KULLIYYAH OF INFORMATION AND COMMUNICATION TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

FINAL YEAR PROJECT REPORT

GIS MAPPING AND SHORTEST-PATH ALGORITHM FOR DRONE-AIDED FLOOD RESCUE OPERATIONS

AHMAD ABU SAIID
1526703

MOHAMMAD NAFEES BIN ZAMAN
1616357

SUPERVISED BY
ASSOC. PROF. DR. RAINI HASSAN

20th DECEMBER 2019
SEMESTER 1, 2019/2020

FINAL YEAR PROJECT REPORT

GIS MAPPING AND SHORTEST-PATH ALGORITHM FOR DRONE-AIDED FLOOD RESCUE OPERATIONS

by

AHMAD ABU SAIID
1526703

MOHAMMAD NAFEES BIN ZAMAN
1616357

SUPERVISED BY
ASSOC. PROF. DR. RAINI HASSAN

In partial fulfillment of the requirement for the
Bachelor of Computer Science
Department of Computer Science
Kulliyyah of Information and Communication Technology
International Islamic University Malaysia

20th DECEMBER 2019
Semester 1, 2019/2020

ACKNOWLEDGEMENTS

Praise and thanks to Allah first and foremost whose blessing enabled us to accomplish this project.

We wish to express our deepest appreciation to our beloved supervisor Dr. Raini Hassan for her kind and unbounded support and constant guidance throughout this research for the Final Year Project of our bachelor's degree in Computer Science.

ABSTRACT

Floods are quite common in Malaysia almost every year, which occurs mostly during the monsoon season. Thousands are affected by the floods and rescuers are not always quick and efficient enough to save the majority of the victims. This research aims to implement Geographic Information System (GIS) mapping with the help of K-Means clustering and weights to identify the major parts of Terengganu that are more susceptible to floods (using a Risk Map) and the construction of an optimal shortest-path (SP) algorithm to conduct the rescue operation in an efficient manner. The results obtained for k-means clustering is a total variance of 89.2% with six clusters being the optimal solution. The SP algorithm outperforms the random path and longest path that could have been used during a flood rescue. The combined risk map and SP algorithm provides a solution for flood rescue teams to plan and conduct their rescue operations in a more effective manner.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF ABBREVIATIONS

RESEARCH PAPER

API	Application Programming Interface
BSS	Between Sum of Squared Distance
GIS	Geographic Information System
PCA	Principal Component Analysis
SP	Shortest-Path Algorithm
SVM	Support Vector Machine
TSP	Travelling Salesman Problem
WSS	Within Sum of Squared Distance

PREVIOUS WORKS

AHP	Analytical Hierarchy Process
FR	Frequency Ratio
LULC	Land Use/Land Cover
MLE	Maximum Likelihood
SPI	Stream Power Index
TWI	Topographic Wetness Index

CHAPTER ONE

INTRODUCTION

1.1 Background

Floods are one of the most common natural disasters around the world and normally results in large human and economic losses. Given the fact that Malaysia's geographical location is in the tropical area, most of the floods naturally occur during the monsoon season that normally cycles every year involving heavy and frequent rainfall during that period (Elgilany, Jamalludin & Saidatulakmal, 2012). Poor drainage systems in some urban areas is another factor that tends to intensify the effects of heavy rain in those areas, although there are some initiatives being taken to rectify those current issues (IUKL GDRC, 2018). In Malaysia alone, 90 percent of the damage originating from natural disasters are contributed by floods (Pradhan, 2010). Annual flood damage costs could go up to MYR400 million, affecting lives, properties, agriculture and major road systems (Pradhan, 2010).

Terengganu is one of the major states that are affected by floods every year and will therefore be the main focus in this research paper. It is located at the east coast of Peninsular Malaysia (beside the South China Sea) and is the neighboring state to Kelantan and Pahang. It is situated between latitudes 05˚51'06"N to 03˚55'37"N and longitudes 102˚21'11"E to 103˚31'28"E. Terengganu covers approximately 12,995 square kilometers (Fadlalla, Elsheikh, Ouerghi & Elhag, 2015) and is comprised of eight districts and 33 towns. The place is normally hot and humid throughout most of the year, averaging at 28˚C to 30˚C during daytime and a little cooler during the night. Its average annual rainfall is between 2575 and 2645 mm with heavy rain occurring mostly from December to January (Fadlalla, Elsheikh, Ouerghi & Elhag, 2015). This

research aims to provide a better means of carrying out rescue operations during floods through two main components, a flood risk map and shortest-path algorithm.

Figure 1: Districts of Terengganu

1.2 Problem Statement

Current rescue operation techniques for floods are not always quick and efficient as they should be.

1.3 Project Objectives

- To construct a flood risk map using PCA, K-Means Algorithm and GIS
- To predict flood risk using machine learning algorithm
- To construct a shortest-path algorithm
- To derive a potential solution for a quicker flood rescue operation

1.4 Project Scope and Overview

This research aims to provide a better means of carrying out rescue operations during floods through two main components, a flood risk map and shortest-path algorithm. To construct the flood risk map, data (mainly spatial data) have been collected for the different towns located in Terengganu. The main features included are Population, Distance from Sea (km), Altitude/Elevation (m), Average Annual

Rainfall (mm), Latitude and Longitude. The initial four features were then used to create clusters using the unsupervised method, K-means clustering. After identifying the clusters, a weighing system was implemented to calculate the risk of each town. These were then classified as low, medium or high risk. The risk map was then constructed and visualized using GIS. For further convenience, a Support Vector Machine (SVM) model has been implemented to predict flood risks for future cases. The main purpose of the flood risk map is to aid flood-rescue organizations to identify which areas of Terengganu would be suitable to set up flood-rescue team bases during the monsoon season to ensure that the teams are as close as possible to the regions that are affected by flood and can reach out to the victims more conveniently.

Next, a shortest-path algorithm has been constructed which mainly comprises of three components, relative coordinate conversion, Euclidean algorithm and Travelling Salesman Problem (TSP). The goal for the shortest-path algorithm is to first receive the collection of victim locations (by their coordinates) that are identified by the deployment of drones. Drones are a very useful component in this situation due to their tiny size, versatility and cost. Once these victim locations are fed into the shortest-path algorithm, it will identify the shortest possible route (from the rescue base) to reach out to all victims once and return back to the rescue base (source). This mechanism could lead to future rescue operations being more efficient and saving more lives in the process.

1.5 Significance of the Project

As floods are a common type of natural disaster, it is essential that we try and find better ways to deal with these kind of situations as they are existent in many parts of the world. This research aims to provide a better solution for flood rescue teams to operate in a shorter period of time and with better efficiency.

CHAPTER TWO

REVIEW OF PREVIOUS WORKS

A total of twelve papers have been reviewed. From these articles, we have separated them into two sections. The first section includes 6 papers for GIS Flood Risk Mapping and the second section includes 6 papers for Shortest-Path Algorithm.

2.1 GIS Flood Risk Mapping

In regards to flood risk mapping, firstly, two papers have been reviewed which focuses on Geographic Information System (GIS) mapping for identifying areas that are prone to flood occurrences in different parts of Malaysia. Tehrany, Pradhan and Jebur (2014) targets a specific part of the Terengganu State known as Kuala Terengganu City. The main techniques involved an ensemble weights-of-evidence (Bayesian Model) and Support Vector Machine (SVM) which were applied to 10 flood conditioning factors (flood inventory, slope, stream power index (SPI), topographic wetness index (TWI), altitude, curvature, distance from the river, geology, rainfall, land use/cover (LULC) and soil type) to classify the parts of the city that are more susceptible to floods. The success rate and prediction rate acquired was 96.48% and 95.67% respectively, which indicates that the model was a success. In the second paper, Mojaddadi et al. (2017) implements SVM and a Frequency Ratio (FR) approach for weighing each flood conditioning factor to be inputted into the SVM model to produce a flood risk map for the Damansara River Catchment area. The model uses a total of 13 flood conditioning parameters (altitude, aspect, slope, curvature, stream power index, topographic wetness index, sediment transport index, topographic roughness index, distance from river, geology, soil, surface runoff, and land use/cover (LULC)) to produce the flood risk map which achieved a success rate of 89.7% and

prediction rate of 78.9%. Both papers have some similarity in terms of their techniques involved but have some different approaches integrated within them as well. They also have used some similar flood conditioning factors for their specific model. Furthermore, the results are very promising which proves that SVM may be a suitable algorithm to be incorporated for flood risk mapping.

