DATA DRIVEN APPROACHES FOR WATER LEVEL PREDICTION

A Thesis Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Anusha Nemilidinne

August 2019

DATA DRIVEN APPROACHES FOR WATER LEVEL PREDICTION

Name

APPROVED:

Dr. Christoph F. Eick, Chairman
Dept. of Computer Science

Dr. Aron Laszka
Dept. of Computer Science

Dr. Xiaojing Yuan
Dept. of Engineering Technology

Dean, College of Natural Sciences and Mathematics

Acknowledgements

I would like to express my gratitude to my thesis advisor Dr. Christoph F. Eick for the useful comments, remarks and engagement throughout the learning process of this master thesis. I would also like to thank the incredible team at the UH Data Analysis and Intelligent Systems Laboratory for their continual insights and motivation for this thesis. I am grateful to God and thankful to my parents, brother and dear friends for their continued support in my life.

DATA DRIVEN APPROACHES FOR WATER LEVEL PREDICTION

An Abstract of a Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Anusha Nemilidinne

August 2019

Abstract

Floods are a hazardous disaster which threaten the United States and its territories as they affect millions of people every year. In this context, having suitable approaches for forecasting flooding will benefit people by reducing property damage and saving lives, by warning citizens of dangerous flooding events in advance. Consequently, many cities have developed sensor-based early warning systems which report water levels in real-time. The goal of this thesis is to take advantage of the data collected by such systems to develop data-driven water level prediction techniques.

In past research, physics-based hydrology models have been developed, such as the National Water Model, which predict water levels by simulating the rise and fall of water. The purpose of this research is to generate alternative, complementary, data-driven water-level forecasting models using existing statistical models and recurrent neural networks which extrapolate the past into the future. We investigate various time series forecasting approaches, in particular: Vector Autoregressive (VAR) and Long Short-Term Memory (LSTM) Networks. The investigated forecasting techniques are applied and evaluated using USGS datasets. Moreover, we analyze the role of soil moisture — a less explored parameter — in flood incidence and conduct some experiments that explore the relationship between rainfall and soil moisture. Finally, we develop a web application called FloodNet, a real time water level prediction system, with a forecast horizon of 2 hours for a location along Buffalo Bayou in Houston, Texas.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Historically, floods have caused tremendous loss of life, environmental harm, and economic costs. Floods are responsible for one-third of all deaths, damages and injuries from natural disasters [44]. When floods occur, communication links and infrastructure are often destroyed, including telephone communications, radio signals, roads, bridges, power plants, and public properties. Moreover, economic activities, such as agriculture and trade, come to a standstill. Repairs following flooding are costly. In addition to the effects already mentioned, floods can spread waterborne diseases [12], such as cholera, because pathogenic microorganisms are most commonly transmitted in contaminated floodwater. Finally, floods may even cause psychosocial effects on humans, including post-traumatic stress disorder (PTSD), depression, and anxiety.

According to the US News [32], many climate scientists predict that if the sea levels rise a mere one foot in the next century, nearly 3.9 million Americans would

be at risk of losing their homes; as examples, Georgia has approximately 28,000 people living in 127 square miles of low-lying land at risk of being flooded, Massachusetts has a dense area with 52,400 people at risk in only 32 square miles of land, and in South Carolina, more than 60,000 residents live in dangerous low-lying areas.

A lot of effort has been put into developing systems that help alleviate the damage from floods through early disaster predictions. Since floods are devastating and cause enormous damage, assessing the risk of flooding is very important. Understanding flood risk will help populations take necessary steps to prepare, enabling millions of lives to be saved and therefore significantly reducing losses. To better cope with flooding, some cities in the United States have developed sensor-based flood warning systems to alert residents of impending floods. For example, in Texas, Houston's Harris County Flood Warning System (HCFWS) and Austin's Flood Early Warning System both provide web-based systems that enable users to observe rainfall data and water level data collected by local sensors.

Ideally, individuals need to be able to access real-time data about rainfall and water level; if individuals know when and where floods will occur in advance, then they can take precautions to protect their properties. For example, individuals can prepare to evacuate quickly, plan routes to a local emergency shelter, and acquire emergency supplies early. Individuals can also proactively create a restoration plan to quickly implement once the flood is over. Therefore, if people receive an accurate flood alert in advance, they can take these actions and lower their losses.

Flood data analysis is a practical prediction discipline that employs approaches to examine datasets with features such as water level, rainfall data, soil moisture, and water speed to better predict floods in the future. In fact, many researchers have researched ways to predict floods. Presently, the National Weather Service (NWS) River Forecast Centers (RFCs) provide water level forecasting and information for 13 river basins across the United States [14]. However, at this time, RFCs cannot predict or forecast water levels for other basins; therefore, more generic methods of flood prediction that are more straightforwardly applicable, particularly for underserved regions, are needed.

This research aims to utilize the sensor-based data mentioned before and develop data-driven approaches for water level prediction. These approaches should extrapolate the past to forecast the future and will be helpful to provide warnings in advance to people about potential impending damage and dangers of flooding. In this thesis, we will focus on developing data-driven approaches to predict water levels by using existing statistical and deep learning models. We train models on historical data and then use the obtained models to predict water levels and flooding. Our research uses multivariate time series forecasting techniques. Specifically, we use the Long Short Term Memory (LSTM) Network, a recurrent neural network architecture, that is capable of remembering information for long periods of time. We transform the input data using the sliding window approach to achieve one-step ahead and multi-step ahead forecasting. We use LSTM autoencoders to achieve many to many type or seq2seq (sequence to sequence) type

prediction scenarios, i.e., multi-step ahead forecasting. We also develop a real-time web application called FloodNet that predicts water levels up to two hours into the future for a location downstream from Buffalo Bayou at West Belt Drive.

This thesis is organized as follows: In Chapter 2, we will discuss the background and related work. Moreover, we will describe some existing flood early-warning systems and recent research on flood-related problems. In Chapter 3, we explain soil moisture and its role in flood incidence. In Chapter 4, we will discuss the algorithms and methodology of our research. In Chapter 5, we will explain the data collection, summary statistics of datasets, autocorrelation analysis and stationarity tests. In Chapter 6, we discuss our experimental setup for different forecasting models and present the experimental results. Finally, Chapter 7 describes FloodNet, real-time water level forecasting system and Chapter 8 includes a conclusion and discusses future work.

Chapter 2

Background and Related Work

In this chapter, we define and discuss flooding. Then, we present surveys conducted on existing flood warning systems. After that, we explore existing flood prediction approaches. Finally, we investigate related work conducted by other researchers and scientists in the field.

2.1 Background

Flooding is a natural disaster that can happen almost anywhere in the world. It causes loss of life, property damage, and major disruption to populations daily routines. It can disrupt the water and electricity supply, block roads, and prevent people from doing their daily tasks or jobs.

According to the World Meteorological Organization (WMO), flooding is one

of the most hazardous disasters, affecting millions of people globally every year. Between 1970 and 2012, 44% of worldwide natural disasters were floods, making flooding the most common cause of other linked disasters, such as mass movement (subsidence, rockfalls, avalanches, and landslides), storms, droughts, extreme temperatures, and wildfires. Flooding is responsible for 14% of the fatalities and 33% of the economic damage resulting from natural disasters. This is equal to 272,251 fatalities and 788.9 billion U. S. dollars of economic damage [14]. In the United States alone, flooding triggers approximately 75% of all presidential disaster declarations. According to the National Weather Service, flooding is a coast-to-coast threat to the United States and its territories in all months of the year [13].

The FLOODSite [11] project categorizes flooding into five types:

2.1.1 Flash Flooding

Flash flooding occurs when there is heavy rainfall in a short period of time, generally less than 6 hours, on saturated soil or dry soil that has poor absorption ability. In areas with steep slopes, rainwater travels downhill at a high speed, overflowing low-lying areas and their surroundings in a very short time. Failure of a water-control structure, like a levee or a dam, can also cause a flash flood [41].

Flash floods are dangerous because of their high-speed water movement, but people tend to underestimate the danger of flash floods, despite their potential to cause widespread damage [11]. According to a report published by the NWS,

in 2014, flash flooding caused 31 fatalities and 16 injuries throughout the United States and was the leading cause of property and crop damage, at a cost of US $2.5 billion [34]. Further, in 2015, flash flooding caused 129 fatalities, 42 injuries, and was the number one cause of damage in the United States, at a cost of $2.1 billion [35]. Based on both reports, the cost of damage due to flash flooding surpassed the cost incurred by other natural disasters, such as hail, tornadoes, hurricanes, and drought. In our research, we are interested in forecasting flash floods.

2.1.2 Coastal Flooding

Coastal floods happen when sea water covers dry land. This type of flooding is triggered by storm surges, large waves, or a rise in sea level due to climate change or tsunamis [30]. As an example, Hurricane Katrina in 2005 was the costliest natural disaster and one of the five deadliest hurricanes in the history of the United States [24].

2.1.3 River Flooding

River flooding can be caused when rainfall happens during an extended period, leading major rivers to overflow. In addition to prolonged rainfall, dam failure, rapid snow melt, and ice jams can also cause river flooding.

2.1.4 Urban Flooding

This type of flooding occurs in an urban area that has poor drainage. In urban areas, people often develop sewage and drainage systems to collect rain water because of the limited surface area available for rain water infiltration. During and after high-intensity rainfall, these systems and canals may not be able to collect, store, and drain the rainwater, causing urban flooding [31].

2.1.5 Ponding/Pluvial Flooding

A ponding flood occurs when rainwater flows into a flat area that has no outlet or is not suitable for drainage. In this type of flooding, rain is the main source of the flood (i.e., the flood is not caused by water overflowing from rivers; it is due to rainwater that is on its way to a reservoir or river). Hence, it is called a pluvial flood. If this rain water is not absorbed by the ground, puddles or ponds develop. This type of flooding is slow or gradual, meaning people have more time to leave the area. Pluvial flooding is similar to urban flooding, but it occurs in rural areas [10].

2.2 Alleviating Flood Problems with Computational Methods and Information Technology

Computational methods and information technology have been utilized to support flood management and prevention in various ways. They serve as tools for flood forecasting, flood warnings, flood preparedness, and response management. According to the American Meteorological Society, flood forecasting is defined as the use of real time precipitation and stream flow data in rainfall-runoff and stream flow routing models to forecast flow rates and water levels for periods ranging from a few hours to days ahead [37].

These models utilize complex physics and mathematical equations that represent the dynamics of the atmosphere and of water flow. Weather conditions are simulated with past and/or current data in huge quantities and in a short time frame. Using these flood forecasting products, agencies issue warnings or alerts to the public. When a flood warning is issued, people can prepare, and emergency personnel can become involved in response or recovery operations like evacuation and search and rescue missions. Street flooding poses a challenge for traffic navigation. In such cases of street flooding, delays and blockages can be a matter of life and death. We will give a general overview of available computational models/tools used for each aspect of flood management. We will focus more on the technological aspects and the use of models/tools and less on the theoretical challenges of creating the models.

2.2.1 Flood Forecasting

In the United States, the National Oceanic and Atmospheric Administration (NOAA) is a federal agency within the Department of Commerce that focuses on studying the atmosphere and ocean conditions. Some of its services are: daily weather forecasts, severe storm warnings, climate monitoring, fisheries management, coastal restoration, and supporting marine commerce [29]. We will discuss the computational models used by the NOAA and its sub-agencies to produce forecasts. Additionally , we discuss widely used forecast models developed by various academic institutions.

One approach to hydrological forecasting or flood forecasting is modelling the statistical relationship between the hydrologic input and output, without explicitly considering the relationships that exist among the involved physical processes. Examples of stochastic models, characterized by randomness, used in hydrology are autoregressive moving average models (ARMA) [40] and Scalar Vector Machines (SVM) [3].However, the relationship between rainfall and runoff is notoriously nonlinear during storm events. Hence, conventional regression techniques cannot be used to model such a relationship. Much research in the field of flood forecasting has been conducted using data-driven approaches like artificial neural networks, linear regression, and the fuzzy rule-based expert system [38]. Our research uses recurrent neural networks, which exhibit temporal dynamic behavior and are designed to work with sequence prediction problems. Since flood forecasting belongs to a time series forecasting problem, we choose to take advantage of the capabilities of LSTMs to capture the temporal dependencies in the

data.

2.2.2 Flood Warning/Monitoring Systems

Flood warning systems rely on flood monitoring systems. Agencies use flood monitoring systems to issue warnings or alerts to the public during emergencies. Flood monitoring systems observe real-time rainfall amount, precipitation, stream flow, and water level collected by sensors placed in strategic areas. The sensors transmit valuable data during heavy rainfall, storms, and/or hurricanes. Some of them also measure wind speed and direction, barometric pressure, air temperature, road temperature, and humidity. We will discuss two widely used flood warning systems which serve inland Texas areas: the Harris County Flood Warning System, and the Rice University and the Texas Medical Center Flood Alert System. Additionally, we will discuss four flood warning systems utilized by other states: the Osceola County Flood Warning System, the Iowa Flood Information System, the NOAA, and the United States Geological Survey.

