A Project Report

on

PREDICTION OF GENESIS TO DECAY ROUTE OF WIND STORM

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

by

V. R. Tejaswi (16WH1A1213)

P. V. Niharika (16WH1A1239)

K. Meghana (16WH1A1241)

Under the esteemed guidance of

Ms. M. L. Prasanthi
Associate Professor

Department of Information Technology

BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN

(Accredited by NBA & NAAC '**A**' Grade, Approved by AICTE, and
Affiliated to JNTUH, Hyderabad)

Bachupally, Hyderabad – 500090

April 2020

DECLARATION

We hereby declare that the work presented in this project entitled **"PREDICTION OF GENESIS TO DECAY ROUTE OF WIND STORM"** submitted towards completion of the major project in IV year of B.Tech IT at "BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN**"**, is an authentic record of our original work carried out under the esteem guidance of

Ms. M. L. Prasanthi, Associate Professor, IT department.

V. R. Tejaswi
(16WH1A1213)

P. V. Niharika
(16WH1A1239)

K. Meghana
(16WH1A1241)

BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN

(Accredited by NBA & NAAC 'A' Grade, Approved by AICTE, and Affiliated to JNTUH, Hyderabad)

Bachupally, Hyderabad – 500090

Department of Information Technology

Certificate

This is to certify that the Project report on "**Prediction of Genesis to Decay route of Wind Storm**" is a bonafide work carried by **V. R. Tejaswi(16WH1A1213), P. V. Niharika(16WH1A1239), K. Meghana(16WH1A1241)** in the partial fulfillment for the award of B.Tech degree in **Information Technology, BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Internal Guide
Ms. M. L. Prasanthi
Associate Professor
Department of IT

Head of the Department
Dr. ArunaRao S L
Professor and HOD
Department of IT

External Examiner

Acknowledgements

We would like to express our sincere thanks to **Dr. K. V. N. Sunitha, Principal**, BVRIT HYDERABAD, for providing the working facilities in the college.

Our sincere thanks and gratitude to **Dr**. **ArunaRao S L, Head**, Dept. of IT, BVRIT HYDERABAD for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Ms. M. L. Prasanthi, Associate Professor, Department of IT,** BVRIT HYDERABAD for her constant guidance, encouragement and moral support throughout the project.

Finally, We would also like to thank our project co-coordinator, all the faculty and staff of IT Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

V. R. Tejaswi
(16WH1A1213)
P. V. Niharika
(16WH1A1239)
K. Meghana
(16WH1A1241)

ABSTRACT

The world has witnessed an escalation of devastating wind storms over the last three decades. Every year, the time between June 1 and November 30 signifies the North Atlantic Storm season. During this period, the warm waters in the Atlantic give birth to tropical cyclones, and few of these tropical cyclones end up making landfall causing major casualties and loss of property. Thousands of people suffer every year due to the wind storms which occur mainly due to the Atlantic Ocean.

Prediction of genesis to decay route of wind storm is a model which helps in predicting the path or route of a wind storm given the genesis which is the starting point of the wind storm. To build this model, Clustering is done on the available Atlantic wind storm dataset. Then the Markov chain is applied on all the clusters that are formed. By applying Markov chain on all the clusters, the path of the wind storm is predicted given the genesis point. The accuracy of predicting their trajectory paths is critical to reduce economic loss and save human lives.

LIST OF FIGURES

TABLE OF CONTENTS

1. INTRODUCTION

1.1 OBJECTIVE

Windstorm, a wind that is strong enough to cause at least light damage to trees and buildings and may or may not be accompanied by precipitation. Wind speeds during a windstorm typically exceed 55 km per hour. Wind damage can be attributed to gusts short bursts of high-speed winds or longer periods of stronger sustained winds.

The world has witnessed an escalation of devastating wind storms over the last three decades. Every year, the time between June 1 and November 30 signifies the North Atlantic Storm season. During this period, the warm waters in the Atlantic give birth to tropical cyclones, and few of these tropical cyclones end up making landfall causing major casualties and loss of property. Build a windstorm track prediction model using the Atlantic Hurricane Database The prediction of windstorm path is done with the developed model, and then the model results are compared with the actual path traversed by storm.

By using the Atlantic Windstorm Database. The dataset provides location details Latitude & Longitude of every storm at every 3-hrs interval from the point of genesis to decay. Short-term forecasting of the windstorm path is relatively straightforward. A long-term prediction model is built that can predict the path of the windstorm few days or weeks in advance.

With the help of historical data, the model should be able to enroute the entire path of the storm from the starting genesis point till the decay point of the storm. This model should also locate the places that will be most affected by the storm in order to take all the precautionary measures needed. Accurate prediction of frequency, severity, and landfall locations are all important for mitigating the risk from these costly disasters. This will be helpful for every individual to know the whether that particular wind storm might cause damage to them or not. If yes then they can take precautionary measures before itself and can protect themselves and reduce their damage loss.

1.2 PROBLEM IN EXISTING SYSTEM

The existing system is able to do the analysis of wind storms. Given a month it will predict the chances of a storm occurring in that particular month and predicts the type of storm based on the intensity of the wind.

The problem in the existing system is that it can only predict the very next location the storm will be travelling to from the genesis point where the storm has begun. And it cannot give the names of the locations that are going to be affected the most by the storm. Returning the next location coordinates that the storm will be travelling to is good but it will not be helpful as no precautionary measures can be taken by the local people on an immediate notice.

1.3 SOLUTION

Prediction of the very next location that the storm will be travelling to is not good enough. The prediction of the genesis to decay route of a wind storm will help in predicting the entire path or route of a wind storm. And also predicting the type of windstorm based on the intensity of it helps in predicting the damage that could be caused by the storm. It also helps the people living in the surrounding areas take certain precautionary measures.
Returning the location names instead of the longitude and latitude values will make the model much more sophisticated and understandable for the users. The accuracy of predicting their trajectory paths is critical to reduce economic loss and save human lives.

1.4 FEATURES

Suggestion forums are dedicated forums which are used to provide suggestions for improvement in an entity. The data is collected from feedback posts on Universal Windows Platform. Often people tend to provide the context in suggestion posts, which gets repetitive in the case of large number of posts under the same topic. Suggestion mining can act as automatic summarisation in this use case, by identifying the sentences where a concrete suggestion is expressed. We observe that the datasets derived from this domain contain a relatively larger number of positive class instances, as compared to the other domains. The sentences are automatically split using Stanford's parser.