Looking into the third paper, Ouma and Tateishi (2014) constructed a public-based flood map for estimating flood risks in the urban parts of Eldoret Municipality, Kenya. The model was integrated with an Analytical Hierarchy Process (AHP) and Geographic Information System (GIS) analysis technique. The flood risk vulnerability mapping followed a multi-parametric approach and integrates some of the flood causing factors such as rainfall distribution, elevation and slope, drainage network and density, land-use/land-cover and soil type. An urban flood risk index (UFRI) was determined by the degree of vulnerability and exposure in each area. The results were validated using flood depth measurements, with a minimum average difference of 0.01m and a maximum average difference of 0.37m in depth of observed flooding in the different flood prone areas. Similarly, with respect to area extents, a maximum error of not more than 8% was observed in the highly vulnerable flood zones. Their proposed approach achieved an average accuracy of 92% which indicates that it was reliable.

In the fourth paper, Kia et al. (2012) developed a flood model from various flood causative factors using ANN techniques and GIS to determine flood-prone areas in the southern part of Malaysia (Johor). The ANN model for this study was developed in MATLAB using seven flood causative factors which includes rainfall, slope, elevation, flow accumulation, soil, land use and geology. The ANN was used to directly produce water levels and then the flood map was constructed in GIS. To measure the

performance of the model, four criteria performances, including a coefficient of determination (R squared), the sum squared error, the mean square error, and the root mean square error were used. The verification results showed satisfactory agreement between the predicted and the real hydrological records. The authors suggest that the model can be used to predict floods in the Johor area with an acceptable accuracy.

For the fifth paper, Tehrany, Pradhan and Jebur (2013) have targeted the state of Kelantan for developing spatial prediction of flood susceptible areas using Rule-Based Decision Trees with a novel ensemble bivariate and multivariate statistical approach in GIS. The main purpose of the research was to compare the prediction performance of the Rule-Based Decision Tree against the combination of a Frequency Ratio (FR) and Logistic Regression (LR) model. A flood inventory map with a total of 155 flood locations was extracted from various sources over the parts of Kelantan. Then the flood inventory data was randomly divided into a testing dataset 70% (115 flood locations) for training the models and the remaining 30% (40 flood locations) were used for validation purposes. The spatial database features included digital elevation model (DEM), curvature, geology, river, stream power index (SPI), rainfall, land use/cover (LULC), soil type, topographic wetness index (TWI) and slope. For validation both success and prediction rate curves were performed. The validation results showed that, the area under the curve for the results of DT against the integrated method of FR and LR was 87% and 90% for success rate and 82% and 83% for prediction rate respectively. This indicates that both models performed well although the model from the combination of FR and LR slightly outperformed the decision tree model showing it is a more suitable method for constructing a flood-susceptibility map.

In the sixth paper, Jalayer et al. (2014) identifies urban flood risk hotspots in the city of Addis Ababa, Ethiopia. The study includes three GIS based frameworks for identifying the flood risk hotspots for residential building and urban corridors. The factors included were Topographic Wetness Index (TWI), a map of residential areas and urban corridors and a geo-spatial census dataset. The Maximum Likelihood (MLE) method was used for estimating the threshold used for identifying the flood risk areas based on a single spatial area. In addition to that, a Bayesian parameter estimation was applied for the purpose of estimating the Topographic Wetness Index threshold for more than one spatial area. The results indicate that 50% of the total land area is prone to flooding while 67% of the population is estimated to live in this flood-risk area and therefore, the overall impact of a flood occurrence in this city would be huge.

For this research, the data collected does not have any target label (risk feature) and therefore requires an unsupervised learning method. Given that matter, K-means clustering, and a weight system was applied to identify the initial flood risk. Another key difference between this research and others is the purpose of the flood risk map. The flood risk map in this research is used to help flood-rescue organizations to identify which areas of a region (Terengganu) would be suitable to set up flood-rescue team bases during the monsoon season while most previous researches were more focused on using the risk map for pre-flood preparations, of how to stop the floods from occurring by initially identifying the high risk areas. This is in contrast to the current research which focuses more on post-flood scenarios.

2.2 Shortest-Path Algorithm

As for the construction of a shortest-path algorithm, six different articles have been reviewed. In the first article, Alhoula and Hartley (2014) aims to provide a shortest-path algorithm for travellers to determine the best routes to take in order to reach their

destination. Based on Dijkstra's Algorithm, they evaluated two implementations which include a static and time dependent network. The results are based on two factors which are the type of network (static or time-dependent) and size of the network (no. of nodes). The execution time of the program for time-dependent networks were much shorter compared to a static network for small, medium and large networks.

For the second article, Takeda, Ito and Matsuno (2016) focuses on a path generation algorithm for search and rescue robots based on insect behaviour (Ladybirds). This implementation focuses on reducing the range of a search area once a target is identified and returns to a larger range once another target is being identified. The path generation algorithm is optimized with the help of the Genetic Algorithm. The results are analyzed by the number of survivors found and the time taken for the search operation using this algorithm compared to a traditional grid-search method. Their algorithm outperformed the grid search for three different cluster sizes of 20, 50 and $100\mathrm{m}^2$. Although that is the case, as the cluster size increased, grid-search performed better indicating that it could be beneficial for larger areas.

In the third paper, Leng and Zeng (2009) focuses on providing an improved shortest path algorithm for computing one-to-one shortest paths on road networks. The aim was to enhance the TWO-Q algorithm to reduce time wastage. The research produced a variation of that algorithm named as Minimum Label Delimiting TWO-Q algorithm (MiLD-TWO-Q). The experimental results showed that their proposed algorithm performs much better for paths that are really short and performs the same for longer paths. The proposed algorithm proved to be more efficient compared to the original TWO-Q algorithm.

Looking into the fourth paper, Becker, Florian and Szczerbicka (2013) introduces a new multi-agent algorithm for the purpose of search and rescue scenarios involving

unknow terrain. This method combines the concept of the flood algorithm (for exploration) and path optimizing features from the ant algorithm. In the first phase, it involves the rapid exploration of the unknown terrain. The second phase involves the construction of shortest paths from the point of interest and back to the original base. These two phases enable the commencement of rescue operations to work in parallel with the ongoing search. For the experimental setup, five different scenarios were chosen for the simulation of the algorithm. This included two maps with a size of 1000 × 1000 cells and three maps with the size of 500 × 500 cells. Each map contained one starting point and its position was randomly chosen from the number of cells, that adjoin the border of the map. For most of the scenarios their proposed multi-agent flood algorithm performed a little slower compared to the Brick and Mortar algorithm. Although that was the case, their proposed algorithm improved as the size of the map and agents increased which indicates that the algorithm scales rather well.

