rows'.format(len(dataset), len(trainingSet), len(testSet)))
    #prepare model
    summaries = summarize2(trainingSet)
    #test model
    predictions = getPredictions(summaries, testSet)
    accuracy = acc(testSet, predictions)
    print('accuracy: {0}%'.format(accuracy))
```

All of these methods are then called using the ‘main()’ function. In this function we can define what the split ratio will be.

## *Results and Analysis*

In conclusion, our final dataset for our prediction contained  the classes ‘Year’, ‘Cause’, ‘Validity’, ‘Intensity’, ‘Continent’, ‘All Fatalities’ and ‘Damage total’. Further cleaning was done due to empty spaces, mainly in the Intensity class. When we ran the algorithm on our dataset (Split ratio at 80%) we were usually receiving percentages over 60%. Given the empty values (0) for our last 2 classes and that several of our attributes had normalized scales (some very small, Validity for example) this is the reason for the high percentages.

Some of the instances in the dataset can be very diverse and obscure. For example an instance may have a validity rating of 4, Caused by a volcano and have 0 casualties and deaths. Whereas another may have an Intensity value of 4 and killed 10000 people but not have a value for magnitude because it was caused by an Earthquake. A lot of work has been done in identifying the central tendencies, preparing the data accordingly and choosing what to use in our prediction. Although we may not have gotten the percentage result we had desired (50-60%), we as a group have learned a very large amount about what is involved in this area from data cleaning, preparation, the use of graphs etc. and we look forward to using these skills in our future projects and possibly jobs in the coming years.

## Sources of information

https://www.youtube.com/watch?v=vz_xuxYS2PM&t=1009s
https://www.machinelearningplus.com/predictive-modeling/how-naive-bayes-algorithm-works-with-example-and-full-code/