2. LITERATURE SURVEY

2.1 APPROACH

The windstorm dataset provides location details, latitude and longitude of every storm at every 3-hrs interval from the point of genesis to decay. Capturing the spatial patterns hidden in the historical data is done to get the domain knowledge which will help in predicting the full path traversed by the hurricane. In order to predict the path historical data analysis is very crucial. But the data obtained may contain incompleteness, inconsistency, or might lack in certain behaviors and all these issues can be resolved by preprocessing.

The historical data is a collected data about past events and circumstances pertaining to a particular subject. Data analysis is performed on this data. Data analysis is a practice in which raw data is ordered and organized so that useful information can be extracted from it. The process of organizing and thinking about data is key to understanding what the data does and does not contain.

The climatology genesis points, sea surface temperature, energy, etc. determine the trajectory of hurricanes and the probability of making landfall. In order to account for the climatology, clustering the hurricanes into four groups is done. A clustering technique called agglomerative clustering is applied on the data. The agglomerative clustering is also called a bottom-up approach. It considers each observation as its own cluster, and pairs of clusters are merged as one moves up the hierarchy.

After clustering, the Markov model is developed and used to predict the plausible paths a windstorm could take from its genesis point. A Markov chain is a mathematical system that experiences transitions from one state to another according to certain probabilistic rules. The defining characteristic of a Markov chain is that no matter how the process arrived at its present state, the possible future states are fixed. In other words, the probability of transitioning to any particular state is dependent solely on the current state and time elapsed.

2.2 INFORMATION GATHERING

Wind storms are some of the most powerful and destructive natural disasters and can cause billions of dollars of damage as well as the loss of thousands of lives. Understanding them, however, has proven extremely difficult. Predicting exactly where, when, and how a hurricane will develop, intensify, and propagate has been notoriously problematic and inaccurate. Using data collected from advanced tools and mathematical models, extremely complex computer models have been developed. These models are utilized by weather agencies around the world to forecast weather patterns, storms, and extreme cases such as hurricanes. As advanced as these models have become, inaccuracies remain.

SAFFIR-SIMPSON wind scale research will be utilizing the Saffir-Simpson Wind Scale, which is the scale that categorizes hurricanes by their intensities (wind speeds) by the following parameters (many scales use knots as the speed unit, but in this research miles per hour will be used)

TABLE 1 Saffir-Simpson scale of hurricane intensity.

Type	Category	Pressure (mb)	Winds (mph)	Winds (kph)	Surge meters (ft)
Tropical Depression	TD	-	< 39	< 64	-
Tropical Storm	TS	-	39-73	64-118	-
Hurricane	1	> 980	74-95	119-153	1.2-1.5 (3.9 – 5.0)
Hurricane	2	965-980	96-110	154-177	1.6-2.4 (5.1 – 8.0)
Hurricane	3	945-965	111-130	178-209	2.5-3.6 (8.1 – 12.0)
Hurricane	4	920-945	131-155	210-250	3.7-5.4 (12.1 – 17.7)
Hurricane	5	< 920	> 155	> 250	> 5.4 (> 17.7)

Figure 2.2.0: Saffir-Simpson Scale of Intensity

3. REQUIREMENT SPECIFICATIONS

3.1 SOFTWARE REQUIREMENTS

The software requirements specify the use of all required software products like data management system. The required software product specifies the numbers and version. Each interface specifies the purpose of the interfacing software as related to this software product.

- Operating system : Windows 7/10
- Coding Language : Python 3
- IDE : Jupyter Notebook/ Google Colab

3.1.1 Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more. The Jupyter Notebook can be used to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

pip3 install jupyter

We can install Jupyter notebook either by pip or else we can directly install by installing Anaconda. Before installing Jupyter we need to ensure that we have the latest version of pip.

3.1.2 Numpy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

3.1.3 Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tools using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

The various features of Pandas include Fast and efficient Data Frame object with default and customized indexing tools for loading data into in-memory data objects from different file formats, data alignment and integrated handling of missing data, reshaping and pivoting of date sets, label-based slicing, indexing and sub setting of large data sets, columns from a data structure can be deleted or inserted and time Series functionality.

3.1.4 Time

Python time method **mktime()** is the inverse function of localtime(). Its argument is the struct_time or full 9-tuple and it returns a floating point number, for compatibility with time(). This method is the inverse function of time.localtime() which converts the time

expressed in seconds since the epoch to a time.struct_time object in local time.

In Python, date, time and datetime classes provide a number of functions to deal with dates, times and time intervals. Date and datetime are an object in Python, so when you manipulate them, you are actually manipulating objects and not string or timestamps. Whenever you manipulate dates or time, you need to import the datetime function.

3.1.5 Geopy

Geopy makes it easy for Python developers to locate the coordinates of addresses, cities, countries, and landmarks across the globe using third-party geocoders and other data sources. When dealing with lat/long values there would be use cases to calculate the distance between two points or places by evaluating the distance between their lat/long.

There are three different abstractions which are used for calculating this distance, Pythagoras theorem works fine for the smaller distance between two points and for more accurate solution geodesic distanc**e** is used using python's geopy module and haversine formula or MYSQL st_distance_sphere is used for a spherical distance.

Geodesic Distance is the shortest distance measured along the ellipsoidal surface of the earth. The tool named geodesic distance calculates distance between two points which are defined by geodetic coordinates in terms of latitude and longitude.

3.1.6 Defaultdict

The defaultdict works exactly like a python dictionary, except for it does not throw KeyError when you try to access a non-existent key.

Instead, it initializes the key with the element of the data type that you pass as an argument at the creation of defaultdict. The data type is called default_factory.

The defaultdict is faster and simpler with small data sets. It is faster for larger data sets with more homogenous key sets. The setdefault has an advantage over defaultdict if we consider more heterogeneous key sets.

3.1.7 Matplotlib

The matplotlib.pyplo**t** is a plotting library used for two dimensional graphics in python programming language. It can be used in python scripts, shell, web application servers and other graphical user interface toolkits.

There are several toolkits which are available that extend python matplotlib functionality. Some of them are separate downloads, others can be shipped with the matplotlib source code but have external dependencies.

There are various plots which can be created using python matplotlib. Some of them are bar graph, histogram, scatter plot, area plot, pie plot

Using matplotlib the visualization of data can be done in a better way. Using this library publication quality plots can be developed with just a few lines of code and also create interactive figures that can zoom, pan, update.

Customization by taking full control of line styles, font properties, axes properties is possible and also export and embed to a number of file formats and interactive environments. Exploinge tailored functionality provided by third party packages is possible.

3.1.8 Cartopy

Cartopy is a Python package designed for geospatial data processing in order to produce maps and other geospatial data analyses. Cartopy makes use of the powerful PROJ.4, NumPy and Shapely libraries and includes a programmatic interface built on top of Matplotlib for the creation of publication quality maps.