In the fifth paper, Arif, Ferdous and Nijami (2012) conducted a comparative study of different path planning algorithms for a water-based rescue system. A total of four algorithms were tested, namely, the Graph Search algorithm, Breadth-First algorithm, Dijkstra's algorithm and the A-Star (A*) algorithm. From the results, it can be seen that each algorithm performs better in different situations. For maps with same cost cells, with one starting-one goal cell and multi goal cells, using Breadth-first algorithm was the best if the computational time is the main priority, but if size of memory was the priority, then A* would be a better alternative. For maps with different cost cells and with one starting-one goal cell A* was the best in terms of computational time and size of memory. In overall, the A-Star (A*) algorithm performed the best compared to the other algorithms.

For the sixth paper, Zhong, Malinen, Miao and Franti (2014) proposed a fast-minimum spanning tree algorithm based on a K-Means model. The proposed algorithm employs a divide-and-conquer scheme to produce an approximate MST with theoretical time complexity of O($N^{1.5}$), which is faster than the conventional MST algorithms having O(N^2). The proposed algorithm comes in two stages. In the first phase, the K-Means is used to partition the dataset into $\sqrt{N}$ clusters. Then the MST algorithm is applied to each cluster which produces $\sqrt{N}$ minimum spanning trees. Each of these trees are then connected to form one MST. In the second phase, $\sqrt{N} - 1$ clusters are formed, and a new MST is achieved from that. Finally, both MSTs are combined together to form a graph which can be considered as a more accurately generated MST graph. The experimental setup showed that their proposed algorithm was computationally efficient, and the approximation was close to a normal MST, thus it does not suffer from less performance.

Each of the mentioned researches conducted for shortest-path algorithms has proven to be beneficial for the specified unique scenario which shows that different problems require different approaches for acquiring the shortest path. In this current case, the Travelling Salesman Problem has been chosen to implement as it considers all possible paths to every single node as well as the consideration of returning back to the starting point.

CHAPTER THREE

METHODOLOGY

3.1 Project Workflow

The workflow can be understood with more convenience through the diagram below.

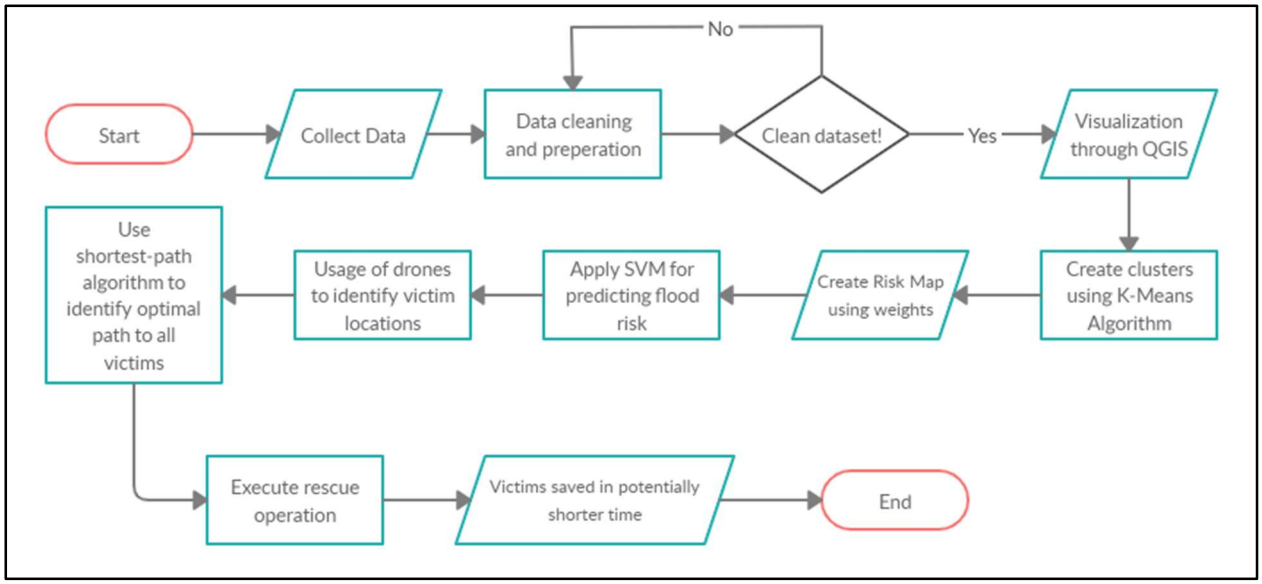


Figure 2: Project Workflow

3.2 Data Collection and Pre-processing

The collection of data involved two different stages. Firstly, the collection of spatial data on Terengganu was retrieved from a dataset that was recorded for the latest general election in Malaysia. This dataset needed to be cleaned and prepared before the next step. Next the four main features were individually collected using Google Maps API, Malaysian state records and weather forecast sources. The combined dataset was pre-processed before moving into the exploratory data analysis stage.

3.3 Exploratory Data Analysis (EDA)

The prepared data was then visualized using RStudio and QGIS to identify patterns and trends in the data. The first comparison was between the towns' distance from sea and altitude of that region. Secondly, the analysis of how many people lived in low or high altitudes. From figure 3 below, it can be understood that most of the

populated towns were situated at areas close to the South China Sea and the land in that area had very low altitudes.

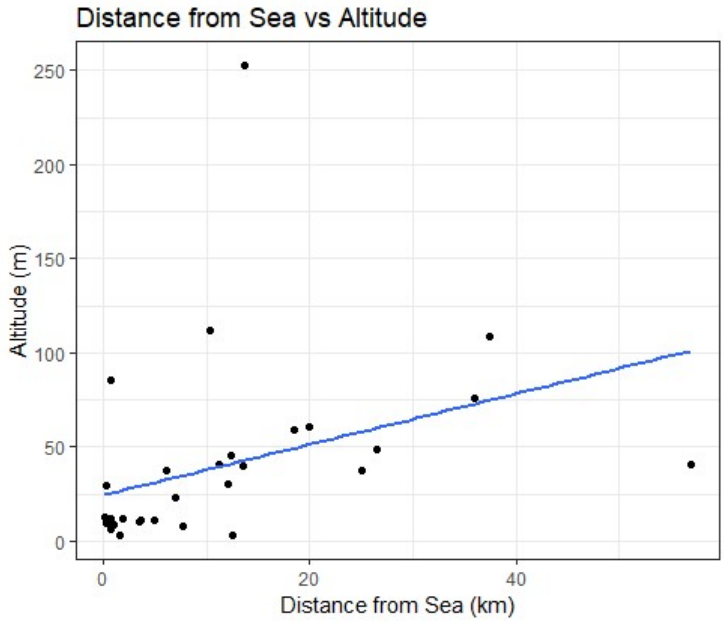


Figure 3: Distance from Sea vs Altitude

In figure 4 below, it can be identified that the towns with high population are located at low altitudes which means more of the Terengganu population is at risk of being affected during flood cases.

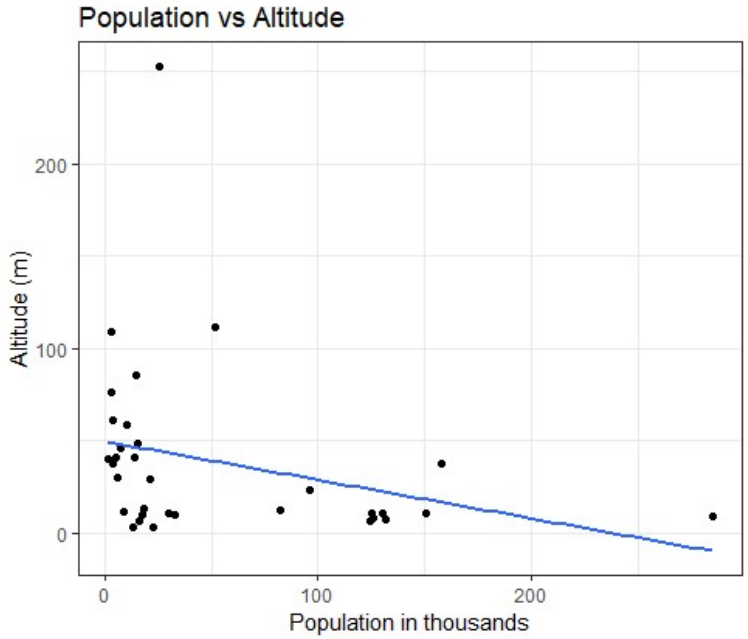


Figure 4: Population vs Altitude

The third analysis was on the population of all villages in Terengganu. From figure 5 below, it can be seen that many of the villages with high population is located closer to the South China Sea. This makes a lot of sense, as the coastal area provides various types of advantages in terms of economy.