2.2.2.1 Houston Harris County Flood Warning System

The Houston Harris County Flood Warning System (HCFWS), funded by the Harris County Flood Control District (HCFCD), reports on and informs people about conditions along Harris Countys streams on a real-time basis. HCFWS also provides a user-friendly, interactive website that displays visualizations of conditions along Harris County watersheds [15]. HCFWS observes current flood conditions,

rainfall, and water levels by sensors placed along watersheds, typically located on bridges over bayous, creeks, or other waterways. Historical and current observed data can be accessed from the systems website. Harris County residents are encouraged to check HCFWS before daily commutes, particularly during or after heavy rainfall, since HCFWS provides an interactive map that can assist users to make decisions before and during a flood situation.

2.2.2.2 Austin Flood Early Warning System

The city of Austin, Texas, utilizes the Flood Early Warning System (FEWS). It monitors rainfall, water levels, and low-water crossings in Austin 24 hours a day, 365 days a year. Similar to the HCFWS interactive map, the city of Austin offers residents the ATXfloods website [1], maintained by the City of Austin Flood Early Warning System team.

2.2.2.3 Osceola County Flood Warning System

The Osceola County EOC works with the National Weather Service, the State of Florida, and the National Hurricane Center to monitor flood and storm threats and advise the community accordingly. When a storm or flood threatens to affect the county, staff monitor the event, relying on information from EOC officials and the National Weather Service for detailed and site-specific information regarding storm conditions and flood threats. The National Weather Service issues

updates and warnings, and any evacuation recommendations come from the Office of Emergency Management. All over-the-air television stations servicing the Osceola County area receive notification from the EOC to broadcast in the event of a threat. Additionally, all satellite and cable television providers and all major AM and FM band radio stations available in Osceola County provide immediate notification upon a threat being declared by the EOC or the NWS.

2.2.2.4 The Rice University and Texas Medical Center Flood Alert System (FAS)

FAS is an integrated system that utilizes radar, rain gage information, bayou gage data, and hydrologic modeling to issue flood warnings for the Rice University/TMC complex. It was designed by Dr. Philip Bedient at Rice University. Currently, version 3 of FAS is active. The key features of FAS3 include: it provides warnings and forecasts for main watersheds, it is interactive and contains real-time data, and it has large back-up servers in Oklahoma. This system is different from the National Weather Service in that NWS provides warnings by county, but FAS3 provides flood warnings for specific areas by monitoring key watersheds. On average, it makes maximum flood condition predictions with 2 to 3 hours of lead time. The official flood alert system is available at [39].

2.2.2.5 Iowa Flood Information System (IFIS)

IFIS is a web platform that allows access to community-based flood conditions, forecasts, visualizations, inundation maps, and flood-related information [19]. It

was developed by the Iowa Flood Center (IFC) at the University of Iowa. IFIS provides a five-day flood risk estimate for more than 1000 communities in Iowa using its rainfall-runoff forecast. It helps people and agencies to make better-informed decisions on flood occurrence, and it alerts communities in advance to help minimize damage from floods. It provides a REST API for users to request time series data for a specific sensor and a particular time period. A sample web service call looks like http://waterservices.usgs.gov/nwis/dv/?format=waterml,2.0sites=01646500siteStatu
It relies on a wide range of installed sensors, like stream sensors, which report depth and elevation, and rain/soil moisture sensors.

2.2.2.6 National Oceanic and Atmospheric Administration (NOAA)

NOAA is a United States scientific agency that monitors conditions of the oceans and the atmosphere. Many cities do not have their own flood early warning system, but NOAA aids cities in monitoring the water status and velocities of their streams. Moreover, NOAA's Office of Water Prediction (OWP) developed the National Water Model (NWM), which is a hydrologic model that simulates observed flood status and forecasts stream flow. NWM forecasts streamflow for 2.7 million rivers and contains other hydrologic information, helping forecasters predict when and where floods might be expected. It was designed to provide more closely integrated water prediction capabilities and therefore to promote resilience to water risks. The NWM prediction model simulates the water cycle with mathematical equations of the different processes. Additionally, it contains a representation of physical processes, such as snow melt and infiltration and water

movement through the soil layers, which vary significantly with changing elevations, soils, vegetation types, and a host of other variables.

2.2.2.7 United States Geological Survey (USGS)

The USGS is a United States scientific agency. It provides real-time or near real-time data and information on current conditions. The scientists at USGS develop new methods and tools to supply timely, relevant, and useful information to users and agencies that use the data to forecast floods. They provide tools to download time series data for lakes, streams, canals, and several other different sites in several states across the United States.

The USGS has measuring stations installed at specific locations across streams, allowing it to provide real-time time series data grouped by county. It provides data for a very wide range of parameters, including water level/flow, physical properties, water quality, gage performance, and meteorological parameters. We can download data manually or we can use their web service. The USGS time series data is updated each hour, helping agencies to provide near-real-time forecasts. The time series data can be downloaded at [47].

Chapter 3

Soil Moisture and its Role in Flood Incidence

Soil moisture is the amount of water stored in the soil and is affected by precipitation, temperature, and soil characteristics. Generally, it is the water held in the spaces between soil particles. There are different types of soil moisture [42].

- Surface soil moisture, which is water in the upper 10 cm of soil
- Root zone soil moisture, which is water in the upper 200 cm of soil and available to plants

Each soil type has a different soil moisture range. For every particular soil type, the water content in the soil has different field capacity and permanent wilting point [6]. It is measured as volumetric water content of soil, and its unit is

$m^3 water/m^3$ soil. Usually, it is described as a percentage. For example, the volumetric water content at field capacity and permanent wilting point are 15-20% and 5-10% respectively, for sandy soils [6].

3.1 How Do We Measure Soil Moisture?

Soil moisture measurements can be obtained from two different sources:

- in situ
- satellite sensors using remote sensing

In situ refers to any measurement that is obtained through direct contact. Therefore, in situ soil moisture measurements are taken using a sample of soil present in a particular area, thus capturing the local soil conditions [50]. Remotely sensed soil moisture data are retrieved from passive and active microwave sensors on board of various satellites [28]. We rarely use in situ measurements because of their scarce availability. There has been an increase in the use of satellite soil moisture observations for flood forecasting and hydrology models. In our research, we have explored several sources of soil moisture datasets. The Soil Moisture Active Passive (SMAP) datasets [9] — which were launched by NASA in 2015 — are either not collecting data currently and have data for only the past few years, or their spatial coverage is limited to certain parts of the world, excluding the United States. Hence, in our research, we used datasets from the International Soil Moisture Network (ISMN) [20] which provides soil moisture datasets at different soil

depths for several regions of the United States. It is a data hosting facility for global in situ soil moisture measurements [20]. Both SMAP and ISMN measure soil moisture using the same units: $m^3 m^{-3}$.

We explored several sources that provide soil moisture data. Some do not have recent data, and some are not available for the United States. For example, Weather API is available only after paying a fee, and in order to access the historical data, we have to choose a subscription plan. Another source, the National Snow and Ice Data Center, contains NASA SMAP soil moisture datasets, but they are only available for specific locations and do not contain recent data.

3.2 Why Is Measuring Soil Moisture Important?

Soil Moisture is an invaluable parameter in hydrological modelling since it influences several components of water cycle like evapotranspiration, infiltration and runoff. It also has been demonstrated that it plays an important role in hydrology predictions and there have been several studies that incorporated soil moisture observations for hydrology modeling [27]. Since soil moisture controls the exchange of heat and energy between land surface and atmosphere, it plays a vital role in the amount of precipitation and hence the amount of rainfall that runs off into near by streams and rivers. In our research we have investigated techniques to model the relationship between rainfall and soil moisture in two ways: soil moisture as a function of rainfall, and rainfall as a function of soil moisture. By studying the relationship between rainfall and soil moisture using historical data,

we can derive meaningful insights about the rainfall-runoff process for a particular watershed, thus helping us better predict floods.

3.3 The Role of Soil Moisture in Rainfall-Runoff

There are several factors that affect runoff. Along with rainfall intensity, rainfall amount, and duration of rainfall, soil moisture also plays a vital role in rainfall-runoff. The total soil water storage capacity is an important factor of rainfall-runoff. The total soil water storage capacity occurs when soil is filled with water [6]. This essentially means that the soil is saturated. When soil reaches its full capacity, it can no longer absorb water. When there is a huge amount of rainfall and the soil is saturated to its capacity, the excess rainfall runs off into nearby streams, increasing the water levels and the risk of flooding. This phenomenon in which excess rainfall runoff into streams is called surface run-off. Further, if the soil's rate of absorption is less than the rate of rainfall occurrence, the soil can't absorb all of it. Immediately after a rainfall event, the soil retains a certain amount of moisture, and this affects soil infiltration capacity.

In summary, several studies in recent decades have used satellite soil moisture observations into the rainfall-runoff model to improve their flood forecasting skills. Soil moisture can be used for several applications, of which flood forecasting is our prime interest. By exploiting the strong physical connection between soil moisture and rainfall, we can forecast floods for different catchment areas by modelling a data-driven approach using deep learning algorithms, incorporating

soil moisture along with several other features like precipitation, discharge, and gage height. However, to do this, detailed and accurate real time soil moisture data are needed, and as we discussed earlier, obtaining such data is a major challenge.

Chapter 4

Methodology and Algorithms

This chapter is devoted to describing the statistical and deep learning models, Vector Auto Regressive (VAR) and Long-short Term Memory (LSTM) Networks, respectively, different architectures of the LSTM model, strategies for multi-step ahead forecasting, performance metrics, and hyper parameter tuning. This chapter also discusses various tools, programming languages, and libraries used to implement the models.

4.1 Time Series Forecasting Approaches

In our research, we use VAR and LSTM models to predict water levels.

4.1.1 VAR (Vector Auto Regressive)

VAR is a stochastic process model used to capture linear inter-dependencies among multiple time series [26]. VAR generalizes the univariate auto regressive (AR) model by allowing more than one evolving variable. The VAR model consists of a set of k endogenous variables $y_t = (y_{1t}, y_{2t}, ..., y_{kt})$ over a time period $(t = 1, ..., T)$. A VAR model p lags is defined as:

$$y_t = c + A_1 y_{t-1} + A_2 y_{t-2} + ... + A_p y_{t-p} + u_t \tag{4.1}$$

where $y_t = (y_{1t}, y_{2t}, ..., y_{kt})$ is a $(k \times 1)$ vector of exogenous variables or intercepts, and A_i is a $(k \times k)$ coefficient matrix for $i = 1, ..., p$. Finally, $u_t = u_{1t}, u_{2t}, .., u_{kt}$ is a $(k \times 1)$ stochastic process or error term where u_t is identically distributed with mean $E(u_t) = 0$ and positive definite co-variance matrix $\sum_u = E(u_t u_t)$.

$$y_{1t} = c_1 + A_{11} y_{1t-1} + A_{12} y_{2t-1} + ... + A_{13} y_{3t-1} + u_1 \tag{4.2}$$

$$y_{2t} = c_2 + A_{21} y_{1t-1} + A_{22} y_{2t-1} + ... + A_{33} y_{3t-1} + u_2 \tag{4.3}$$

$$y_{3t} = c_3 + A_{31} y_{1t-1} + A_{32} y_{2t-1} + ... + A_{33} y_{3t-1} + u_3 \tag{4.4}$$

As we can see from Equations 4.2, 4.3, and 4.4, each variable depends on all other variables, meaning we can use VAR to successfully detect the interdependencies between time series. We use the VAR method from the package vars in R [49].

4.1.2 LSTM (Long Short-Term Memory)

LSTM neural networks are a particular type of recurrent neural network that have gained a lot of attention within the machine learning community [17]. Simply, LSTM networks [36] contain internal contextual state cells that act as long-term or short-term memory cells. The output of the LSTM network is controlled by the state of these cells. The input to the current time step is not only the actual input; it is also the cell state and hidden state of past time steps. As time passes, it is less likely that the next output depends on a very old input. This time dependency is the contextual information to be learned. LSTMs learn this contextual information by using a forget date, which helps them to learn when to remember and when to forget. Figure 4.1 shows an LSTM cell [36].

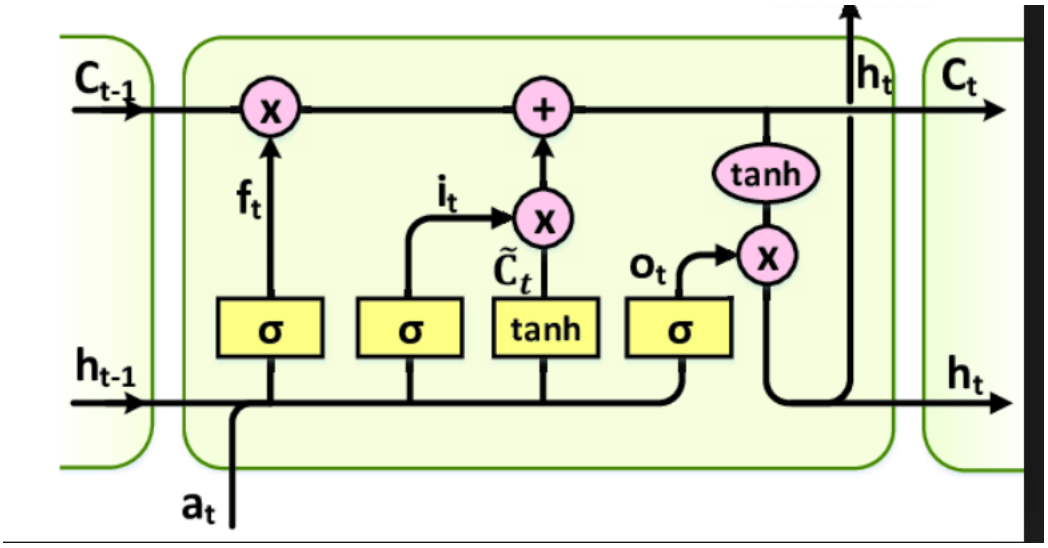


Figure 4.1: Repeating module in an LSTM

4.1.2.1 The Core Idea Behind LSTM

The key to LSTMs is the cell state. LSTMs can add information or remove from the cell by using gates. Gates represent a sigmoid neural net layer and a pointwise multiplication operation. We know that a sigmoid function outputs numbers between 0 and 1, which means it controls how much of each component should be let through. A value of 0 means discard everything, and a value of 1 means pass everything.