Key features of cartopy are its object oriented projection definitions, and its ability to transform points, lines, vectors, polygons and images between those projections.Cartopy is especially useful for large area or small scale data, where Cartesian assumptions of spherical data traditionally break down. It is suitable to be used in a variety of scientific fields and has an active development community.

3.1.9 Agglomerative clustering

Agglomerative clustering also called Hierarchical Agglomerative Clustering, or HAC is a "bottom up" type of hierarchical clustering. In agglomerative clustering, each data point is defined as a cluster. Pairs of clusters are merged as the algorithm moves up in the hierarchy.

In agglomerative clustering, each document is treated as a single cluster at the beginning of the algorithm. After that, clusters can be combined through a variety of methods. They all involve calculating dissimilarities between objects; Exactly how that dissimilarity is calculated can vary. The most commonly used are single linkage, complete linkage, average linkage, centroid method and ward's method.

Scikit-learn formerly scikits.learn and also known as sklearn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

3.1.10 reverse_geocoder

Geocoding is the process of translating physical addresses into latitudes and longitudes in order to enable your data to be displayed and analyzed in a geographic context on top of a map.Content Manager enables geocoding of data with street level precision by importing a table containing addresses or by geocoding addresses manually. Load the data to be geocoded and then visualize it on top of the map.

Reverse Geocoding is the process of translation of geographic coordinates latitudes and longitudes into physical addresses. One can geocode addresses from data files that are saved locally in the computer. Geocodes can be retrieved from addresses contained in a spreadsheet, or can be entered manually. The imported data is geocoded automatically and can then be visualized in a geographic context in order to assess the geographic quality of your data.

3.2 HARDWARE REQUIREMENTS

The hardware requirement specifies each interface of the software elements and the hardware elements of the system. These hardware requirements include configuration characteristics.

- Hard Disk : 100 GB.
- Monitor : 15 VGA Color.
- RAM : 2 GB.

4. DESIGN OF THE SYSTEM

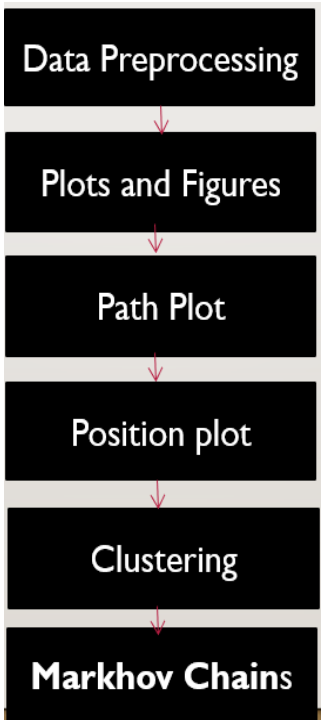

Figure 4.1.0: Design of the system

4.1 Data Preprocessing

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a

proven method of resolving such issues.

In Real world data are generally incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data. Noisy: containing errors or outliers. Inconsistent: containing discrepancies in codes or names.

Our dataset at the beginning before normalization looks like the one in the figure 4.1.

Out[2]:

	ID	Name	Date	Time	Status	Latitude	Longitude	Maximum Wind	Minimum Pressure
44827	AL072006	FLORENCE	20060913	1200	EX	45.5N	55.6W	70	967
22949	AL041952	BAKER	19520907	1800	HU	40.4N	60.1W	95	969
18184	AL031936	UNNAMED	19360626	1800	TS	26.2N	95.5W	40	-999
23583	AL031954	ALICE	19540625	1400	HU	25.0N	97.6W	95	-999
32598	AL101977	DOROTHY	19770928	1200	HU	35.5N	59.7W	65	988

Figure 4.1.1: Dataset Before Preprocessing

:[6]:

	ID	Name	Status	Latitude	Longitude	Maximum Wind	Minimum Pressure	TIME
49100	AL122015	KATE	EX	41.3N	50.4W	55	981.0	2015-11-12 12:00:00
49101	AL122015	KATE	EX	41.9N	49.9W	55	983.0	2015-11-12 18:00:00
49102	AL122015	KATE	EX	41.5N	49.2W	50	985.0	2015-11-13 00:00:00
49103	AL122015	KATE	EX	40.8N	47.5W	45	985.0	2015-11-13 06:00:00
49104	AL122015	KATE	EX	40.7N	45.4W	45	987.0	2015-11-13 12:00:00

Figure 4.1.2: Dataset After Preprocessing

Before data preprocessing (fig 4.1.1) there are few missing values in the data set and which are represented by -999. These kinds of missing values can cause noise so they are replaced by the mean value obtained for that column (fig 4.1.2). And even the time and date are changed into a more understandable format by merging them into a single column.

4.2 Data Analysis

The process of evaluating data using analytical and logical reasoning to examine each component of the data provided. This form of analysis is just one of the many steps that must be completed when conducting a research experiment. Data from various sources is gathered, reviewed, and then analyzed to form some sort of finding or conclusion. There are a variety of specific data analysis methods, some of which include data mining, text analytics, business intelligence, and data visualizations.

Data analysis is important in business to understand problems facing an organization, and to explore data in meaningful ways. Data in itself is merely facts and figures. Data analysis organizes, interprets, structures and presents the data into useful information that provides context for the data.

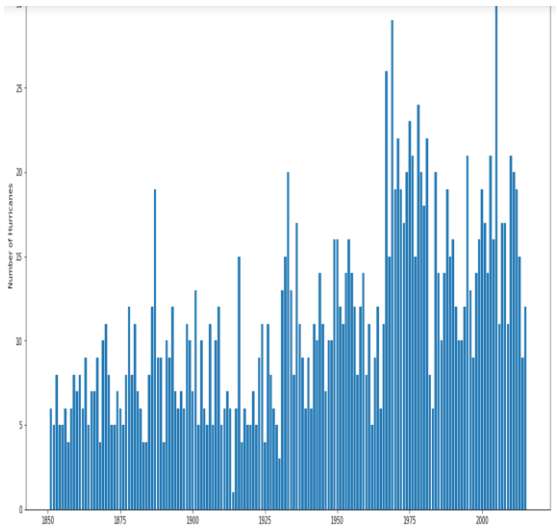

Figure 4.2.1 Frequency of storm occurrence in a year

From the above bar graph (fig 4.2.1) analysis can be made on the frequency of storms that have occurred in the past years and with the help of this one can try to recognize if there is any pattern in these series of storms occurring. And also identify the years in which most intensive windstorms have occurred and try to predict the damage that could be caused by that kind of storm if it occurred now or in future.