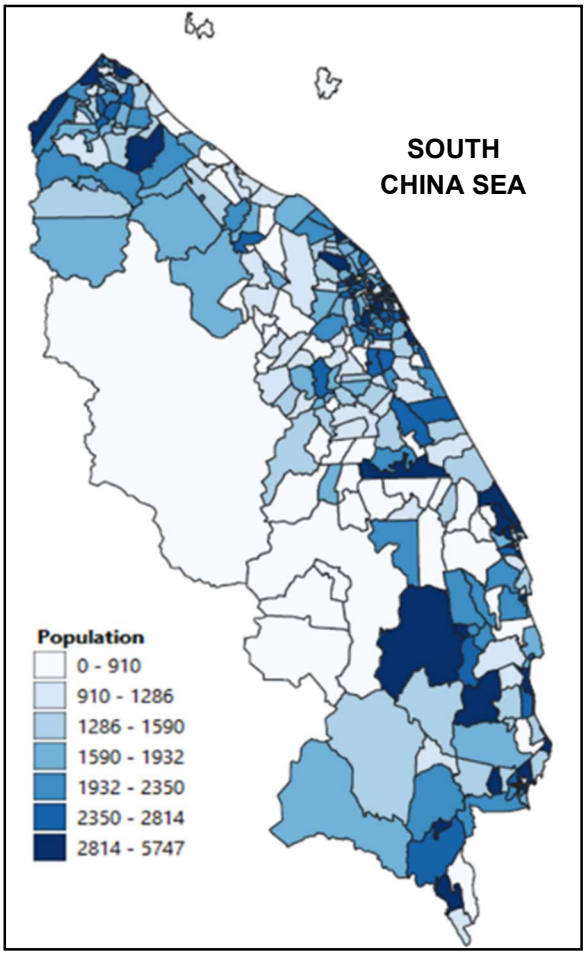


Figure 5: Population of Terengganu

3.4 K-Means Clustering

The four main features, population, distance from sea, altitude and average annual rainfall were used to create clusters. To begin with, Principal Component Analysis (PCA) was applied for dimensionality reduction and scaling. From that, the first two principal components (composite combination of all attributes) were used for the K-

Means model. Next, the elbow method (figure 6) was implemented in order to identify the optimal number of clusters through the lowest within sum of squared distance (WSS). The optimal no. of clusters was six. This was further verified by applying the Gap-Stat and Silhouette method as well.

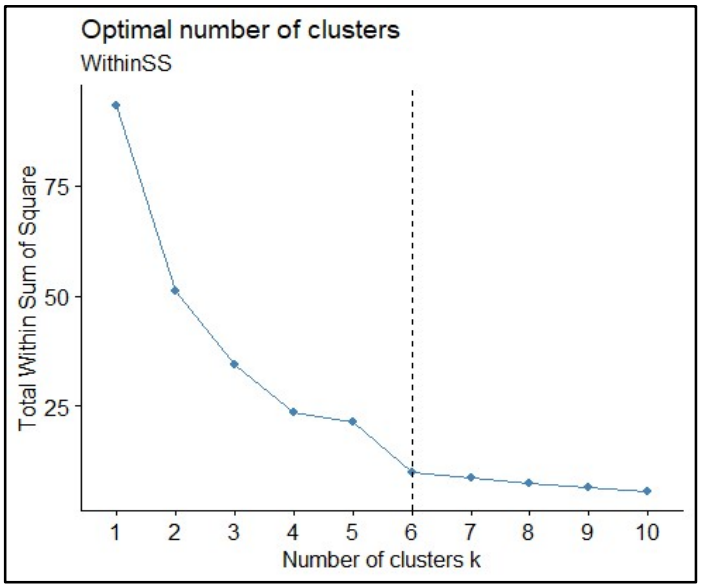


Figure 6: Elbow Method

The K-Means model was then executed with k equals to 6. The clusters can be seen in figure 7 below .

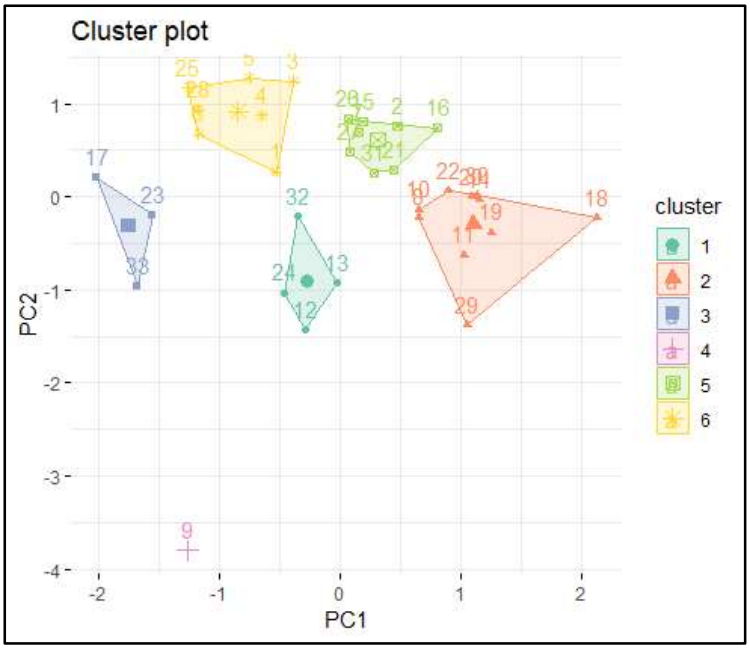


Figure 7: K-Means Clustering

3.5 Risk Weights and Risk Map

To classify these clusters according to low, medium and high risk, a weighing system was applied. Firstly, each attribute in each cluster was averaged to get their mean values. Next, for each attribute, the mean values of all six clusters were compared with each other and given a score from 1 to 6, where 1 refers to lowest risk and 6 refers to highest risk. Once all attributes were compared, the risk values were summed up together to provide a final score for each cluster. Given that the maximum points a cluster can acquire is 24, this value was split into three sections, low for a value close to 8, medium for a value close to 16 and high for a value close to 20. The outcome resulted in each cluster being assigned a low, medium or high risk classification.

A flood risk map and heat map were then created using QGIS which can be seen in the result section below (Figure 9 & 10).

3.6 Support Vector Machine (SVM)

Since, the target label (risk feature) has been acquired, an SVM classifier with a linear kernel and 10-fold cross validation was implemented for predicting the risk feature which could be used for future scenarios instead of going through the complicated method of K-Means clustering and risk weights. This can be seen as turning unsupervised learning into supervised learning.

3.7 Shortest-Path Algorithm (SP)

Once the flood risk map has been created, this helps to identify which areas that rescue team bases can be set up during the monsoon season in preparation of floods. Each base is considered as an origin point for the shortest-path algorithm. Once a flood occurs in a specific town, drones would be deployed from the closest base to identify the locations (coordinates) of flood victims. This collection of coordinates

would then be fed into the SP algorithm to identify the best route that can be taken from the base to reach all victims (nodes) once and return back to the base (origin point) with those victims.