The first step in our LSTM is to decide what information needs to be thrown away from the cell state. A forget layer is used to accomplish this. It looks at h_{t-1} and x_t, and it outputs a number between 0 and 1 for each number in the cell state C_{t-1}. So, a 1 means keep the information completely, and a 0 means get rid of the information. Any number between 0 and 1 implies that part of the previous cell state is remembered. The next step is to decide what new information will need to be stored in the cell state. This is achieved using input gate i_t. The input gate layer determines which values to update and outputs new candidate values which are added to the current state. In the next step, the above-mentioned forget gate and input gate are applied to the previous cell state and the new candidate state to produce the new cell state. The next step is to decide the output of an LSTM cell.

LSTMs are quite popular for many reasons. LSTMs solve the vanishing gradient problem, which is commonly seen in recurrent neural networks. This problem is due to a long series of multiplication of small numbers. Vanishing gradient descent [16] hampers models from learning long sequences of data. LSTMs solve

the problem by using an identity activation function, which has a derivative of 1.0, so the gradient neither explodes nor vanishes [17]. LSTMs are very good at detecting temporal dependency between time series. Therefore, they are ideal for such problems. In our research, we use the sequential and functional API of the keras library in Python to learn the LSTM model.

4.2 One-step Ahead Forecasting

One-step ahead forecasting means predicting the target variable at the current time step based on past values of input variables. How many time steps we go back in time — look back window — is usually determined using autocorrelation analysis [2]. Autocorrelation analysis is the correlation of a time series with its own present and future values. This process is explained in detail in Chapter 5.

4.3 Multi-step Ahead Forecasting

Multi-step ahead forecasting means predicting the target variable several or multiple time steps into the future. This type of forecasting is also called many to many type prediction or sequence to sequence (seq2seq) prediction. The window that depicts how far into the future the model is able to predict is called the look ahead window. The look ahead window is selected based on the model's performance, as the length of the look ahead window varies and is usually determined by trial and error or by using autocorrelation analysis.

4.4 Architectures of LSTM

In this research, we use three different types of LSTM architectures, each of which is suitable to a particular prediction problem that is one-step ahead or multi-step ahead forecasting:

- Stacked LSTM
- Single-layer LSTM
- LSTM Autoencoder

4.4.1 Stacked LSTM and Single-layer LSTM

We use single-layer LSTM and stacked LSTM for one-step ahead forecasting, whereas the LSTM autoencoder is used for multi-step ahead forecasting. Stacked LSTM means that we use more than one LSTM hidden layer stacked on top of each other. We use a batch normalization layer between two hidden layers to deal with the covariate shift. The covariate shift [18] is defined as a change in the distribution of input domain, which is difficult for the algorithms to deal with. This problem generally arises when there are many hidden layers. In this scenario, one layer feeds its output to the second layer, which in turn feeds to the next layer, and so on. This causes an internal covariate shift. Hence, we normalize each batch using mean and variance to deal with this problem [18].

4.4.2 LSTM Autoencoder

As mentioned earlier, one effective approach to seq2seq prediction problems is Encoder-Decoder LSTM (Autoencoder). Because of this capability of encoder-decoder architecture, the Mean Squared Error (MSE) is observed to be lower than that of the stacked and single-layer LSTM models. In general, the encoder-decoder architecture is composed of two networks: one for reading the input sequence and encoding it into a fixed-length vector, and another for decoding the fixed-length vector and outputting the predicted sequence [5]. In our research, we are interested only in future prediction. The architecture of the LSTM autoencoder is shown in Figure 4.2.

The first step is to generate sub time series from the dataset, which is described in detail in Section 4.5. The data ($X^1, X^2 .....X^n$) where each sample is of shape 4×3 is then fed to an encoder, which is a layer of LSTM cells, as shown in the Figure 4.2. So each sample is of shape time steps $\times$ number of features. The number of time steps is 4 which is determined using autocorrelation analysis as discussed later in this chapter. The number of features is equal to the number of variables in the dataset. For the Buffalo Bayou dataset, the number of features is equal to 3 since the dataset contains rainfall, gage height and discharge data. For the Titusville dataset, the number of features is equal to 2 since it contains rainfall and soil moisture data. The encoder converts the input sample into a vector output, meaning it automatically extracts features into a vector. Then, the encoded feature vector is fed to a decoder using a repeat vector (not shown in the figure). RepeatVector repeats the hidden state output of the LSTM layer in the last

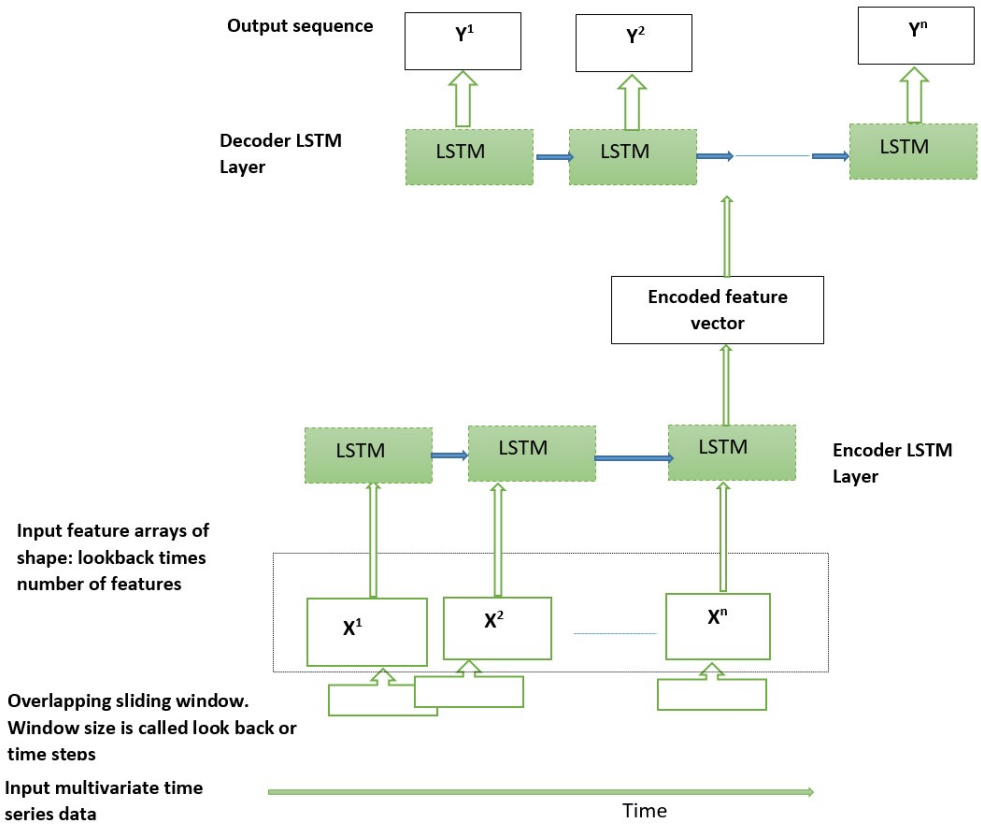


Figure 4.2: LSTM Autoencoder Architecture

time step. The repetition factor is equal to the look ahead window, which is the number of timesteps for which we predict water levels. The decoder consists of one LSTM layer and then a dense layer, which is applied to the output of the LSTM at each time step. The output of the decoder ($Y^1, Y^2 .....Y^n$) is future predictions.

4.4.3 Strategies for Multi-Step Time Series Forecasting

The one-step ahead forecast model can also be used to predict multiple time steps into the future. We use three strategies to achieve multi-step ahead forecasting:

- Iterative Strategy Using Actual Values (IMS_a)
- Iterative Strategy Using predictions (IMS_p)
- Multiple input multiple output (MIMO) models

Iterative Strategy Using Actual Values

The Iterative/Recursive Strategy using Actual Values involves prediction of water levels at multiple time steps in the future using actual/observed values of input features at past time steps. Initially, the water level at the current time step, w_t is predicted using the observed values of rainfall, water level, and gage height at the previous four time steps if the look back window is four. Then, the water level at the next time step, w_{t+1}, is predicted using the observed values of rainfall, water level, and gage height at the previous four time steps. This process is repeated until the model predicts for the whole look ahead window. That is, if we would like to predict 10 time steps into the future, then the look ahead window will be 10. Algorithm 1 shows the pseudo code for the above strategy.

Algorithm 1 IMS_a

```
predicted_waterlevel ← [] # Array to store predicted water levels for T time steps ahead
test_data[0:length(test_data)] # Test data where each sample is of shape look_back_window × n_features
i ← 0
while i < 10 do
    predicted_waterlevel[i] ← model.predict(test_data[i])
end while
```

Iterative Strategy Using Predictions

The Iterative/Recursive Strategy using Predictions predicts water levels at multiple time steps in the future using predicted values of input features. It is also called iterative forecasting because when calculating a K-step ahead forecast, we iteratively feed the forecasts of the model back in as input for the next prediction. We will get a low performance over longer time horizons K, because, due to the accumulation of error in the individual forecasts, this strategy suffers from high variance. Algorithm 2 shows the pseudo code for the above strategy.

Algorithm 2 IMS_p

```
T ← 4 # T is the look back window
sliding_window # Array containing a single sample of shape look_back_window × n_features
predicted_waterlevel ← [] # Array to store predicted water levels for K time steps ahead
i ← 1
while i <= k do
    predictions ← model.predict(sliding_window)
    predicted_waterlevel[i] ← retrieve waterLevel_forecast from predictions
    Shift sliding_window by removing first row and appending the predictions
end while
```

MIMO

We implement the MIMO strategy using LSTM auto encoders. In this strategy, the model takes multiple inputs as input, i.e., past values of all the variables at different time steps, and gives multiple outputs in parallel, i.e., predicted water level at different time steps into the future. Hence this strategy is called MIMO. As discussed earlier, autoencoders are good at multi-step time series forecasting. The models built using this strategy are capable of generating forecasts for multiple

time steps into the future in parallel as opposed to the iterative strategy where the models predict multiple time steps into the future iteratively using sliding window approach. Autoencoders have been shown to outperform the two strategies discussed above. The results of all models and strategies are discussed in Chapter 6.

4.5 Methodology

The input to LSTM is three-dimensional, which means each sample is of shape $timesteps \times features$. Hence, it is necessary for us to transform the data into a format that is suitable for LSTM. We achieve this using the sliding window approach as shown in Figure 4.3.

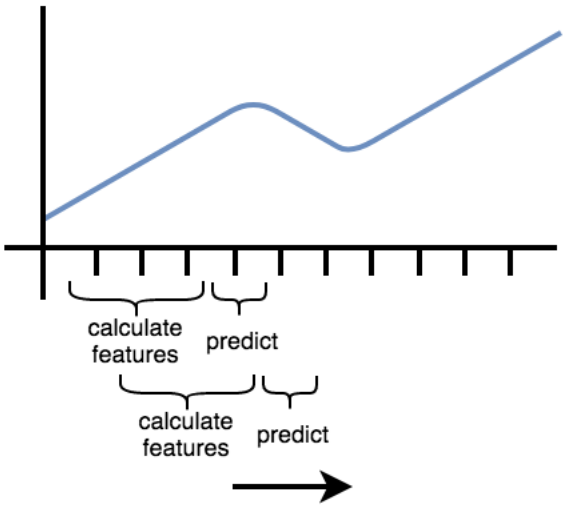


Figure 4.3: Sliding Window Approach

It is also called a rolling mechanism. This mechanism takes time series as input

and generates a sub time series. For example, each sub time series looks like:

$$\begin{bmatrix} 789cft/sec & 0.15inches & 39ft \\ 800cft/sec & 0.05inches & 41.5ft \\ 875cft/sec & 0.12inches & 43.6ft \\ 1200cft/sec & 0.01inches & 44.5ft \end{bmatrix}$$

The above matrix represents how each sample looks. In our case study, we take timesteps to be 4 and the number of features to be 3, since the dataset contains values for rainfall, discharge, and gage height. Each row indicates values of the three features at a particular time step. Therefore, the first row indicates values at time step t, the second row at t+1, the third row at t+2 and the fourth row at t+4. he first column is for discharge, the second column is for rainfall, and the third column is for gage height. We repeat this process for the Titusville dataset.