4.3 CLUSTERING

Cluster is a group of objects that belongs to the same class. In other words, similar objects are grouped in one cluster and dissimilar objects are grouped in another cluster. The climatology (genesis points, sea surface temperature, energy, etc) determine the trajectory of hurricanes and the probability of making landfall. In order to account for the climatology, we will cluster the hurricanes into four groups. The clustering analysis is made to gain some valuable insights from the data by seeing what groups the data points fall into when the clustering algorithm is applied.

The Agglomerative Hierarchical Clustering Technique is being used which is one of the clustering techniques, initially each data point is considered as an individual cluster. At each iteration, the similar clusters merge with other clusters until one cluster or K clusters are formed.

The basic algorithm of Agglomerative is straight forward.

- Compute the proximity matrix
- Let each data point be a cluster
- Repeat: Merge the two closest clusters and update the proximity matrix
- Until only a single cluster remains

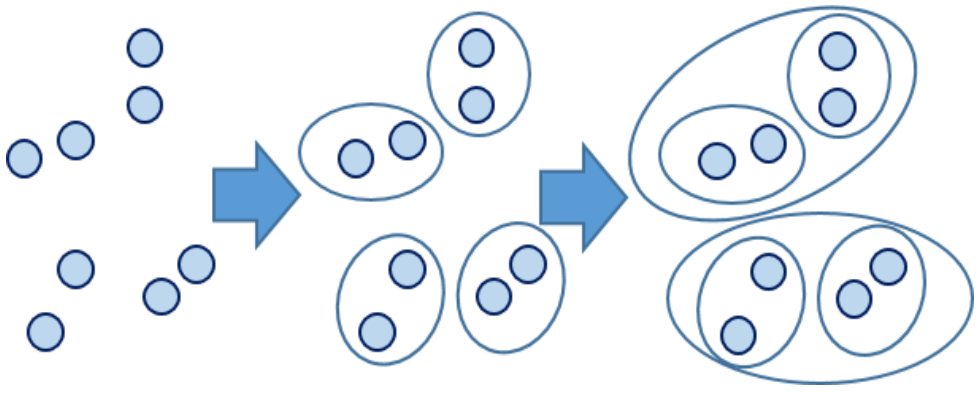

Figure 4.3.1: Clustering

4.4 Markov Chain Model

A Markov chain is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event. The changes of state of the system are called transitions. The probabilities associated with various state changes are called transition probabilities. The process is characterized by a state space, a transition matrix describing the probabilities of particular transitions, and an initial state or initial distribution across the state space.

The Markov Chain model is the best model that can be used in order to predict the entire route of the storm from start point to the decay point of it. With the help of historical data, it was possible to construct Markov Chains that would establish probabilities of a storm's movement as it progressed through its life cycle. Analyzing the data from every major storm, one large Markov Chain was constructed.

In particular the random walk example, could provide a useful tool in predicting how a certain storm will behave. When one considers how a windstorm intensifies, it can be interpreted as changing from one state to the next. When enough windstorms are analyzed, perhaps Markov Chains can provide effective analyses on intensification of particular storms. Though it is not always the case, a storm will most often strengthen or weaken the state immediately adjoining the state in which it currently occupies.

For example, a Category 2 storm will most often strengthen to a Category 3 storm, or weaken to a Category 1 storm (though a storm's intensity will decrease rapidly over land or colder water). Thus, treating storm intensification as a random walk was a logical way to begin modeling its behavior.

After a storm has formed that shows significant potential for becoming dangerous, various organizations such as the National Hurricane Center, a subdivision of the National Weather Service and the National Oceanic and Atmospheric Administration begin tracking it. Data received via satellite, buoy, radar, or other technologies is then collected to provide up to date and accurate assessments of the storm.

This data is recorded every 3 hours and can be found online. Using the recorded data relative to each storm's intensity, it was possible to construct a Markov Chain for each individual storm from the past years. The construction of each Markov chain went as follows:

A storm begins in a state, which is almost always as a Tropical Depression; the next data point would be the first entry on the Markov chain. If the storm remained a Tropical Depression, then the matrix coordinates corresponding to both a row and column corresponding to a Tropical Depression would be increased by 1.

If the storm increased to a Tropical Storm, then the entry in the matrix would be in the Tropical Depression row and tropical storm column. In some cases, storms had at least one data point that jumped a category. Say, for example, a Category 1 storm increased to a Category 3 storm within one 3-hour window, then the coordinates corresponding to the Category 1 row and the Category 3 column would be increased by one. Rows with 4 or more entries are reflective of this occurrence.

After all the data points were analyzed, the rows were divided by the total number of entries in them. These are the probabilities that that particular storm will be in that particular state given the data point immediately before it.

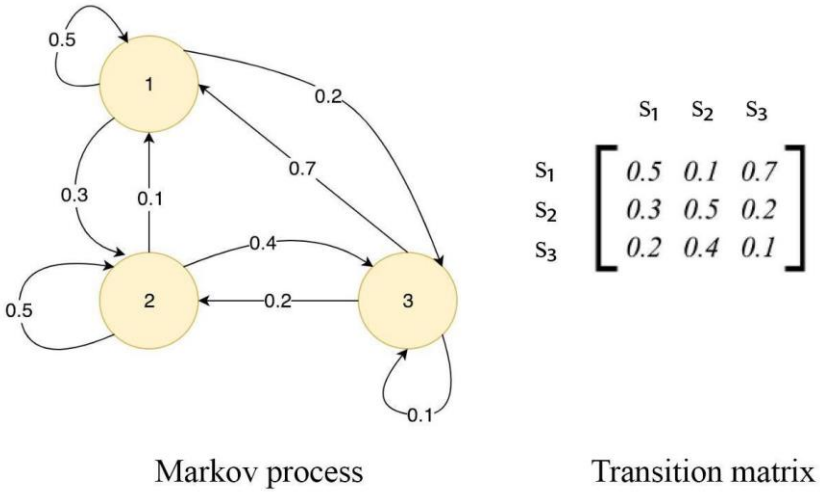

Figure 4.4.1: Markov process and Transition matrix

5.MODULES

5.1 ANALYSIS MODULE

The purpose of Data Analysis is to extract useful information from data and take the decision based upon the data analysis. Where breaking a whole data into its separate components for individual examination is done. Data analysis is a process for obtaining raw data and converting it into information useful for decision-making by users. Data is collected and analyzed to answer questions, test hypotheses or disprove theories.