The algorithm comprises of three components. The first component is the relative coordinate conversion which converts each victim(s) coordinate relative to the rescue base coordinates (origin). The second component included is the Euclidean Algorithm for the purpose of calculating the distance between each and every node (victim location). Lastly, the main component is the Travelling Salesman Problem algorithm for pairing every single node with one another and identify the best route for the rescue teams to take from the base to reach each victim location once and return back to their base.

Coords: x_1, x_2, y_1, y_2

Function Distance:

$math.sqrt((-x_2)^2 + (y_1 - y_2)^2)$

Function Conversion:

INPUT *coords*
For *p* ***in*** *coords*
$a = (-Origin1) + points[p-1][0]$
$b = (-Origin2) + points[p-1][1]$

Function TotalDistance:

$d = 0$
For *i* ***in*** *tour*
$x_1 = converted\_coords[tour[i-1]][0]$
$y_1 = converted\_coords[tour[i-1]][1]$
$x_2 = converted\_coords[tour[i]][0]$
$y_2 = converted\_coords[tour[i]][1]$
$d = d + distance(x_1, y_1, x_2, y_2)$
$x_1 = converted\_coords[tour[len(tour)-1]][0]$
$y_1 = converted\_coords[tour[len(tour)-1]][1]$
$x_2 = converted\_coords[tour[0]][0]$
$y_2 = converted\_coords[tour[0]][1]$
$d = d + distance(x_1, y_1, x_2, y_2)$

Main Function:

INPUT *converted_coords*
$Tour \rightarrow length\ of\ data$
$AllPossibleTour \rightarrow list(permutation(tour))$
Call *shortest_path*
Call *longest_path*
Call *random_path*

Figure 8: Shortest-Path Algorithm

CHAPTER FOUR

RESULTS AND EVALUATION

4.1 Machine Learning Models (K-Means and SVM)

The K-Means model was implemented for 6 clusters which achieved a total within sum of squared distance (WSS) of 10.04 and between sum of squared distance (BSS) of 83.35. The WSS should always be low as it is the comparison of points within the same cluster, and a smaller WSS means more similarity which should be the goal for points inside a cluster. The BSS on other hand, is for looking at the difference between each cluster so a high value is great as it indicates that each cluster is unique from the others. The total variance achieved was 89.2% which indicates that the model was effective. Furthermore, this was compared to other models that did not include scaling or PCA, and this model performed much better with the highest total variance.

For the SVM classifier, a linear kernel with a soft margin of two was used. The model was iterated through the ten-fold cross validation method. The accuracy of the prediction results was 90.17%. The prediction of high and medium risk was very good, while low risk was okay, but not as great as the high and medium prediction. The ‘High’ risk class had a precision and recall of 0.90 and 1.00 respectively, while the ‘Medium’ risk class achieved 0.82 for both precision and recall. The ‘Low’ risk class had a precision of 1.00 and recall of 0.50. The F-1 score can also be seen through the table below. To sum it up, the SVM model worked well in predicting accurately in most cases.

True Label	Predicted Label			
		High	Low	Medium
	High	18	0	0
	Low	0	2	2
	Medium	2	0	9

Table 1: Confusion Matrix of SVM

	Precision	Recall	F1-Score
High	0.90	1.00	0.95
Low	1.00	0.50	0.67
Medium	0.82	0.82	0.82

Table 2: Precision, Recall & F1-Score of SVM

4.2 Flood Risk Map and Heat Map

The flood risk map and heat map were created using QGIS. This helps to identify which areas that rescue teams can set up their base during the monsoon season. The heat map helps to identify the main region where there are several towns with risk, which indicates multiple rescue bases should be set up around that area. From the heat map, it can be seen that towns around Kuala Terengganu would require more or larger rescue teams. See figure 9 and 10 below.

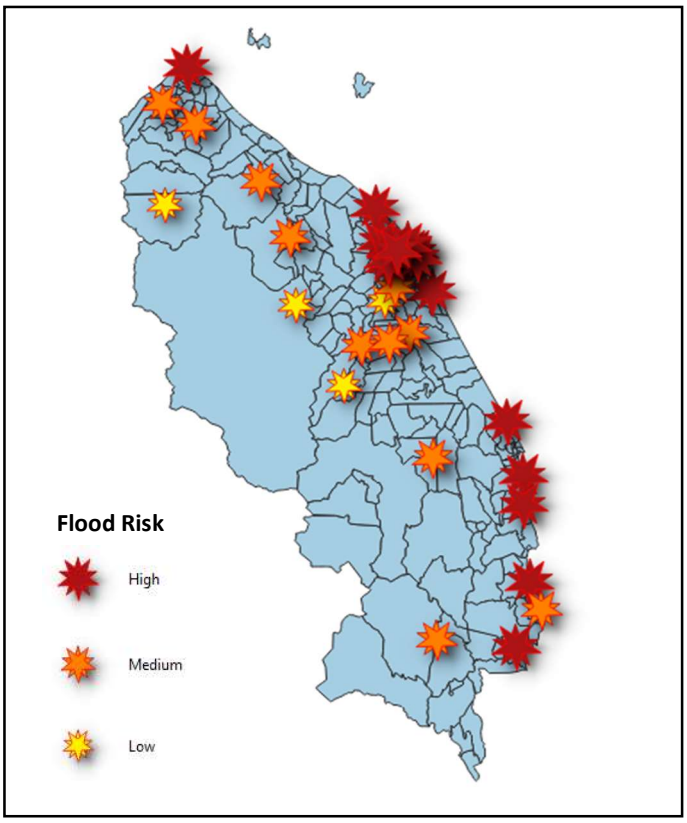


Figure 9: Flood Risk Map

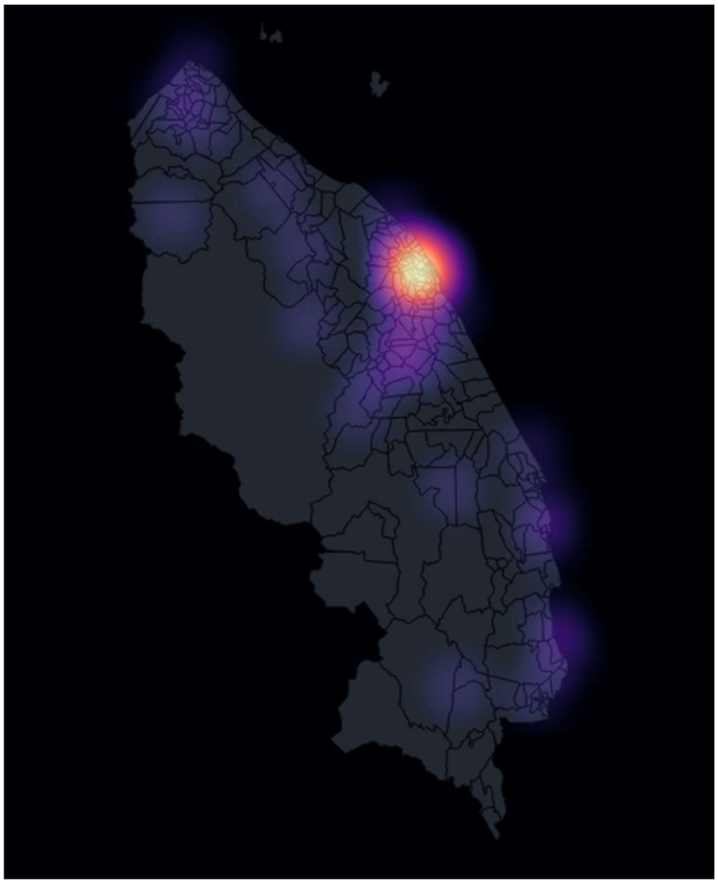


Figure 10: Heat Map

4.3 Shortest-Path Algorithm (SP)

The shortest-path algorithm was tested and worked perfectly as expected. It was then compared to the random path and longest path to see if the SP algorithm does indeed perform better. The results show that the SP algorithm outperforms the other two paths. A demo can be seen below (figures 11 & 12).