Scaling

Machine learning algorithms perform better if all input data features have consistent scale and distribution. We can either normalize or standardize the input data to achieve this consistency. Standardizing a dataset means rescaling the distribution of values so the mean of observed values is 0 and the standard deviation is 1. This is implemented using Standard scaler provided by the sklearn library. Normalization means rescaling the data from the original range so all values are in the range of 0 to 1. It is usually conducted considering the features minimum and maximum values, implying normalization is prone to outliers. Therefore, if the data has outliers, then normalization may not be the best way to rescale the

data. The sklearn library of Keras offers different types of scaling mechanisms.

Among the available scalers, Robust scaler is claimed to be most robust to outliers. Robust scaler removes the median and scales the data according to the interquartile range (IQR), which is the range between the first quartile (25th quantile) and the third quartile (75th quantile). In our research, there has been no significant improvement between the performance of the models using Robust scaler or Standard scaler. Therefore, we standardize the dataset using Standard scaler from the sklearn preprocessing library in Python.

Train-test Split

We split the whole dataset into a train and test dataset and fit the scaler only to the training data - not to the entire dataset - because we want to see how well the model is able to generalize the test set. We do not fit the scaler to the whole dataset because doing so will leak the information about the distribution of test set to the model which results in overfitting. In other words, the test set acts as new/unseen data. The test-train split ratio is our choice, and in our research, we choose 80% as the training set and the remaining 20% as the test set.

4.6 Evaluation Measures

In our research, we use different forecasting methods. In order to measure performance and validate the results of different forecasting methods, we use the following evaluation metrics:

- Root Mean Squared Error (RMSE)
- Mean Absolute Error (MAE)

MAE measures the difference between predicted values and actual values. The MAE formula is given by Equation 4.5, where y_i represents predictions, x_i represents actual values, and n is the number of observations. As the name suggests, MAE computes the average of the absolute error.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \tag{4.5}$$

Root Mean Squared Error (RMSE) is commonly used as a measurement for model predictions. RMSE is an accuracy evaluation method that is used to compare prediction errors of different models. It computes the square root of the average of squared errors. The RMSE formula is shown in Equation 4.6, where y_i represents the prediction outcomes and x_i represents the actual values.

$$RMSE = \frac{1}{N}\sqrt{\sum_{i=1}^{n}(y_i - x_i)^2} \tag{4.6}$$

4.7 Tools

4.7.1 Python and R

In our research, we use Python and R to design and implement the algorithms. Python is a powerful language for data science that comes with built-in libraries

for building machine learning models. Python has a rich set of libraries for almost any use case that we can come up with, from image manipulation and scientific calculations to server automation. Several Integrated Development Environments (IDEs) support development in Python. In our research, we use PyCharm 2018.1.4 Community Edition [22], which is an open source software that can be downloaded freely from the internet. R is a language and environment for statistical computing and graphics. We use R studio to develop VAR models since R is very effective in statistical methods [7].

4.7.2 Keras

Python contains an open source neural network library called Keras [23]. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or MXNet. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier. The code is hosted on GitHub, and community support forums include the GitHub issues page and a Slack channel. It provides both sequential and functional API for developing deep learning models.

4.7.3 Jupyter

In our research we use the ipython notebook called Jupyter which is present in the PyCharm IDE. It is very powerful for interactively developing and implementing data science projects [9]. It is a rich tool that integrates code, text, and output in a single document, and it supports the execution of each module of code separately, which saves time when dealing with modules that require lengthy execution time. Its intuitive workflow promotes iterative and rapid development, making notebooks an increasingly popular choice at the heart of contemporary data science, analysis, and, increasingly, science at large. Best of all, as part of the open source project Jupyter, notebooks are completely free.

4.7.4 Hyperparameter Tuning

A hyperparameter is a configuration of the neural network that cannot be estimated from the data. We cannot estimate the best value for a hyperparameter on a given problem. It is generally estimated using trial and error [28]. The model hyperparameters include: learning rate for training a neural network, batch size, number of training epochs, number of neurons, optimization algorithm, loss function, learning rate and momentum, network weight initialization, neuron activation functions, and drop out regularization.

We perform experiments with different configurations like varying the number of training epochs, keeping batch size and number of neurons constant. By visualizing the summary statistics (Root Mean Square Error and Mean Absolute

Error) of different configurations using box plots, we choose the best model. We run each experiment multiple times because of the stochastic nature of neural networks and then choose the best hyperparameters for the model. The amount of learning per batch (learning rate), the number of updates per epoch (batch size), and the number of epochs all are dependent on each other. We use K-Fold Cross Validation to evaluate the performance of deep learning models. It provides a very good estimate of the models performance on unseen data.

Chapter 5

Data Acquisition and Analysis

This chapter is devoted to the description of datasets and exploratory data analysis. First, we will describe the objectives of our case study, located in the state of Texas. We will then discuss retrieval of datasets, autocorrelation analysis, stationarity of datasets, and summarization of data using statistics and plots.

5.1 Motivation and Objectives

There are several variables that affect flood occurrence. Flood occurrence is a complex process impacted by various variables, such as rainfall amounts, ground cover, river/stream topography, capacity of the watercourse to drain water, and drainage basin conditions prior to rainfall events. Moreover, forecasting some of those variables is also challenging; for example, rainfall amounts are affected by

various variables that are not easy to predict, such as wind speed and wind direction. Our case study focuses on predicting water levels across Buffalo Bayou in Harris County in Houston, Texas. Harris County currently has a flood warning system called the Harris County Flood Warning System (HCFWS) that is fully dedicated to issuing live flood warnings and hourly weather forecasts based on the warnings received from the National Weather Services (NWS) Advanced Hydrologic Prediction Service (AHPS). AHPS employs physics-based simulation models. In our research, we aim to predict water levels and therefore flooding using data-driven approaches by utilizing past/historical data. The advantage of this approach is that we do not make any assumptions about the physical processes, thereby reducing the complexity involved. Our work aims to serve as a complementary or alternative approach to the existing physics-based hydrology models.

5.2 Data Collection and Preprocessing

Yue Cao's dissertation Design and Implementation of A DAG-Based Water Level Prediction Approach [4] used a dataset from the Harris County Flood Control District. There are many problems associated with using this dataset, like highly irregular sampling rates and missing data, and downloading the data is challenging. In order to overcome this problem, we looked for more complete, better-quality datasets that could be used for our research project. The best water-level dataset provider in the United States is the United States Geological Survey (USGS) website. The only problem encountered with this source is that some

states have precipitation data only for the past 120 days. In the case of Buffalo Bayou, part of the Buffalo Bayou watershed, the USGS dataset contains rainfall, gage height, and discharge variables for the past 120 days. Precipitation for this sample averages at 49 inches per year. The wettest month is June, when rainfall totals 5.93 inches, on average. This dataset is used to develop water level prediction models using different architectures of LSTM.

Apart from this dataset, we also use a secondary dataset provided by the International Soil Moisture Network (ISMN). This dataset consists of a time series of rainfall and soil moisture values for Titusville, Florida. This dataset is used to learn prediction models that try to predict soil moisture from rainfall and vice versa. It is also used to shed light on correlations between the two variables.

5.3 Data Acquisition

5.3.1 USGS Datasets

Through the USGS Water Data for the Nation website [47], we can access water resources data collected from approximately 1.9 million sites in all 50 states. The USGS investigates the occurrence, quantity, quality, distribution, and movement of surface and underground waters.

The USGS website provides four data categories: surface water, groundwater, water quality and water use. For our case study, we are interested in the surface water data. There are two data retrieval approaches: manual and automated retrieval.

5.3.1.1 Manual Retrieval

To access current conditions for surface water data, we first have to choose one of the site selection criteria options. Once we select the site, we are directed to a web page that lets users select the state, variables they are interested in, output formatting options, and time period. The data can then be downloaded at [47].

5.3.1.2 Automated Retrieval

Since the precipitation data is available only for the past 120 days, we formulated the idea of retrieving the data using automated retrieval. The USGS provides a REST API that takes in the begin date, end date, site number, output format and feature codes [45]. The USGS website updates the data every hour. We have written a Python utility which runs every hour, makes a request to the website using the API [46], downloads the data, and appends it to the previous dataset. This helps us to gather more data, since the more data input, the better the prediction accuracy of models will be. We use the requests package in Python to make a web service call.

5.3.1.3 Data Entities

The USGS dataset has 5 variables. They are: selected site ID, gage height, precipitation, discharge and timestamp.

- **Sensor** identification information.

- **Gage height**, which is also known as stage is the height of the water in the stream above a reference point [48].
- **Discharge** is also known as streamflow. It is the volume of water flowing past a given point in the stream in a given period of time [48]. Streamflow is measured in cubic feet per second (ft^3/s). Discharge is a better indicator of conditions along the river than gage height.
- **Precipitation** is water released from clouds. It can take many forms, like rain, freezing rain, sleet, snow, or hail. Most precipitation falls as rain.
- **Timestamp** of the sample. The time granularity is 15 minutes.

Table 5.1 shows a sample of the USGS dataset. From now on, we call this dataset the BuffaloBayou dataset.

Table 5.1: Buffalo Bayou Dataset

Date	**Discharge (cft/sec)**	**Gage height (feet)**	**Precipitation (inch)**
4/11/2019 13:00	201	36.8	0.0
4/11/2019 13:15	202	36.81	0.02
4/11/2019 13:30	196	36.82	0.02
4/11/2019 13:45	195	36.77	0
4/11/2019 14:00	196	36.76	0.02
4/11/2019 14:15	200	36.77	0.06
4/11/2019 14:30	203	36.8	0.03
4/11/2019 14:45	201	36.83	0.03

5.3.2 International Soil Moisture Network Dataset

In our research we use rainfall and soil moisture dataset from the International Soil Moisture Network (ISMN). Before downloading the data from this website, we need to create an account. This interface allows us to select a particular area on the map from which we want soil moisture data. For our project, we use soil moisture data for Titusville, Florida, because it has a relatively high number of rainfall events, which improves the reliability of the models predictions. We can directly search for the station name using the Search Station feature. After the selection is made, we click Download, and the interface provides us with a link to the zipped folder. The folder contains .stm files containing different variables. Each variable has a different file for each depth. Since we are interested only in rainfall and soil moisture, we choose the file that contains data for the top 0.05 m of the soil. .stm files can be opened using any text editor, and the fields are separated by spaces. The .stm file is converted into a .csv file using Python. A snapshot of the data, called the Titusville dataset, is presented in Table 5.2. The data can be downloaded by visiting [21]. The time granularity for this dataset is one hour.

Table 5.2: Titusville Dataset

time	precipitation (mm)	soilMoisture (%)
1/3/2018 10:00	2.4	0.22
1/3/2018 11:00	2.8	0.223
1/3/2018 12:00	2.9	0.286
1/3/2018 13:00	0.3	0.332
1/3/2018 14:00	2.3	0.337
1/3/2018 15:00	3.3	0.377
1/3/2018 16:00	1.3	0.38
1/3/2018 17:00	1.7	0.383
1/3/2018 18:00	0	0.384

5.4 Exploratory Data Analysis

Before performing any analysis, it is important to deal with missing values. The Titusville dataset contains missing values, so to deal with this problem, each missing value is replaced with the value at the previous timestep.

5.4.1 Visualization of Data Distribution for the Buffalo Bayou Dataset

In order to create water-level forecasting models, we use the past 120 days of daily rainfall amount, discharge, water-level, and soil moisture data from 01/26/2019 to 05/26/2019. Before we proceed with the experiments, we conduct exploratory

data analysis to understand the characteristics of data. Table 5.3 shows the summary statistics of the dataset. The table shows that there is a total of 11587 obser-

Table 5.3: Summary Statistics for Buffalo Bayou Dataset

Statistic	**Discharge (cft/sec)**	**Precipitation (inch)**	**Gage height (ft)**
count	11587	11587	11587
mean	402.1	0.0012	37.7
std	574.11	0.014	2.73
min	90.3	0	35.66
25%	130	0	36.14
50%	162	0	36.46
75%	345	0	37.89
max	5150	0.58	56.19

vations. The mean value water level is 37.7 feet. The maximum amount of rainfall is 0.58 inches. The rainfall amount, water-level, and discharge data were collected by sensors residing at the gage station owned by the USGS. We use box plots and histograms to graphically represent the data distribution. Using a box plot, we are able to see the data representation in their quartiles. The bottom and top of the box represent the first and third quartile, and the band inside the box represents the second quartile or median. There is a whisker in the box plot which extends to the most extreme data points; that is, the largest and smallest values which are not outliers, and the value should be no more than 1.5*IQR (interquartile range) from the box. Any data outside the whisker is considered an outlier. Figure 5.1 shows the line plots of all variables in the Buffalo Bayou dataset.