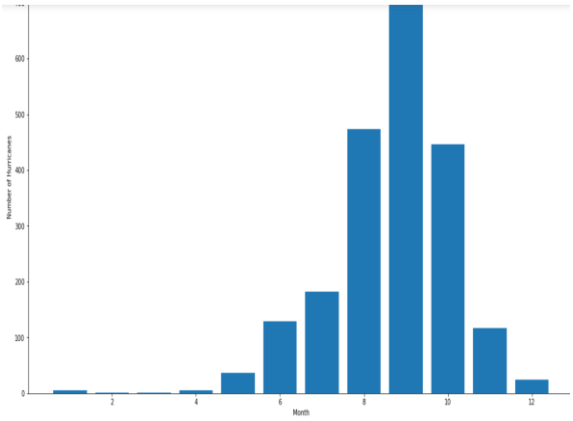

Figure 5.1.1: Frequency of storm occurrence in a month

The above fig 5.1 illustrates the chances of a windstorm to occur in a particular month. By plotting, analysis of the data can be done and useful information can be extracted.

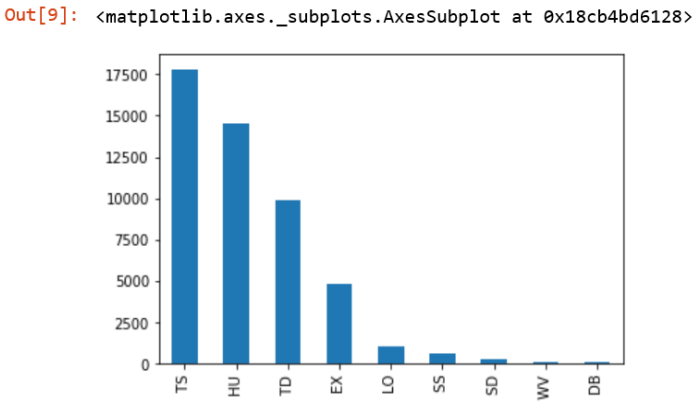

Figure 5.1.2: Frequency of various kinds of storms

From the fig 5.1.2 analysis can be made that tropical storms are the most frequently occurring kind of storms among the nine of them.

5.2 CLUSTERING MODULE:

Cluster is a group of objects that belongs to the same class. In other words, similar objects are grouped in one cluster and dissimilar objects are grouped in another cluster. We use agglomerative clustering techniques. Agglomerative clustering also called Hierarchical Agglomerative Clustering, or HAC is a "bottom up" type of hierarchical clustering. In agglomerative clustering, each data point is defined as a cluster (fig 5.2.1). Pairs of clusters are merged as the algorithm moves up in the hierarchy. In order to account for the climatology, we will cluster the hurricanes into four groups.

Storms that originate in the East Atlantic near to the equator are categorized under one group. These storms have the time to gather energy over warm waters, and they usually curve up.

Similar to the previous group, the genesis points of these hurricanes are near to the equator as well — but more towards the west. The trajectory of these storms are typically straight paths towards certain regions.

The genesis of these storms is further from the Equator, and they are not very strong. Their trajectories typically curve up, and hardly make landfalls

The genesis of these storms is near the Gulf-of-Mexico. Given the proximity of the genesis points to the land, these storms do not have the time to gather energy, but have high probability of landfall.

For this purpose agglomerative clustering technique is used where it recursively merges the pair of clusters that minimally increases a given linkage distance.

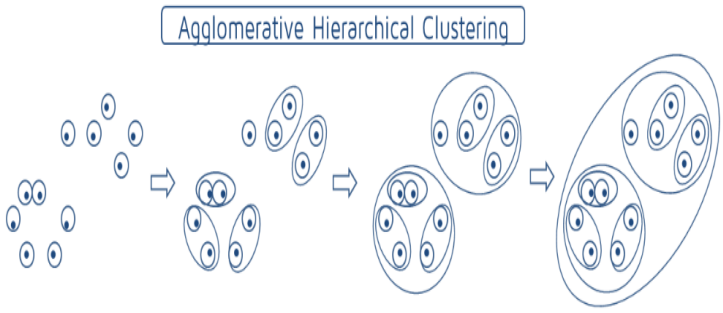

Figure 5.2.1: Agglomerative Hierarchical Clustering

5.3 PATH PREDICTION MODULE

The Markov transition model developed was used to predict the plausible paths the storm could take from its genesis point. An ensemble of ten simulations were run, and the resulting trajectories are shown in the plot below. If results from any simulation mimicked the path or is reasonably well then, that path will be chosen. In summary, the Markov transition model would be a simple way to predict the path of windstorms with the least amount of data, just the genesis points.

Using the simulation for path and intensity it is possible to be able to track a real storm and determine the threat that it poses to a particular geographical location. For example, say a storm becomes a Tropical Depression on any given grid space and the threat that storm poses to a given area needs to be determined. Analyzing the intensity with which each of these simulations strikes the area will give the expected intensity as well.

Though these models produced realistic simulations, there is no question it could continue to be improved upon. This model ignores much of the governing physics that drives hurricanes, as well as differences between seasons. It also does little to answer questions on variation between seasons. It simply takes all of the available data from historical data and produces simulations that act based on all of it. The best way to improve this model would be to become more and more specific with the information it analyzes.

By running simulations using constructed Markov Chains, it is possible to evaluate the general efficacy of this model. As seen with Hurricane Charley, a storm that begins far away from Florida, in this case in the Caribbean Sea off the coast of South America, would have a small chance to strike Florida. As seen in our simulations, however, the closer the storm moves toward the at-risk area, the higher the chances become that this area will be struck. This provides for realistic simulations, tracking, and predictions of hurricanes. Markov Chains can therefore be useful tools in hurricane study.