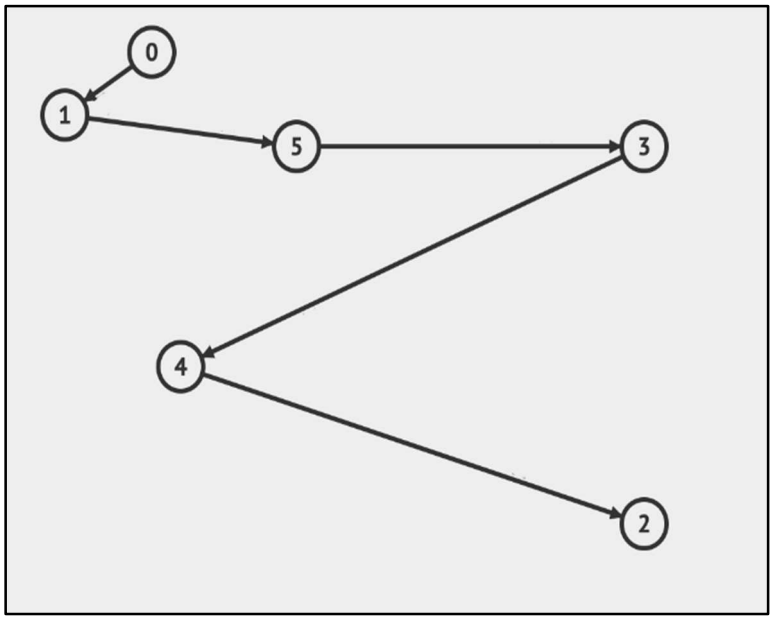


Figure 11: Demo of Shortest-Path Algorithm

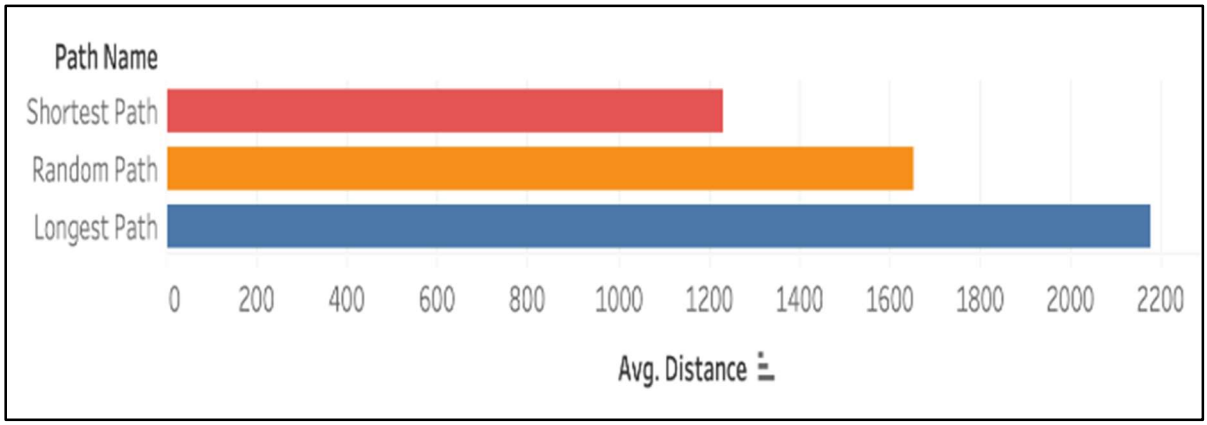


Figure 12: Comparison of different paths

CHAPTER FIVE

FUTURE WORKS AND CONCLUSION

5.1 Discussion and Future Works

The combination of data science as well as computer science for improving rescue operations, whether it be floods or other natural disasters, can prove to be very beneficial and this research has approached the flood rescue operation in a unique way to try to further enhance and improve the rescue planning and operations during floods. Furthermore, this approach could be used for other types of natural disasters that require search and rescue operations.

This research contains multiple different elements and is therefore quite huge and can be improved in different ways. Firstly, by having more data to further improve the machine learning models. Next, by taking into account other real-life factors such as blockages. In addition to that, a real-time image recognition for drones could be implemented so that humans are not required to conduct the surveillance from drones manually. Lastly, the consideration of a fairness-path which takes into account, the victim locations that require higher attention due to higher levels of water, the collapsing of structures or other emergency matters, thus having more priority than other victims.

5.2 Conclusion

This research has captured the importance of improving flood-rescue operations with the modern technology that is available out there. The main components of this research include machine learning and GIS for creating a flood risk map to identify suitable locations for rescue teams to set up, as well as the construction of a shortest-path algorithm for rescue-teams to reach each victim location once and return back to base. The machine learning models worked very well with K-Means achieving a total

variance of 89.2% and SVM achieving a prediction accuracy of 90.17%. Following up after the risk map, the shortest-path algorithm has performed as expected and does much better than the random path as well as longest path comparisons. The proposed method should aid rescue teams to reach out to victims in a shorter period of time.

REFERENCES

Alhoula, W., & Hartley, J. (2014). Static and time-dependent shortest path through an urban environment: Time-Dependent Shortest Path. *Proceedings of 2014 Science and Information Conference, SAI 2014*, 1027–1029. https://doi.org/10.1109/SAI.2014.6918315

Becker, M., Blatt, F., & Szczerbicka, H. (2013). A multi-agent flooding algorithm for search and rescue operations in unknown terrain. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8076 LNAI*, 19–28. https://doi.org/10.1007/978-3-642-40776-5-5

Elgilany, A., Jamalludin, S. & Saidatulakmal, M. (2012). Importance of floods management in malaysia. *International Conference on Environmental Research and Technology (ICERT 2012)*.

Fadlalla, R., Elsheikh, A., Ouerghi, S., & Elhag, A. R. (2015). Flood Risk Map Based on GIS , and Multi Criteria Techniques (Case Study Terengganu Malaysia), (August), 348–357.

IUKL GDRC. (2018, February 21). Seminar On Forecasting Flood And Control Technology. Retrieved from https://iukl.edu.my/igeo/2nd-i-geo-seminar-series-17-june-2014-seminar-forecasting-flood-control-technology/.

Jalayer, F., De Risi, R., De Paola, F., Giugni, M., Manfredi, G., Gasparini, P., … Renner, F. (2014). Probabilistic GIS-based method for delineation of urban flooding risk hotspots. *Natural Hazards*, *73*(2), 975–1001. https://doi.org/10.1007/s11069-014-1119-2

Kia, M. B., Pirasteh, S., Pradhan, B., Mahmud, A. R., Sulaiman, W. N. A., & Moradi, A. (2012). An artificial neural network model for flood simulation using GIS: Johor River Basin, Malaysia. *Environmental Earth Sciences*, *67*(1), 251–264. https://doi.org/10.1007/s12665-011-1504-z

Leng, J., & Zeng, W. (2009). PRB 47(1993) 991.pdf, (i), 1979–1982.

M.MasudurRahmanAl-Arif, S., H. M. Iftekharul Ferdous, A., & Hassan Nijami, S. (2012). Comparative Study of Different Path Planning Algorithms: A Water

based Rescue System. *International Journal of Computer Applications*, *39*(5), 25–29. https://doi.org/10.5120/4817-7058

Mojaddadi, H., Pradhan, B., Nampak, H., Ahmad, N., & Ghazali, A. H. bin. (2017). Ensemble machine-learning-based geospatial approach for flood risk assessment using multi-sensor remote-sensing data and GIS. *Geomatics, Natural Hazards and Risk*, *8*(2), 1080–1102. https://doi.org/10.1080/19475705.2017.1294113

Ouma, Y. O., & Tateishi, R. (2014). Urban flood vulnerability and risk mapping using integrated multi-parametric AHP and GIS: Methodological overview and case study assessment. *Water (Switzerland)*, *6*(6), 1515–1545. https://doi.org/10.3390/w6061515

Pradhan, B. (2010). Flood susceptible mapping and risk area delineation using logistic regression, GIS and remote sensing. Journal of Spatial Hydrology, 9(2).