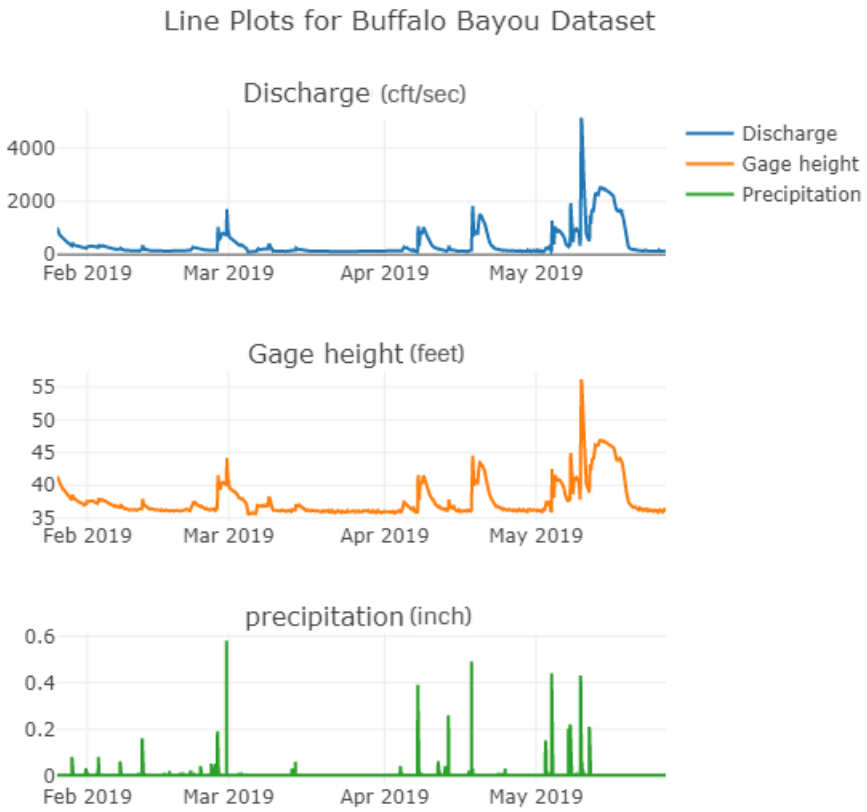


Figure 5.1: Line Plots for Buffalo Bayou Dataset

As we can see from the line plots, there is a high correlation between discharge and gage height/water level. This is obvious because as the discharge, i.e., the volume of water flowing across a particular measuring point, increases, the water level also increases. Most of the rainfall values are 0 since the data is recorded every 15 minutes, and the highest amount of rainfall received was a little less than 0.6 inches in May 2019. Hence, the line plot for precipitation is a horizontal line at 0 with occasional peaks indicating rainfall events.

Figures 5.2, 5.3, and 5.4 display box plots for the Buffalo Bayou dataset. As evident from the box plot for rainfall, most of the rainfall values are 0. Most water level values for the Buffalo Bayou stream lie between 35 and 40 feet. Most of the discharge values lie between 100 cft/sec and 1000 cft/sec.

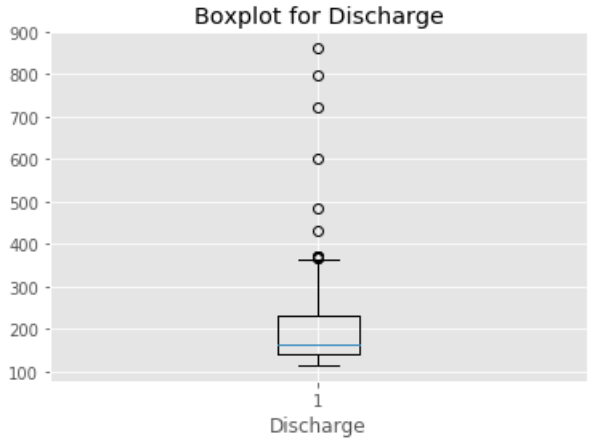


Figure 5.2: Box Plot for Discharge

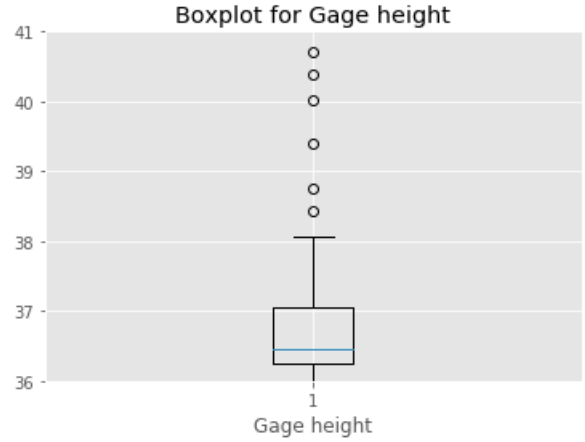


Figure 5.3: Box Plot for Water Level

Figure 5.4 displays box plot for non-zero rainfall events for the Buffalo Bayou

dataset.The box plot for rainfall is not for the whole dataset, since 96.5% of the dataset has a rainfall value of 0, and the box plot will show all the non-zero values as outliers. Therefore, the box plot only includes rainfall values that are not zero.

The histogram in Figure 5.5 indicates that there are only 309 rainfall events be-

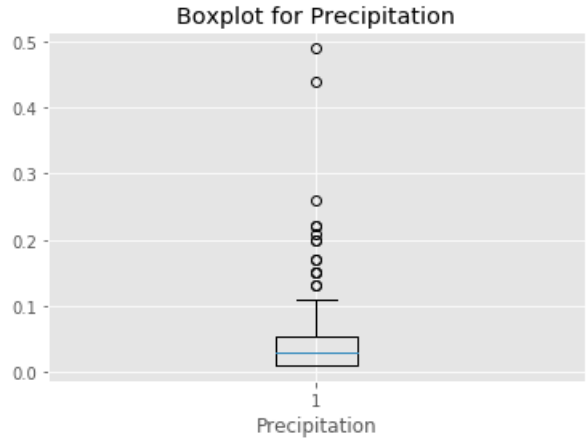


Figure 5.4: Box Plot for Rainfall

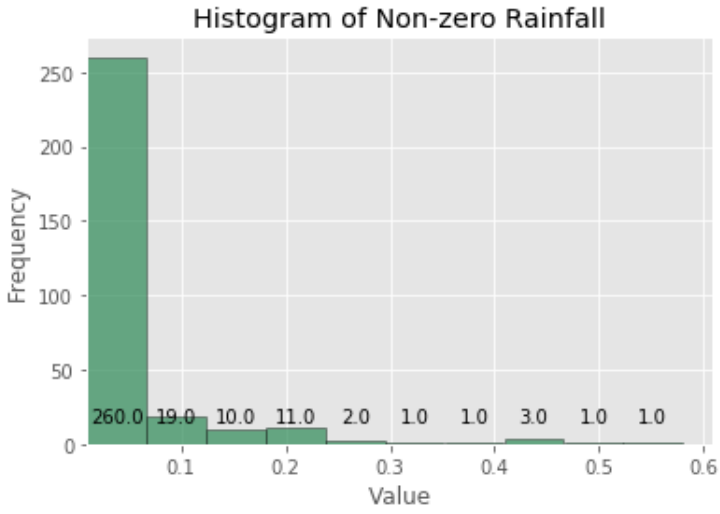
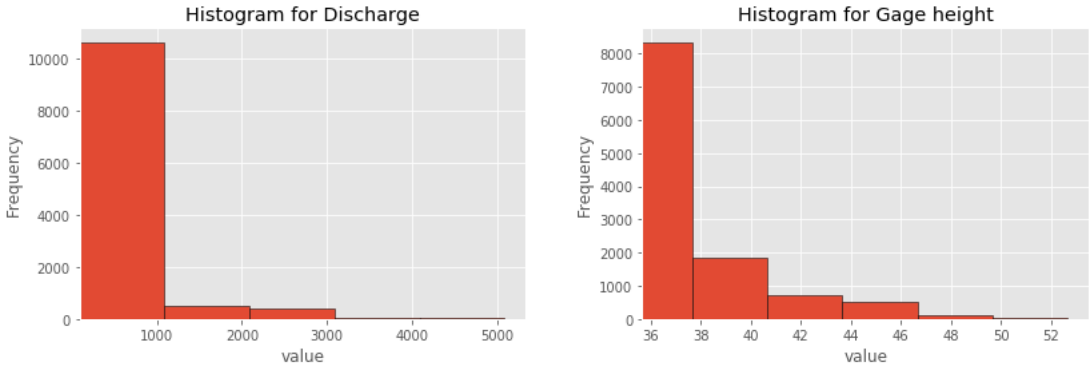


Figure 5.5: Histogram for non-zero Rainfall

tween January 26, 2019 and May 26, 2019 which is about 2.6%. Most of them are

between 0.01 inches and 0.1 inches. Figure 5.6 shows histograms for the remaining variables. The highest amount of rainfall received is 0.6 inches. The histogram for



(a) Histogram for Discharge (b) Histogram for Water level

Figure 5.6: Histograms

gage height shows that most of the values are between 36 and 38 feet. The highest water level recorded is 52.6 feet. The histogram for discharge shows that most of the values are in the range between 90 cft/sec and 1000 cft/sec.

5.4.2 Visualization of Data Distribution for the Titusville Dataset

Table 5.4 shows the summary statistics for the Titusville dataset. There is a total of 12389 observations. The maximum amount of rainfall Titusville received is 9.8 mm. The maximum soil moisture is 0.388%.

Table 5.4: Summary Statistics for the Titusville Dataset

Statistic	**precipitation**	**soilMoisture**
count	12389	12389
mean	0.077424	0.218107
std	0.588639	0.063491
min	0	0.128
25%	0	0.172
50%	0	0.196
75%	0	0.244
max	9.8	0.388

We explore the distribution of soil moisture using a histogram in Figure 5.7. This dataset contains a time series of rainfall and soil moisture values between January 1, 2018 and June 2, 2019. The histogram shows the distribution of soil moisture values in the top 0.05m of the soil. Most of the soil moisture values are in the range of 0.01% to 0.25%. Based on the range of soil moisture values for different types of soils, we can say that the soil type of Titusville in Florida is sandy [6]. The histogram shows that it is a bimodal distribution where soil moisture increases to 0.17%, then decreases until 0.32%, then again increases to the second-highest peak at 0.33% and then again decreases.

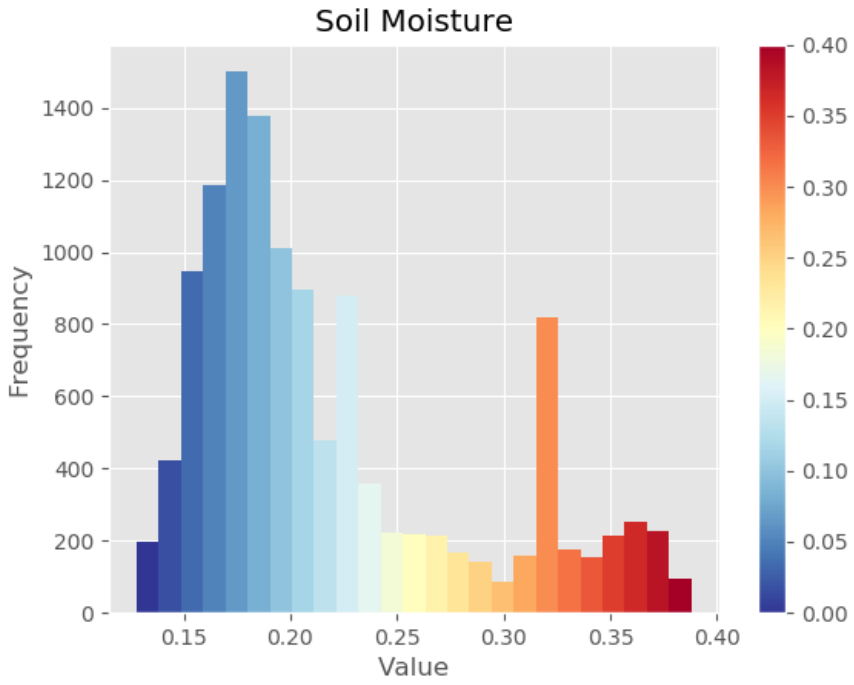


Figure 5.7: Histogram for Soil Moisture

The line plots for the Titusville dataset are shown in Figure 5.8. We can see that there is some correlation between rainfall and soil moisture. However, soil moisture values are most often constant for consecutive time steps. This reflects the sandy nature of the soil, which has low absorption capacity and therefore causes a higher chance of rainwater running down the area into nearby streams, causing an increase in the water level. This means that studying the relationship between rainfall and soil moisture and consequently being able to predict soil moisture values is very significant for predicting floods.