6. IMPLEMENTATION

```
import numpy as np
import pandas as pd
import time
import datetime
from time import mktime
from datetime import datetime
import geopy

all_data = pd.read_csv(r'/Users/megha/OneDrive/Desktop/MAJOR PROJECT/atlantic.csv')
all_data = all_data[['ID','Name','Date','Time',
'Status','Latitude','Longitude','Maximum Wind','Minimum Pressure']]
all_data.sample(5)
all_data = all_data.replace(to_replace=-999, value=0, regex=True)
x = all_data[["Minimum Pressure"]].mean(axis=0)
all_data = all_data.replace(to_replace=0, value=x, regex=True)
all_data.sample(5)

TIME = []
for entry in all_data.iterrows():
    try:
        if entry[1].Time == 0:
            TIME.append(datetime.strptime(str(entry[1].Date)+'
'+str(entry[1].Time),'%Y%m%d %H'))
        else:
            TIME.append(datetime.strptime(str(entry[1].Date)+'
'+str(entry[1].Time),'%Y%m%d %H%M'))
    except:
        TIME.append(np.nan)
        continue
```

```
len(TIME) == len(all_data)
all_data = all_data[['ID','Name','Status','Latitude',
'Longitude','Maximum Wind','Minimum Pressure']]
all_data['TIME'] = TIME
all_data.tail()

from collections import defaultdict
def get_year():
    years = defaultdict(list)
    for entry in all_data.iterrows():
        years[entry[1].TIME.year].append(entry[1].ID)
    yearz = {}
    each_yr = []
    for key in years:
        each_yr.append(key)
        yearz[key] = set(years[key])
    num_hurricanes = []
    for x in yearz.values():
        num_hurricanes.append(len(x))
    return (each_yr, num_hurricanes)

def get_months():
    months = defaultdict(list)
    for entry in all_data.iterrows():
        months[entry[1].TIME.month].append(entry[1].ID)
    m = {}
    each_month = []
    for key in months:
        each_month.append(key)
        m[key] = set(months[key])
    num_hurricanes = []
```

```
    for x in m.values():
        num_hurricanes.append(len(x))
    return (each_month, num_hurricanes)

import matplotlib.pyplot as plt
all_data["Status"].value_counts().plot(kind='bar')
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)
year_info = get_year()
plt.bar(year_info[0], year_info[1])
plt.xlabel('Year')
plt.ylabel('Number of Hurricanes')
plt.title('Number of Hurricanes in the Atlantic')
fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)
month_info = get_months()
plt.bar(month_info[0], month_info[1])
plt.xlabel('Month')
plt.ylabel('Number of Hurricanes')
plt.title('Hurricanes by Month')

import re
new_lat = []
new_long = []

for entry in all_data.iterrows():
    match = re.search('\d+.\d+', entry[1]['Latitude'])
    match2 = re.search('\d+.\d+', entry[1]['Longitude'])
    new_lat.append(float(match.group()))
    new_long.append(float('-'+match2.group()))
```

```
all_data['Longitude'] = new_long
all_data['Latitude'] = new_lat
long = all_data.groupby('ID')['Longitude'].apply(list)
lat = all_data.groupby('ID')['Latitude'].apply(list)
print("Are lat and long the same length? " + str(len(lat)==len(long)))
print(lat.head())
print(long.head())

import matplotlib
import cartopy.crs as ccrs
import cartopy
fig, ax = plt.subplots()
fig.set_size_inches(20, 15)
ax = plt.axes(projection = ccrs.PlateCarree())
ax.set_extent([-120,0,90,0], ccrs.PlateCarree())
ax.gridlines(xlocs=range(-120,5,5), ylocs=range(0,100,5),draw_labels=True)
ax.coastlines()

for index in range(len(lat)):
    if (lat.index[index] == 'AL122005'):
        ax.plot(long[index],lat[index],'b-',linewidth=2)
    elif (lat.index[index] == 'AL182012'):
        ax.plot(long[index],lat[index],'g-',linewidth=2)
    else:
        ax.plot(long[index],lat[index],'r-',linewidth=0.2)
plt.show()

fig, ax = plt.subplots()
fig.set_size_inches(20, 15)
ax = plt.axes(projection = ccrs.PlateCarree())
ax.set_extent([-120,0,90,0], ccrs.PlateCarree())
```

```
ax.gridlines(xlocs=range(-120,5,5), ylocs=range(0,100,5),draw_labels=True)
ax.coastlines()
start_mid_end = pd.DataFrame(columns=['id', 'start_long','start_lat',
'mid_long','mid_lat','death_long','death_lat'])

for index in range(len(lat)):
    l = int(len(long[index])/2)
    ax.plot(long[index][l],lat[index][l],'ro',markersize=3)
    ax.plot(long[index][-1],lat[index][-1],'bo',markersize=3)
    ax.plot(long[index][0],lat[index][0],'go',markersize=3)
    start_mid_end = start_mid_end.append({'id':long.index[index],
'start_long':long[index][0],'start_lat':lat[index][0],
'mid_long':long[index][l],'mid_lat':lat[index][l],'death_long':long[index][-
1],'death_lat':lat[index][-1]},ignore_index=True)
plt.show()
fig, ax = plt.subplots()

fig.set_size_inches(20, 15)
ax = plt.axes(projection = ccrs.PlateCarree())
ax.set_extent([-120,0,90,0], ccrs.PlateCarree())
ax.gridlines(xlocs=range(-120,5,5), ylocs=range(0,100,5),draw_labels=True)
ax.coastlines()
slope = []

for index in range(len(lat)):
    ax.plot([long[index][0],long[index][-1]],[lat[index][0],lat[index][-1]],'r-',linewidth=0.2)
    if (long[index][-1]-long[index][0] == 0):
        slope.append(None)
    else:
        slope.append((lat[index][-1] - lat[index][0])/(long[index][-1] - long[index][0]))
all_slopes = pd.Series(data=slope,index=lat.index)
```

```
plt.show()

from geopy.distance import geodesic
dist = []
for index in range(len(lat)):
    d = 0
    for pos in range(len(lat[index])-1):
        p1 = (lat[index][pos], long[index][pos])
        p2 = (lat[index][pos+1], long[index][pos])
        d = d + geodesic(p1,p2).miles
    dist.append(d)
all_dist = pd.Series(data=dist,index=lat.index)
len(all_dist) == len(all_slopes)
----

slope_dist = pd.concat([all_dist, all_slopes], axis=1)
slope_dist.columns =['distance traveled','slope']
slope_dist.sample(5)
plt.scatter(slope_dist['distance traveled'], slope_dist['slope'])
plt.ylim(-25,25)
start_mid_end
plt.scatter(start_mid_end['mid_long'], start_mid_end['mid_lat'])
plt.show()
plt.scatter(start_mid_end['start_long'], start_mid_end['start_lat'])
plt.show()
plt.scatter(start_mid_end['death_long'], start_mid_end['death_lat'])
plt.show()

from sklearn.cluster import AgglomerativeClustering
from collections import defaultdict
def doAgglo(X, num_clusters):
```

```
model = AgglomerativeClustering(n_clusters=num_clusters, affinity = 'euclidean', linkage
= 'ward')
    labels = model.fit_predict(X)
    return (labels)

def get_plot(X1,X2,lab,t):
    fig = plt.figure()
    ax = fig.add_subplot(111)
    scatter = ax.scatter(X1,X2,c=lab, s=50)
    plt.xlim(-120,0)
    plt.ylim(0,90)
    plt.title(str(t) + ' Clusters')

def c_mean_2(vals,labels,n_clusters):
    sorter = defaultdict(list)