Takeda, T., & Matsuno, F. (2016). Path Generation Algorithm for Search and Rescue Robots Based on Insect Behavior, 270–271.

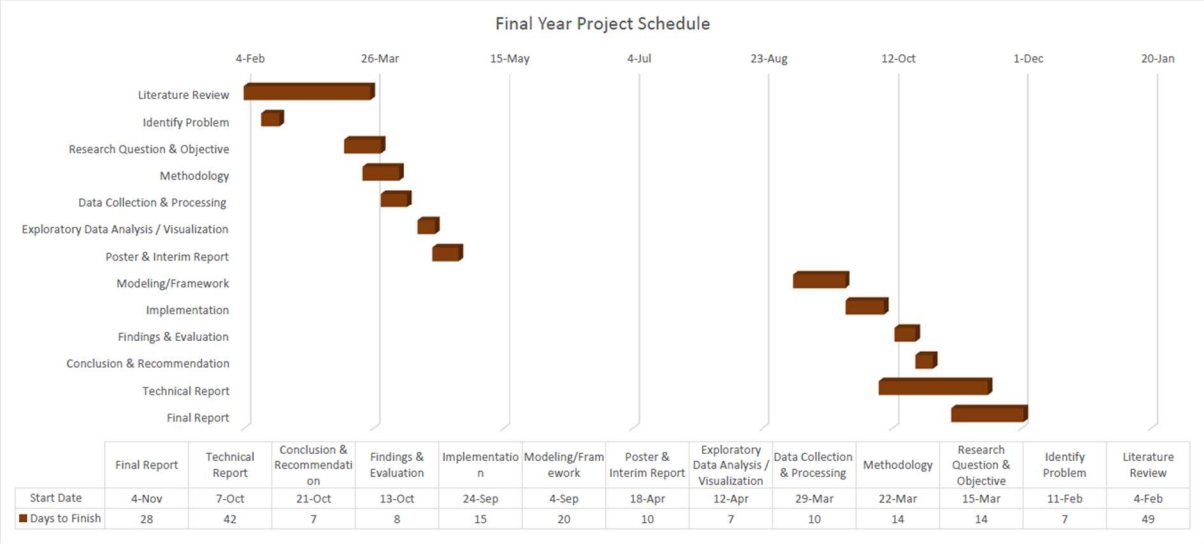
Tehrany, M. S., Pradhan, B., & Jebur, M. N. (2013). Spatial prediction of flood susceptible areas using rule based decision tree (DT) and a novel ensemble bivariate and multivariate statistical models in GIS. *Journal of Hydrology*, *504*, 69–79. https://doi.org/10.1016/j.jhydrol.2013.09.034

Tehrany, M. S., Pradhan, B., & Jebur, M. N. (2014). Flood susceptibility mapping using a novel ensemble weights-of-evidence and support vector machine models in GIS. *Journal of Hydrology*, *512*, 332–343. https://doi.org/10.1016/j.jhydrol.2014.03.008

Zhong, C., Malinen, M., Miao, D., & Fränti, P. (2015). A fast minimum spanning tree algorithm based on K-means. *Information Sciences*, *295* (October), 1–17. https://doi.org/10.1016/j.ins.2014.10.012

APPENDICES

APPENDIX A: GANTT CHART (Project Schedule)



	Final Report	Technical Report	Conclusion & Recommendation	Findings & Evaluation	Implementation	Modeling/Framework	Poster & Interim Report	Exploratory Data Analysis / Visualization	Data Collection & Processing	Methodology	Research Question & Objective	Identify Problem	Literature Review
Start Date	4-Nov	7-Oct	21-Oct	13-Oct	24-Sep	4-Sep	18-Apr	12-Apr	29-Mar	22-Mar	15-Mar	11-Feb	4-Feb
Days to Finish	28	42	7	8	15	20	10	7	10	14	14	7	49

APPENDIX B: DATASET SCREENSHOT

	A	B	C	D	E	F	G	H
1	TOWN	DISTRICT	POPULATION	DISTANCE FROM SEA (KM)	ALTITUDE/ELEVATION (M)	ANNUAL RAINFALL (MM)	LATITUDE	LONGITUDE
2	AIR PUTIH	KEMAMAN	1937	13.47	39.93	2800	4.2684	103.2102
3	CUKAI	KEMAMAN	82425	1.83	12	2874	4.2500	103.4167
4	JERTIH	BESUT	13526	12.51	2.74	3046	5.7336	102.4897
5	JABI	SETIU	13973	11.26	40.84	3129	5.6779	102.576
6	PERMAISURI	SETIU	6054	12.03	29.87	3256	5.5224	102.7485
7	LANGKAP	SETIU	3457	19.9	60.96	3189	5.3702	102.823
8	BATU RAKIT	SETIU	29764	3.61	10.97	2797	5.4500	103.05
9	TEPUH	KUALA NERUS	96077	6.96	22.86	2461	5.3576	103.0638
10	ALUR LIMBAT	MARANG	25747	13.69	252.98	2225	5.1939	103.0735
11	RU RENDANG	MARANG	33144	0.28	9.75	2342	5.2211	103.1981
12	WAKAF MEMPELAM	MARANG	157764	6.16	37.8	2395	5.2871	103.112
13	BUKIT PAYUNG	MARANG	52032	10.29	111.86	2453	5.2328	103.1006
14	PENGKALAN BERANGAN	MARANG	7584	12.36	45.72	2210	5.1066	103.1394
15	LADANG	KUALA TERENGGANU	130445	0.38	10.67	2473	5.3293	103.1507
16	KUALA BESUT	BESUT	16269	0.7	6.1	2805	5.8330	102.5551
17	KOTA PUTERA	BESUT	126091	1.01	8.53	2864	5.2977	103.0851
18	HULU BESUT	BESUT	5000	56.9	40.84	2907	5.4589	102.4969
19	BANDAR	KUALA TERENGGANU	285065	0.78	9.14	2466	5.3302	103.1408
20	BULUH GADING	KUALA NERUS	150638	5	10.67	2315	5.3364	103.0908
21	BATU BURUK	KUALA TERENGGANU	125259	0.76	10.97	2490	5.3150	103.1588
22	SURA	DUNGUN	22803	1.66	2.74	2517	4.7179	103.4361
23	MANIR	HULU TERENGGANU	131627	7.78	7.62	2524	5.3073	103.0837
24	SEKAYU	HULU TERENGGANU	2755	36	75.9	2888	4.9646	102.9677
25	KUALA BERANG	HULU TERENGGANU	15167	26.46	48.77	2225	5.0734	103.0114