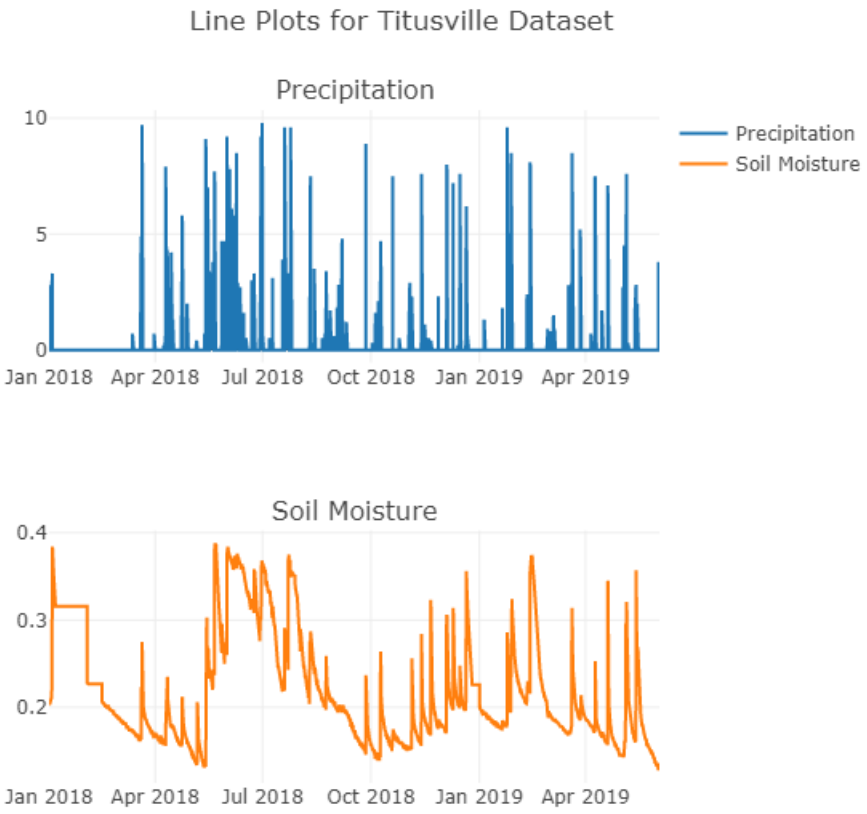


Figure 5.8: Titusville Dataset Line Plots

5.4.3 Data Stationarity

In time series forecasting, it is a good practice to make the series stationary, that means remove any systematic trends and seasonality from the series before modelling the problem. Statistical modelling and deep learning techniques on stationary time series are easier and more effective. A stationary series is one in which the properties like mean, variance, and covariance do not vary with time [22]. It's

clear that predicting future values for a stationary dataset is easier than predicting for one that is not stationary. Many methods exist to check whether a time series is stationary or non-stationary. In our research, we use statistical tests.

Statistical Tests

Statistical tests make strong assumptions about data stationarity. They are used to inform us if the null hypothesis needs to be rejected or accepted. One of the most widely used tests for checking the stationarity of multivariate time series data is the Johansen's test. To compute the Johansen's test, we use the package statsmodels.tsa.stattools.vecm [43] in Python and the function coint_johansen. If the eigen values are less than 1, then we can say that the time series data is stationary. Table 5.5 shows the eigen values for the Buffalo Bayou dataset.
The eigen values for all the features are less than 1. Hence we confirm that the time

Table 5.5: Eigen Values of Johansen's Test for the Buffalo Bayou Dataset

Feature	**Eigen Values**
Discharge	0.3388
Precipitation	0.0028797
Gage Height	0.000247186

series data of Buffalo Bayou dataset is stationary. Table 5.6 shows that the eigen values for both soil moisture and rainfall are less than 1. Therefore, we confirm that the time series data of the Titusville dataset is stationary.

Table 5.6: Eigen Values of Johansen's Test for Titusville Dataset

Feature	Eigen Values
Precipitation	0.2629
Soil Moisture	0.001874

5.5 Auto-correlation

Auto Correlation Plots

Auto correlation is defined as a mathematical representation of the degree of similarity between a given time series and a lagged version of itself over successive time intervals [51]. When we have time series data where future values can be predicted based on preceding values of the same time series, then we can say that the time series data exhibit auto correlation.

The time lag depends on the problem, and in order to determine the time lag for a given set of time series data, we use an auto correlation graph for each feature. For example, in the domain of flood prediction, the future values of water level might depend on the water levels of two or more past time steps. Sometimes, if we examine data too far in the past, then the predicted future values might not be reliable. In order to determine a suitable time lag for a given time series, we can plot an auto correlation graph, and based on the correlation values at each time lag, we can pick a suitable time step. Figures 5.9, 5.10 and 5.11 show the auto correlation plots for the Buffalo Bayou dataset.

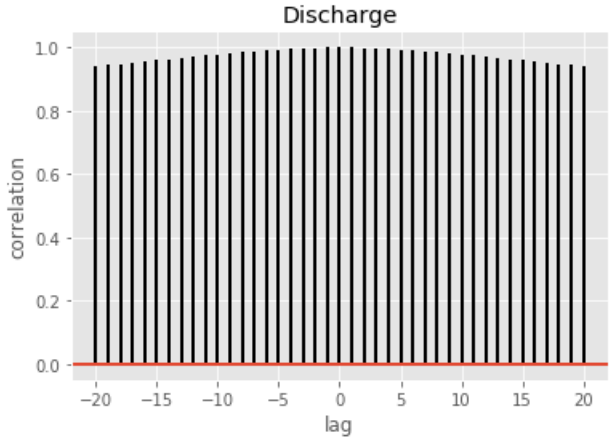


Figure 5.9: Auto Correlation Plot for Discharge of the Buffalo Bayou Dataset

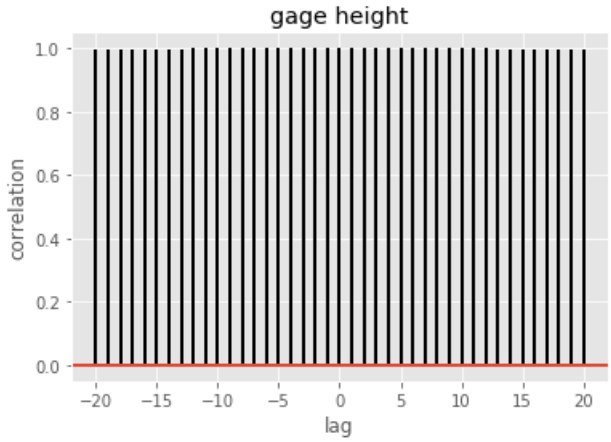


Figure 5.10: Auto correlation Plot for Gage Height of the Buffalo Bayou Dataset

From the correlation plots shown, we see that discharge and gage height for the current time step are highly correlated with those of previous time steps up to a lag order of 20. The auto correlation for precipitation decreases with the number of time steps, i.e., only observations from recent time steps are relevant. Hence

we choose a lag of four as a compromise between discharge, gage height, and precipitation. In order to reliably and accurately predict future water levels using the Buffalo Bayou dataset, we need to use a look-back window of four, which means we shift the whole time series four time steps back and use the lagged series to calibrate the model.

From the correlation plots shown in Figures 5.12 and 5.13, we see that soil mois-

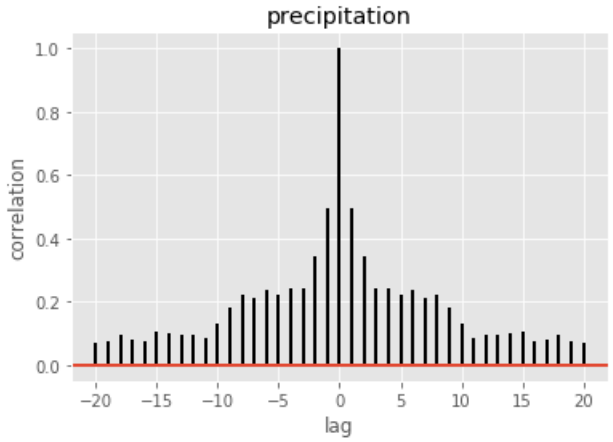


Figure 5.11: Auto correlation Plot for Rainfall of the Buffalo Bayou Dataset

ture for the current time step is highly correlated with the soil moisture levels of previous time steps up to a lag order of 20 or more (not shown in the plot). In contrast, the rainfall values at the current time step are significantly correlated with the values at the previous two time steps. After that, correlation drastically decreases. Hence, a lag order of two is suitable for this case.

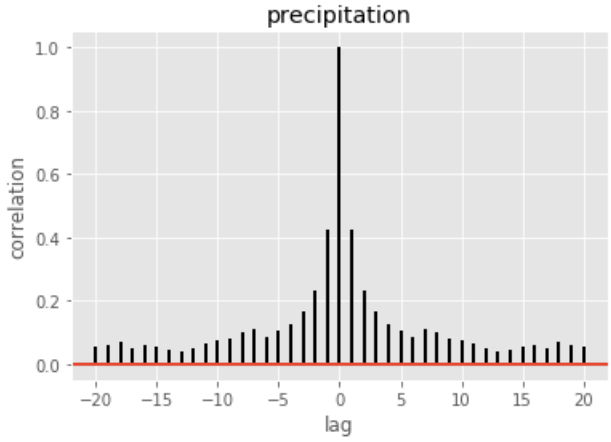


Figure 5.12: Autocorrelation Plot for Precipitation of the Titusville Dataset

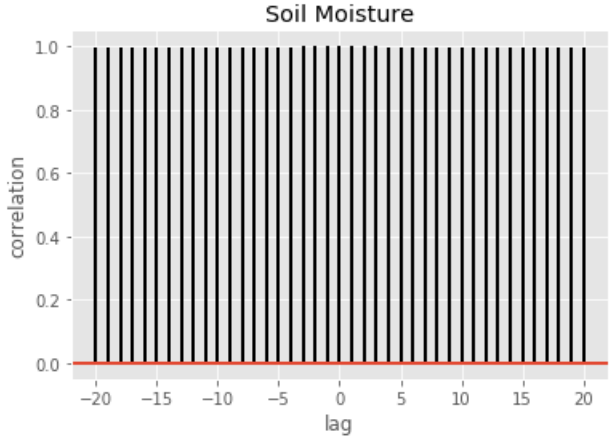


Figure 5.13: Autocorrelation Plot for Soil Moisture of the Titusville Dataset

Correlation Matrix

A correlation matrix can be a graph or a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables [24]. This matrix is used as an additional input to data analysis, helping

us model the time series. The default algorithm used by corr() in Python is the Pearson correlation. Table 5.7 shows the correlation matrix for the Buffalo Bayou dataset using Pearson's algorithm. We can conclude that gage height and dis-

Table 5.7: Correlation Heat Map for the Buffalo Bayou Dataset

Feature	**Discharge**	**Precipitation**	**Gage height**
Discharge	1	0.104409	0.985585
Precipitation	0.104409	1	0.099831
Gage height	0.985585	0.099831	1

charge are highly correlated. Precipitation has very low correlation with the other two features. Table 5.8 shows that rainfall and soil moisture are not significantly correlated.

Table 5.8: Correlation Matrix for the Titusville Dataset

Feature	**Precipitation**	**Soil Moisture**
Precipitation	1	0.086
Soil Moisture	0.086	1

Chapter 6

Experimental Results

In our research, we utilized two different datasets. These datasets are described in detail in Chapter 5. We used the Titusville dataset to analyse the relationship between rainfall and soil moisture using VAR and LSTM which are described in Chapter 5. We used the Buffalo Bayou dataset to build models using various architectures and implement different multi step time series forecasting strategies described in Chapter 4 and compare their performances.

6.1 Water Level Forecasting

As described earlier, we used the Buffalo Bayou dataset for water level forecasting. For all the experiments, we used the following hyperparameters after performing hyperparameter tuning as shown in Table 6.1.

Table 6.1: Hyperparameters

Hyperparameter	Value
Number of Neurons	200
Optimizer	Adam
Loss	Mean Absolute Error
Batch Size	100
Epochs	100

6.1.1 Architecture: Single-layer LSTM

6.1.1.1 One-step Ahead Water Level Forecasting

Scenario 1: Predicting all three input variables

Figure 6.1 shows the architecture used. As shown, we use only one hidden layer of LSTM, and the output layer is a dense layer.

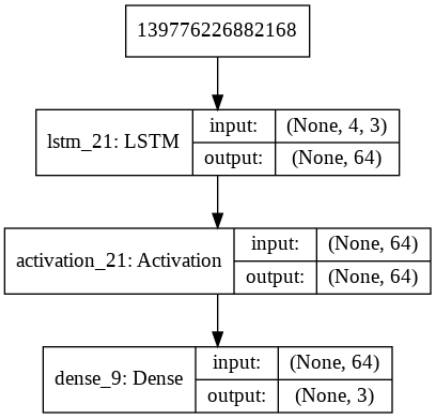


Figure 6.1: Single-layer LSTM

The model takes in the values of all three variables at the previous four time steps

and predicts all three variables, rainfall, water level, and discharge for the next step, as shown in Figure 6.1. Of the dataset, 67% is taken as training data, and the rest is taken as test data. Each of the input variables are standardized before training the model. Table 6.2 shows RMSE and MAE values for the water level in feet for the test set.

Table 6.2: Scenario 1: Performance Metrics – Single-Layer LSTM

RMSE	**MAE**
1.18ft	0.97ft

Upon observing the RMSE and MAE values for water level, we conclude that the predicted water level is off by 1.18 feet, on average. Visualizing the predicted values of the whole test set does not give an accurate picture of how well the model performed. Therefore, we choose to visualize only a subset of the test set of 100 samples. Figures 6.2, and 6.3 shows the line plots for observed and predicted water levels.