    for index in range(len(labels)):
        sorter[labels[index]].append(vals[index])
    some = 0
    for cluster_num in range(n_clusters):
        arr = np.asarray(sorter[cluster_num])
        some = some + np.sum(np.square(np.asarray(arr) - np.mean(arr)))
    return some/len(vals)

err = []

for i in range(1,7):
    temp = 0
    lab = doAgglo(start_mid_end[['mid_long','mid_lat']],i)
    get_plot(start_mid_end['start_long'],start_mid_end['start_lat'],lab,i)
    get_plot(start_mid_end['mid_long'],start_mid_end['mid_lat'],lab,i)
    get_plot(start_mid_end['death_long'],start_mid_end['death_lat'],lab,i)
```

```
    temp = temp + c_mean_2(start_mid_end['mid_long'],lab,i)
    temp = temp + c_mean_2(start_mid_end['mid_lat'],lab,i)
    temp = temp + c_mean_2(start_mid_end['start_long'],lab,i)
    temp = temp + c_mean_2(start_mid_end['start_lat'],lab,i)
    temp = temp + c_mean_2(start_mid_end['death_long'],lab,i)
    temp = temp + c_mean_2(start_mid_end['death_lat'],lab,i)
    err.append(temp)

fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(range(1,7),err)
lab = doAgglo(start_mid_end[['mid_long','mid_lat']],3)
df0 = pd.DataFrame()
df1 = pd.DataFrame()
df2 = pd.DataFrame()
dfALL = pd.DataFrame()

for index in range(len(start_mid_end)):
    if (lab[index] == 0):
        df0 = df0.append({'id':start_mid_end['id'][index],'lats':lat[start_mid_end
['id'][index]], 'longs':long[start_mid_end['id'][index]]},ignore_index=True)
        dfALL =
dfALL.append({'id':start_mid_end['id'][index],'lats':lat[start_mid_end['id'][index]],
'longs':long[start_mid_end['id'][index]]},ignore_index=True)
    elif (lab[index] == 1):
        df1 = df1.append({'id':start_mid_end['id'][index],'lats':lat[start_mid_end['id'][index]],
'longs':long[start_mid_end['id'][index]]},ignore_index=True)
        dfALL =
dfALL.append({'id':start_mid_end['id'][index],'lats':lat[start_mid_end['id'][index]],
'longs':long[start_mid_end['id'][index]]},ignore_index=True)
    elif (lab[index] == 2):
```

```
        df2 = df2.append({'id':start_mid_end['id'][index],'lats':lat[start_mid_end['id'][index]],
'longs':long[start_mid_end['id'][index]]},ignore_index=True)
        dfALL =
dfALL.append({'id':start_mid_end['id'][index],'lats':lat[start_mid_end['id'][index]],
'longs':long[start_mid_end['id'][index]]},ignore_index=True)

print(len(dfALL))

len(df1) + len(df0) + len(df2)
df0.head()
df1.head()
df2.head()

fig, ax = plt.subplots()
fig.set_size_inches(20, 15)
ax = plt.axes(projection = ccrs.PlateCarree())
ax.set_extent([-120,0,90,0], ccrs.PlateCarree())
ax.gridlines(xlocs=range(-120,5,5), ylocs=range(0,100,5),draw_labels=True)
ax.coastlines()

for index in range(len(df0)):
    ax.plot(df0['longs'][index],df0['lats'][index],'r-',linewidth=0.2)

for index in range(len(df1)):
    ax.plot(df1['longs'][index],df1['lats'][index],'b-',linewidth=0.2)

for index in range(len(df2)):
    ax.plot(df2['longs'][index],df2['lats'][index],'g-',linewidth=0.2)

plt.show()
```

```
def f_transf(lati,longi):
    if (lati < 90) and (lati >= 0) and (longi < 0) and (longi > -120):
        return (int(lati) * 121) + int((-longi))
    else:
        return None

def index_to_coord(index):
    return((index/121),(-(index%121)))

def make_markhov_values(data, mat_in, f_tranf):
    for coord_list in data.iterrows():
        l = len(coord_list[1]['lats'])
        for index in range(l-1):
            pos = f_tranf(coord_list[1]['lats'][index],coord_list[1]['longs'][index])
            pos_pr = f_tranf(coord_list[1]['lats'][index + 1],coord_list[1]['longs'][index + 1])
            if (pos == None) or (pos_pr == None):
                break
            mat_in[pos][pos_pr] = mat_in[pos][pos_pr] + 1
        pos1 = f_tranf(coord_list[1]['lats'][-1],coord_list[1]['longs'][-1])
        if (pos1 != None):
            mat_in[pos1][-1] = mat_in[pos1][-1] + 1
    return mat_in

def fix_probability(mat_in):
    for i in range(len(mat_in)):
        if sum(mat_in[i]) == 0:
            mat_in[i][-1] = 1
        l = mat_in[i]/sum(mat_in[i])
        mat_in[i] = l
    return mat_in
```

```
def generate(matrix, start_lat, start_long,f_to, f_back):
    path = []
    dead_hur = False
    curr_index = f_to(start_lat,start_long)
    path.append(f_back(curr_index))
    while not dead_hur:
        arr = (np.random.choice(range(0,len(matrix)), 1, p=matrix[curr_index]))
        curr_index = arr[0]
        if (curr_index != len(matrix) - 1):
            path.append(f_back(curr_index))
        else:
            dead_hur = True
    return path
size = 121 * 90
mat0 = np.zeros(shape=(size,size))
mat0 = make_markhov_values(df0,mat0,f_transf)
mat0 = fix_probability(mat0)
mat1 = np.zeros(shape=(size,size))
mat1 = make_markhov_values(df1,mat1,f_transf)
mat1 = fix_probability(mat1)
mat2 = np.zeros(shape=(size,size))
mat2 = make_markhov_values(df2,mat2,f_transf)
mat2 = fix_probability(mat2)
matALL = np.zeros(shape=(size,size))
matALL = make_markhov_values(dfALL,matALL,f_transf)
matALL = fix_probability(matALL)

import reverse_geocoder as rg
def make_predict(actual_longs,actual_lats, num, matrix):
    fig, ax = plt.subplots()
    fig.set_size_inches(20, 15)
```

```
    ax = plt.axes(projection = ccrs.PlateCarree())
    ax.set_extent([-120,0,90,0], ccrs.PlateCarree())
    ax.gridlines(xlocs=range(-120,5,5), ylocs=range(0,100,5),draw_labels=True)
    ax.coastlines()

    start_lat = actual_lats[0]
    start_long = actual_longs[0]
    for i in range(num):
        arr = list(map(list,

zip(*generate(matrix,start_lat,start_long,f_transf,index_to_coord))))
        ax.plot(arr[1],arr[0])
        print(i, len(arr[1]), len(arr[0]))

        for j in range(len(arr[1])):
            x = arr[1][j]
            y = arr[0][j]
            coordinates = (x,y)
            loc = rg.search(coordinates)
            print(loc)
    ax.plot(actual_longs,actual_lats,'bo',markersize=3)
    plt.show()

def predict_path(genesis):
    batch_size = 1;
    atl1_id = None;

    try:
        hurricane = df0.loc[df0['id'] == atl_id]
        make_predict(list(hurricane['longs'])[0],list(hurricane['lats'])[0],batch_size,mat0)
```

```
except:
    try:
        hurricane = df1.loc[df1['id'] == atl_id]
        make_predict(list(hurricane['longs'])[0],list(hurricane['lats'])[0],batch_size,mat1)
    except:
        try:
            hurricane = df2.loc[df2['id'] == atl_id]
            make_predict(list(hurricane['longs'])[0],list(hurricane['lats'])[0],batch_size,mat2)
        except:  make_predict(genesis[0],genesis[1],batch_size,matALL)
```