APPENDIX C: PCA & K-Means Algorithm (R Source Code)

```
library(dplyr)
library(cluster)
library(ggplot2)
library(NbClust)
library(factoextra)

setwd("C:/Users/user/Desktop/Sem 8/FYP Model/")

data <- read.csv("Terengganu Data.csv")

View(data)

data <- data %>% rename(DISTANCE_FROM_SEA = DISTANCE.FROM.SEA..KM.)
data <- data %>% rename(ALTITUDE = ALTITUDE.ELEVATION..M.)
data <- data %>% rename(ANNUAL_RAINFALL = ANNUAL.RAINFALL..MM.)

tdata <- data[3:6]

typeof(tdata$POPULATION)
typeof(tdata$DISTANCE_FROM_SEA)
typeof(tdata$ALTITUDE)
typeof(tdata$ANNUAL_RAINFALL)


head(tdata)

#apply Principal Component Analysis
pca <- prcomp(tdata, center = TRUE, scale = TRUE)

summary(pca)
pca
attributes(pca)
pca$x

fviz_eig(pca)

pca_data <- data.frame(pca$x[,1:2])

head(pca_data)
View(pca_data)

#determining optimal no. of clusters
set.seed(123)

#elbow method using within sum of squared distance
v <- vector()

for (i in 1:10)
  v[i] <- sum(kmeans(pca_data, i)$withinss)
plot(1:10, v, type = "b", main = paste('Clusters'),
     xlab = "No. of Clusters", ylab = "WSS")

fviz_nbclust(pca_data, kmeans, method = "wss") +
  geom_vline(xintercept = 6, linetype = 2) +
  labs(subtitle = "WithinSS")

#gap-stat method
fviz_nbclust(pca_data, kmeans, nstart = 10, method = "gap_stat", nboot =
50) +
```

```
  labs(subtitle = "Gap-Stat")

#silhouette method
fviz_nbclust(pca_data, kmeans, method = "silhouette") +
  labs(subtitle = "Silhouette")


#apply k-means clustering model
kmodel <- kmeans(pca_data, 6)

attributes(kmodel)
kmodel
head(kmodel)
kmodel$cluster
kmodel$size
kmodel$withinss

result <- data.frame(data, kmodel$cluster)
head(result)
View(result)
result <- result %>% rename(Cluster = kmodel.cluster)


#visualization
names(result)
b <- ggplot(result, aes(x=factor(Cluster))) +
  geom_bar(stat="count", width=0.7, fill="steelblue") +
  labs(x = "Cluster", y = "No. of Towns", title = "K-Means Result")
b

clusplot(pca_data, kmodel$cluster, main = "2D Reperesentation of Clusters",
         color = TRUE, shade = TRUE, labels = 2, lines = 0)

fviz_cluster(kmodel, pca_data, ellipse.type = "norm")

fviz_cluster(kmodel, pca_data, palette = "Set2", ggtheme = theme_minimal())


#write data-frame into a new csv file
write.csv(result, file = "result.csv", row.names=FALSE)

#analysis of clusters
r <- data.frame(tdata, Cluster = result$Cluster)
r %>% filter(Cluster == 1)
r %>% filter(Cluster == 2)


analysis <- data.frame(CLuster = numeric(), Population = double(),
Distance_from_sea = double(),
                       Altitude = double(), Annual_rainfall = double())

for (i in 1:6)
  analysis[i,] <- c(i,
              mean(result$POPULATION[result$Cluster == i]),
              mean(result$DISTANCE_FROM_SEA[result$Cluster == i]),
              mean(result$ALTITUDE[result$Cluster == i]),
              mean(result$ANNUAL_RAINFALL[result$Cluster == i]))

analysis
```

APPENDIX D: SVM Model (Python Source Code)

```
import pandas as pd
import numpy as np
import sklearn
import matplotlib
from sklearn import svm
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.model_selection import cross_validate

np.random.seed(123)

data = pd.read_csv("Risk Data.csv", sep = ",")
#print(data.head())

data = data[["POPULATION", "DISTANCE_FROM_SEA", "ALTITUDE",
"ANNUAL_RAINFALL", "Risk"]]
#print(data.head())
target = "Risk"

x = np.array(data.drop([target], 1))
y = np.array(data[target])

x_train, x_test, y_train, y_test =
sklearn.model_selection.train_test_split(x, y, test_size = 0.2,
random_state= 123)

classes = ["High", "Medium", "Low"]

model = svm.SVC(kernel = "linear", C = 2)

model.fit(x_train, y_train)

prediction = model.predict(x_test)

acc = metrics.accuracy_score(y_test, prediction)

print(acc)

#ten-fold cross validation
cv_predict = cross_val_predict(model, x, y, cv = 10)
cv_score = cross_val_score(model, x, y, cv = 10)
avg_score = np.mean(cv_score)
print(cv_predict)
print(cv_score)
print(avg_score)

conf_matrix = metrics.confusion_matrix(y, cv_predict)
print(conf_matrix)
evaluation = metrics.classification_report(y, cv_predict)
print(evaluation)
metrics.confusion_matrix()
```

APPENDIX E: SHORTEST-PATH ALGORITHM (Python Source Code)

```
import itertools as it
import math
import pandas as pd
import random
import numpy as np
import matplotlib.pyplot as plt


def distance(x1,y1,x2,y2):
    return math.sqrt((x1-x2)**2 + (y1-y2)**2)

def convertion(a1,b1,Data):
    x_list = []
    y_list = []
    points = Origin.values
    for p in range (0, len(points)):
        a = (-a1)+ points[p-1][0]
        b = (-b1)+ points[p-1][1]
        x_list.append(a)
        y_list.append(b)

    c_list = {'X_coordinate':x_list,'Y_coordinate':y_list}

    df = pd.DataFrame(c_list, columns=['X_coordinate', 'Y_coordinate'])
    export_csv = df.to_csv(r'/Users/Nafees/Desktop/export_dataframe.csv',
index=None, header=True)



def shortest_path(lbest, ibest, allpossibletours):
    for i in range(len(allpossibletours)):
        l = totaldistance(allpossibletours[i])
        if l < lbest:
            lbest = l
            ibest = i
    print('Total distance of shortest path {}
meters'.format(np.around(lbest, decimals=3)))
    print("Shortest path direction is ->", allpossibletours[ibest])
    return
def longest_path(lbest,ibest,allpossibletours):
    for i in range(len(allpossibletours)):
        l = totaldistance(allpossibletours[i])
        if l > lbest:
            lbest = l
            ibest = i
    print('Total distance of longest path {}
meters'.format(np.around(lbest, decimals=3)))
    print("longest path direction is ->", allpossibletours[ibest])

def random_path(tour):
    u = list(it.permutations(tour))
    ulen = len(u)
    random_num = random.randint(1,ulen)
    l = totaldistance(allpossibletours[random_num])
    print('Total distance of random path {} meters'.format(np.around(l,
decimals=3)))
    print("longest path direction is ->", allpossibletours[random_num])
def totaldistance(tour):
    d=0
```

```
    for i in range(1,len(tour)):
        x1 = Cities[tour[i-1]][0]
        y1 = Cities[tour[i-1]][1]
        x2 = Cities[tour[i]][0]
        y2 = Cities[tour[i]][1]
        d = d+distance(x1,y1,x2,y2)
    x1 = Cities[tour[len(tour)-1]][0]
    y1 = Cities[tour[len(tour)-1]][1]
    x2 = Cities[tour[0]][0]
    y2 = Cities[tour[0]][1]
    d = d + distance(x1, y1, x2, y2)
    return d

Origin = pd.read_csv('/Users/Nafees/Desktop/Book1.csv')
convertion(5,6,Origin)
Data = pd.read_csv('/Users/Nafees/Desktop/export_dataframe.csv')
if Data.empty:
    raise Exception("""At least one of coords must be specified.""")
else:
    Cities = Data.values
    x = Cities.shape
    if x == (1,2):
        raise Exception("""Please enter another coords""")
    else:
        n = len(Data)
        tour = np.arange(n)
        np.random.shuffle(tour)
        allpossibletours = list(it.permutations(tour))
        lbest = int(input("Please enter the total area of affected land in
Squre Kilometer: "))*1000
        ibest = 0
        long = 0
        shortest_path(lbest,ibest,allpossibletours)
        print(' ')
        print(' ')
        longest_path(long,ibest,allpossibletours)
        print(' ')
        print(' ')
        random_path(tour)
```

END OF FINAL REPORT