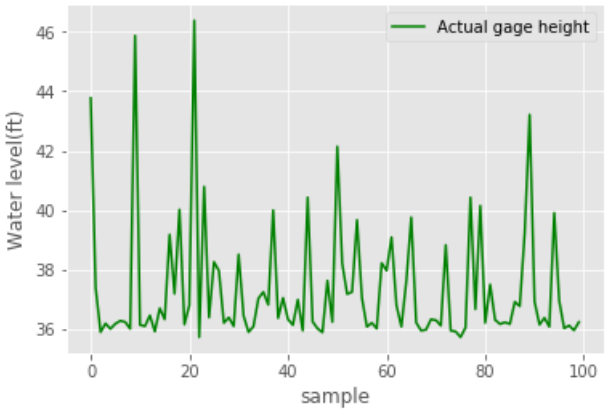


Figure 6.2: Scenario 1: Observed Water Level

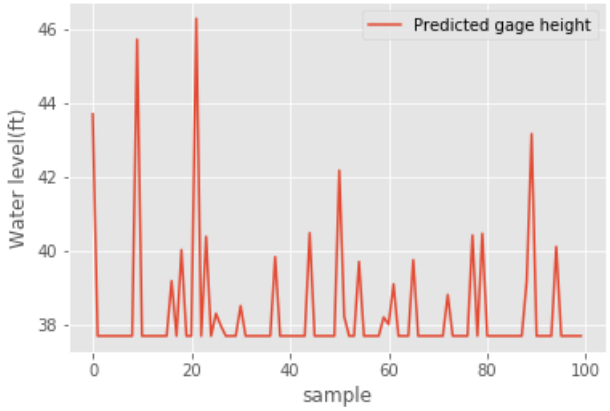


Figure 6.3: Scenario 1: Predicted Water Level

The predicted values for the test set show that the model is not capable of predicting water levels below a certain level, which in this case is 37 feet. This is quite evident because the mean absolute error of the model on the test set is 0.97 feet or the RMSE is 1.18 feet, which means the predicted water level values are always above a certain value, which is 0.97/1.18 feet off from the actual water level. There might be several reasons leading to this result. One reason is that the distribution of water levels in the training set and the test set is drastically different. For example, the water level values in the training set may all be between 36 and 37 feet, whereas in the test set, they are in between 40 and 43 feet. In order to avoid this issue, we shuffled the samples in the whole dataset and then split it into training and test sets to make the distribution of the target variable consistent across both sets. Even then, the model performs as shown above. We later show that the encoder/decoder architecture of LSTM is good at preventing this scenario.

Scenario 2: Predicting only discharge and gage height

Since we saw in Chapter 5 that the correlation between rainfall and discharge/gage height is very low, predicting rainfall along with discharge and gage height degrades the performance of the model. It is also evident that predicting rainfall is difficult and that there are several other factors that influence rainfall. Hence, we trained the model to predict only discharge and water level. This is evident in Table 6.3.

Table 6.3: Scenario 2: Performance Metrics – Single-Layer LSTM

RMSE	**MAE**
1.32ft	0.92ft

The MAE for Scenario 2 is slightly less than that for Scenario 1. Figures 6.4 and 6.5 show the observed and forecasted water levels for 100 random samples in the test set.

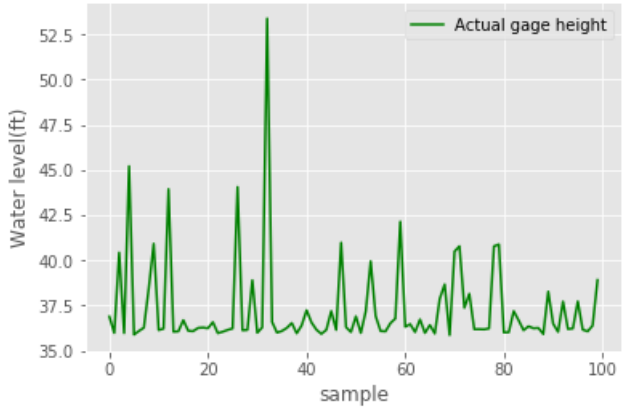


Figure 6.4: Scenario 2: Observed Water Level

The model is still not able to predict water levels below a particular value, which

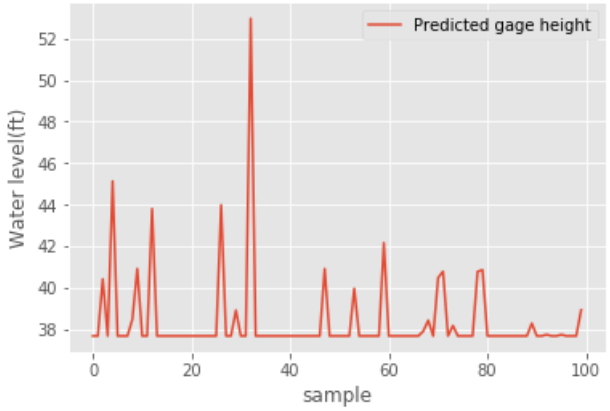


Figure 6.5: Scenario 2: Predicted Water Level

is observed to be around 37 feet.

6.1.1.2 Multi-step ahead forecasting

Now, we predict four time steps, meaning one hour into the future. We use the model that is trained based on Scenario 2 because predicting rainfall and using predicted values of rainfall instead of actual values in fact degrades the models performance. Figure 6.6 shows the observed water level, predicted water level using actual values of all three input variables (IMS_a strategy), and predicted water level using the predicted values of all three input variables (IMS_p strategy) for four time steps into the future. Figures 6.6 and 6.7 depict multi-step water level forecasts for four time steps. We can see that IMS_a performs better compared to IMS_p since it is quite evident that using predictions results in accumulation of error.

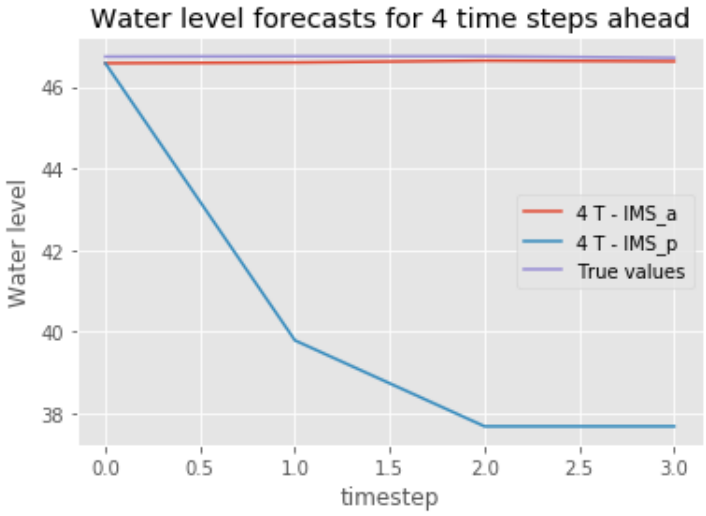


Figure 6.6: Sample 1: Comparison of Multi-step Ahead Strategies

Also in Figure 6.6, true water level values for four time steps are in the range between 46 and 47 feet, therefore IMS_a performs better when compared to the right plot where the actual water level forecasts are in the range between 36 and 37 feet. Note that in the IMS_p strategy, we use the predicted values for water level and discharge and the observed values for rainfall to achieve multi-step forecasting. We also applied these strategies using the model trained based on Scenario 1, but the results do not add any value and hence are not included here.

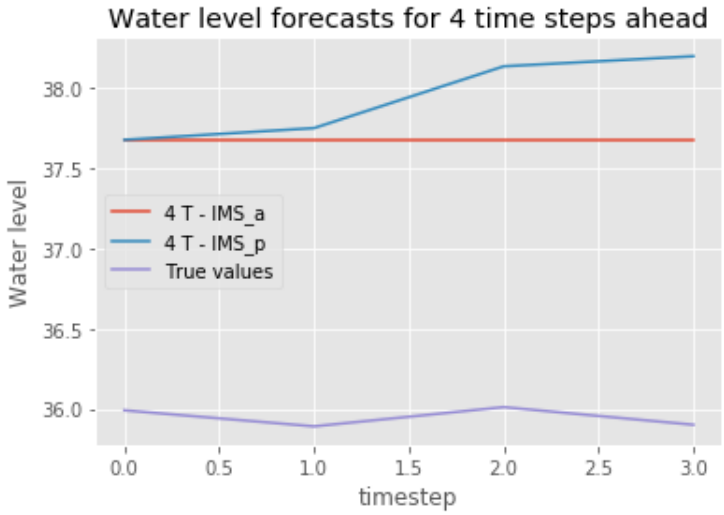


Figure 6.7: Sample 2: Comparison of Multi-step Ahead Strategies

6.1.2 Architecture: Stacked LSTM

The main aim of using the stacked LSTM architecture is to see whether adding more layers to the single-layer model or making the architecture deeper boosts the performance of the model. We use three hidden LSTM layers and one normal feed-forward output layer. Figure 6.8 shows the architecture of stacked LSTM.

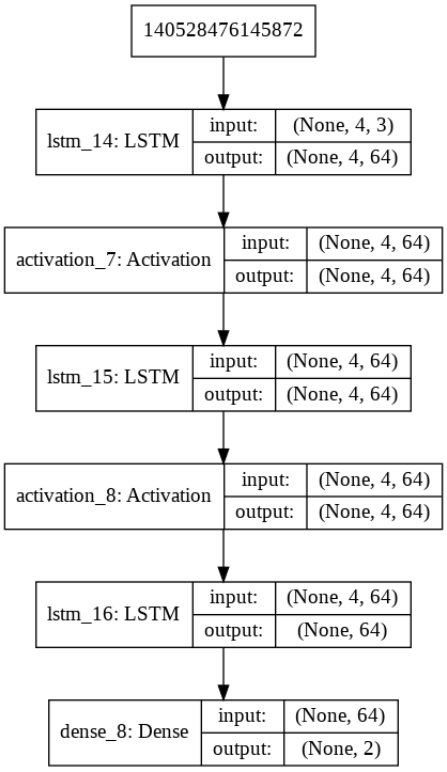


Figure 6.8: Deep LSTM

We observe that increasing the layers of the model did boost the performance slightly, and we then tweak the neural network model where the subsequent LSTM layers after the top layer are initialized with the hidden states and cell states of the previous LSTM layer. Generally, the internal states of each LSTM layer are initialized randomly. We only report the performance metrics (RMSE and MAE) since there is no major difference in the predictions of the water level on the test set. It is evident from Table 6.4 that using a deep architecture slightly improved the models ability to predict water levels.

Table 6.4: Performance Metrics - Stacked LSTM

RMSE	MAE
1.18ft	0.93ft

6.1.3 Architecture: LSTM Autoencoder

Figure 6.9 shows the architecture of the LSTM autoencoder used. The LSTM au-

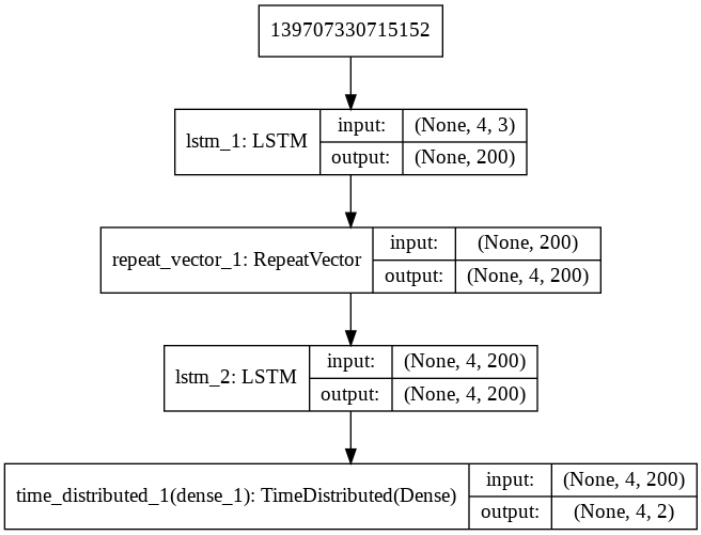


Figure 6.9: LSTM Autoencoder

toencoder consists of a single LSTM layer with 200 neurons. The decoder consists of another LSTM layer of 200 neurons. The output is a dense layer which is applied to each time step in the previous layer, thus producing a sequence of predictions. Therefore, this architecture is called seq2seq. RepeatVector acts as an adapter, connecting the encoder and decoder. The main purpose of the repeat vector is to repeat the output in the last time step of the previous layer four times, where four is the look ahead window. In summary, the encoder converts our

input sequences into a single vector that contains information about the entire sequence, then it repeats this vector n times (where n is the number of time steps in the output sequence, also called the look ahead window), and it applies an LSTM decoder to turn this constant sequence into the target sequence. We present the results for Scenario 2. Table 6.5 shows the RMSE and MAE of predicted water levels for four time steps.

Table 6.5: Performance Metrics for LSTM Autoencoder

Timestep	**RMSE**	**MAE**
t	0.077ft	0.03ft
t+1	0.096ft	0.04ft
t+2	0.130ft	0.05ft
t+3	0.198ft	0.06ft

As seen in Table 6.5, the RMSE or MAE for all of the time steps are significantly smaller than those of the single-layer LSTM/stacked LSTM architecture. This means the LSTM autoencoder is able to capture the dependencies of the variables between the time steps. We see that single-layer LSTM or stacked LSTM architectures were not able to learn or predict water levels that are below 37.6 feet. The LSTM autoencoder overcomes that problem. Also, it is obvious that the RMSE and MAE values increase as we go farther into the future. Figures 6.10 and 6.11 show the predicted and observed gage height for the look ahead window of four. These plots show the water level values for a random sample from the test set, meaning we randomly select a sample from the test set and feed it into the model.