7. TESTING

Output for Prediction of Genesis to Decay route of Wind Storm

Input:

```
In [43]: predict_path(([-40], [15]))
```

Figure 7.1: Input

Output:

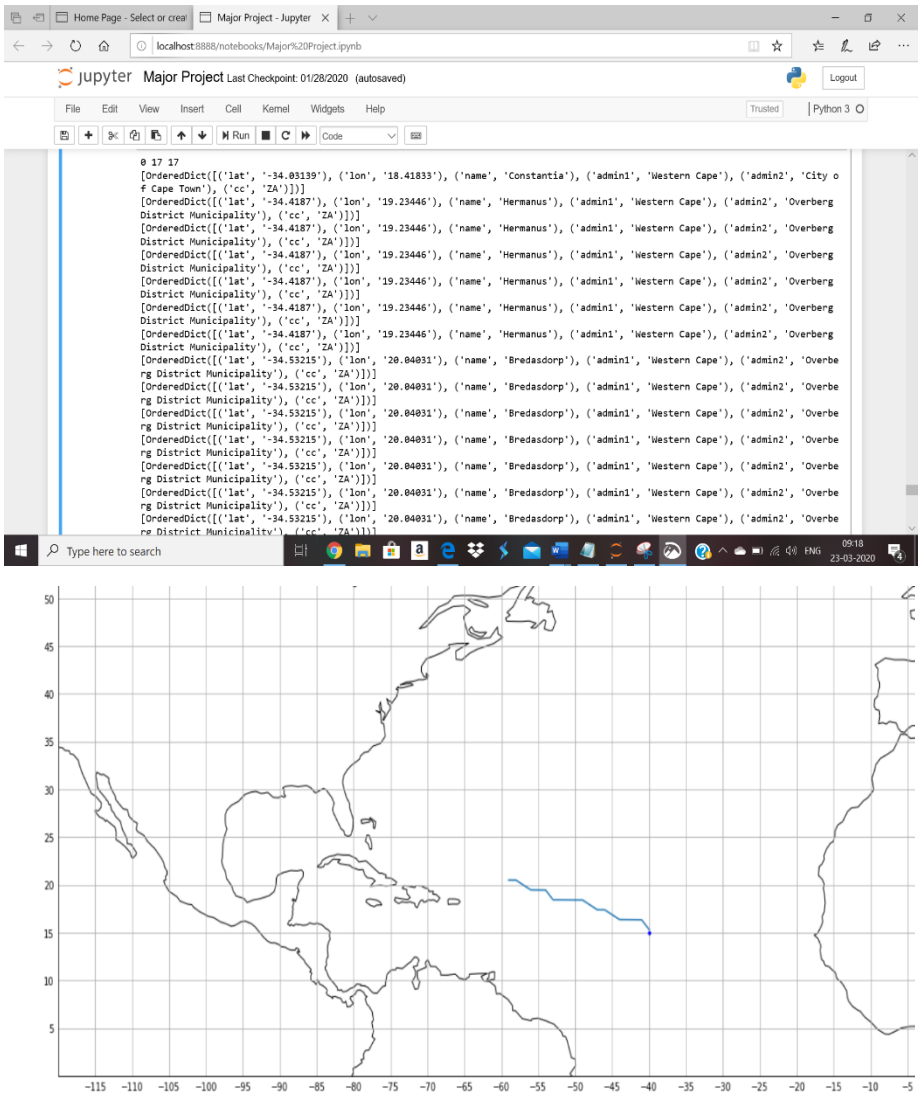

Figure 7.2: Output

8. CONCLUSION AND FUTURE SCOPE

8.1 CONCLUSION

Prediction of genesis to decay route of wind storm is a model which helps in predicting the path or route of a wind storm. Firstly, the Data preprocessing takes place where it is a data mining technique that involves transforming raw data into an understandable format. Next Data Analysis process takes place. In this step Data from various sources is gathered, reviewed, and then analyzed to form some sort of finding or conclusion. With the help of agglomerative clustering techniques, we will cluster the hurricanes into four groups. Finally, we use the Markov chain process. In this using the simulation made by the Markov model for the path and intensity it is possible to be able to track a real storm and determine the threat that it poses to a particular geographical location.

8.2 FUTURE SCOPE

The Future Scope of this model would be better if this model is integrated to a web page as it can be easily accessed by every individual. Individual can have a clearer picture on what is a wind storm. In this model the path is predicted from genesis point to decay point and also gives all the locations that will be damaged so including all this if the model can tell the intensity of the wind storm then it would be much beneficial to take certain actions.

9. REFERENCES

1. Taylor S. Cox, Calvin S.H. Hoi, Carson K. Leung*, and Caden R. Marofke "An Accurate Model for Hurricane Trajectory Prediction" IEEE Paper 2019, Department of Computer Science, Winnipeg, MB, Canada.

2. For gathering information about wind storms - *https://www.almanac.com/content/hurricane-forecast-facts-and-common-questions*

3. For understanding the concept of synoptic times and to get a clearer idea of why is that an important measure in measuring the time of occurrence of a particular wind storm- https://nsidc.org/cryosphere/glossary/term/synoptic-hour

4. For using Agglomerative Clustering for the model for prediction of the route - https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html

5. Using Keras, Preprocessing data, Model Creation – Machine learning Mastery - machinelearningmastery.com