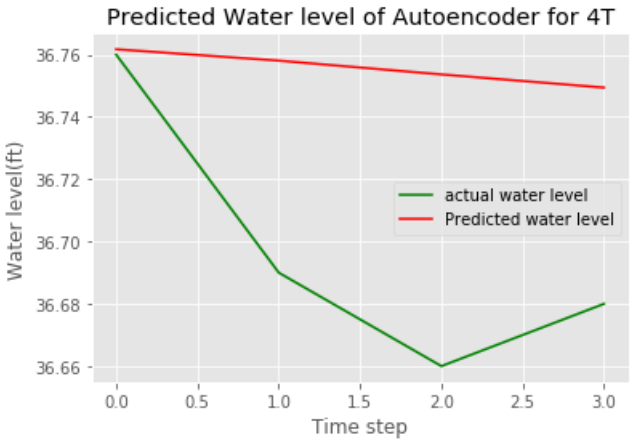


Figure 6.10: Sample 1 - Multi-step Ahead Forecasting Using the LSTM Autoencoder

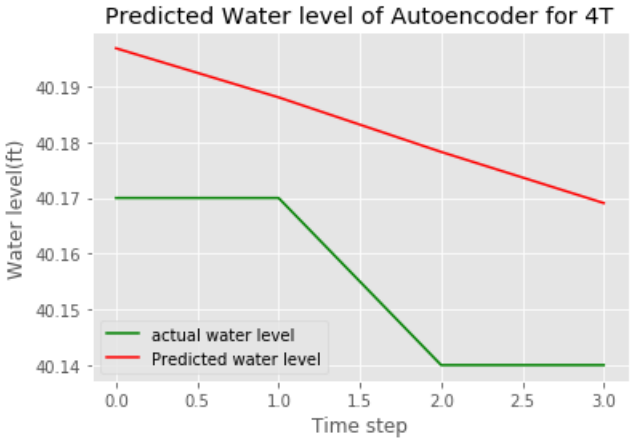


Figure 6.11: Sample 2 - Multi-step Ahead Forecasting Using LSTM Autoencoder

Figures 6.12 and 6.13 show observed and predicted water levels of 100 random samples for current time step t.

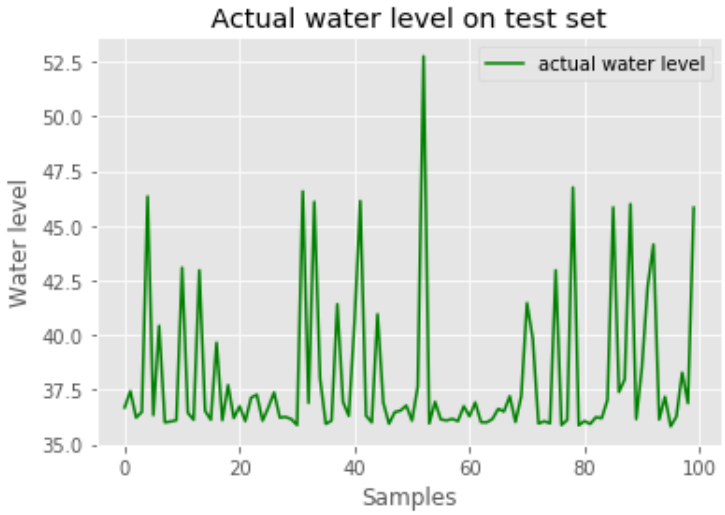


Figure 6.12: Actual Water Levels

Comparing Figure 6.13 with Figure 6.3, we see that the LSTM autoencoder can predict water levels below a threshold of 37 feet, while the single-layer LSTM or stacked LSTM could not. The fluctuations at the bottom of Figure 6.13 are well-captured, whereas the bottom of the plot is a flat line in Figure 6.3. The plots for predicted water levels at t+1, t+2, and t+3 time steps look similar to Figure 6.13 and hence are not shown here. We are not sure why the LSTM autoencoder performs more effectively than the other two architectures.

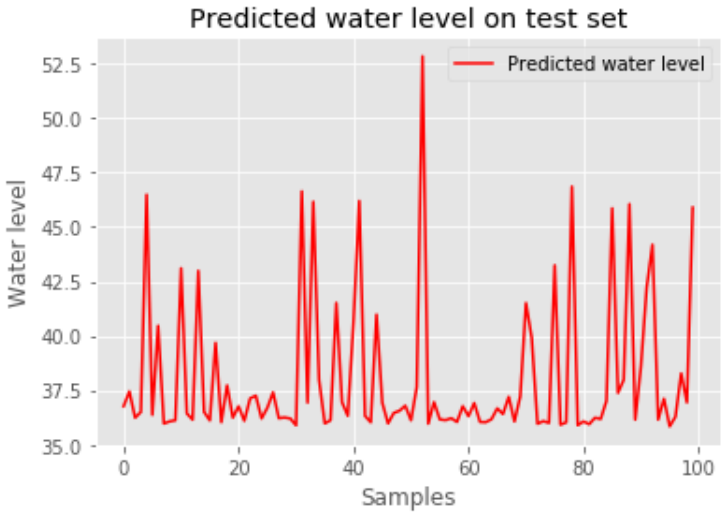


Figure 6.13: Predicted Water Levels Using the LSTM Autoencoder

6.1.3.1 Vector AutoRegressive (VAR)

Table 6.6 shows the performance metrics of VAR for Buffalo Bayou dataset. Since VAR inherently uses regression, a summary of the model gives us the value for the coefficient of determination R^2 as 0.99, which means the proportion of variance in the water level that is predictable from lagged versions of rainfall, discharge, and gage height is 0.99.

Table 6.6: Performance Metrics VAR for Buffalo Bayou

R^2	**RMSE**	**MAE**
0.99	1.52ft	1.03ft

Figure 6.14 shows that VAR poorly captures the temporal structure of the variables and is unable to predict water levels higher than 37.3 feet.

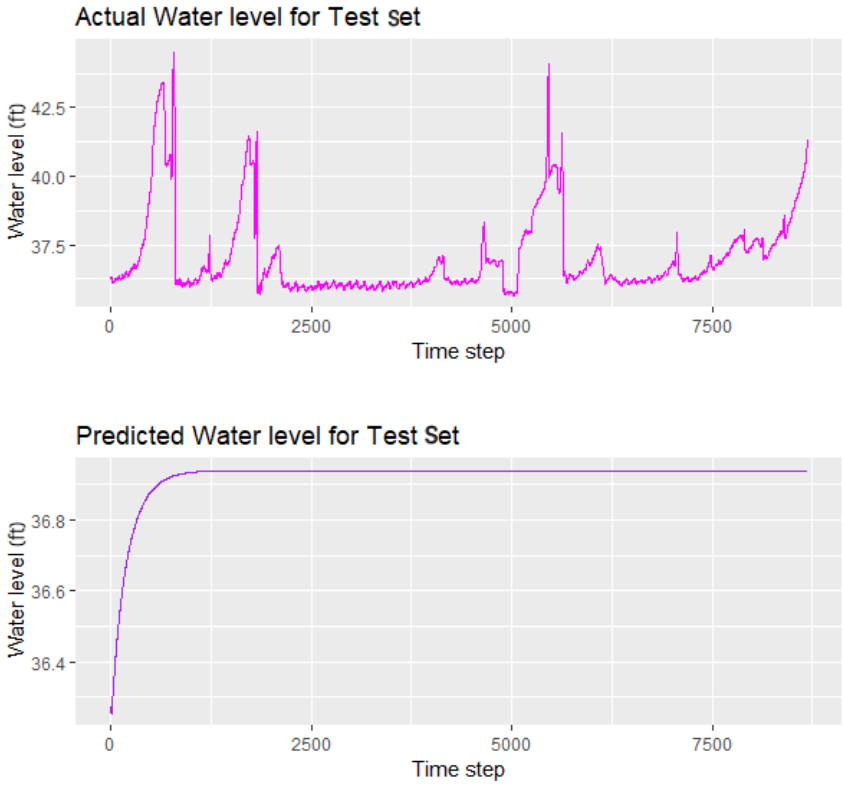


Figure 6.14: Actual and Predicted Water Levels Using VAR

6.2 Analysis of the Relationship between Soil Moisture and Rainfall for the Titusville Dataset

We used the Titusville dataset to analyze the relationship between soil moisture and rainfall. It is challenging to predict soil moisture given the sparsity of rainfall data. Most of the rainfall values are 0, and it is quite challenging to predict the effect of rainfall on soil moisture. It is challenging because the absorption capacity of soil depends on the soil texture and structure, and it is hard to collect information about the soil type of the location from which the data is gathered.

6.2.1 LSTM Autoencoder

We use the LSTM autoencoder to predict both soil moisture and rainfall given the values of both rainfall and soil moisture at past time steps. Each time step is one hour. The look back window and look ahead window lengths are four time steps. We can see from Table 6.7 that the RMSE and MAE for soil moisture content

Table 6.7: Performance Metrics of Soil Moisture Using LSTM Autoencoder

Timestep	**RMSE**	**MAE**
t	0.0066	0.0023
t+1	0.0081	0.0026
t+2	0.0091	0.0033
t+3	0.0098	0.0037

increases as we go farther into the future, and it varies, on average, between 0.6% and 0.9%. We run the model with 100 epochs, but the validation loss increases after 60 epochs, which leads to overfitting. Figure 6.15 shows the training and validation loss for 100 epochs. Table 6.8 indicates that the predicted rainfall values

Table 6.8: Performance Metrics of Rainfall Using LSTM Autoencoder

Timestep	**RMSE**	**MAE**
t	0.52mm	0.09mm
t+1	0.56mm	0.10mm
t+2	0.51mm	0.10mm
t+3	0.57mm	0.11mm

largely deviate from the actual values, suggesting several other factors result in

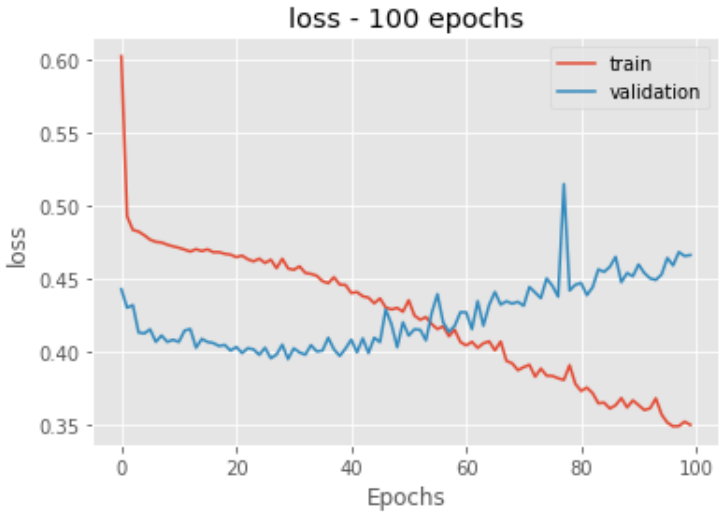


Figure 6.15: Loss for 100 Epochs

rainfall, and soil moisture is only one of them. Figure 6.16 shows the predicted soil

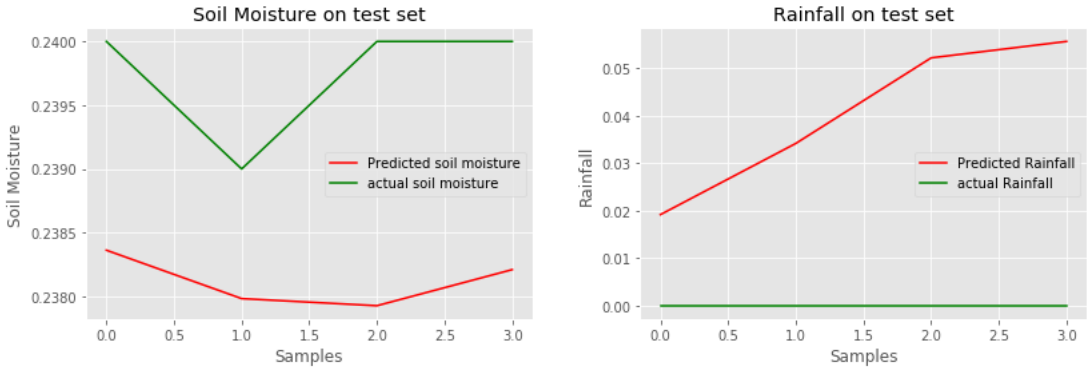


Figure 6.16: Predicted Soil Moisture and Rainfall - LSTM Autoencoder

moisture and rainfall for four time steps into the future for a random sample in the test set. If we observe the plot on the right of the figure, though observed rainfall values at four time steps are all 0, the predicted values significantly vary. This result indicates that there is a low correlation between rainfall and soil moisture as seen in Chapter 5. Hence, predicting both rainfall and soil moisture based on

past values results in inaccurate predictions for rainfall. Hence, we predict only soil moisture, and we report the performance metrics in Table 6.9. The RMSE and

Table 6.9: Performance Metrics of Soil Moisture Using LSTM Autoencoder

Timestep	**RMSE**	**MAE**
t	0.0058	0.0016
t+1	0.0071	0.0020
t+2	0.0088	0.0026
t+3	0.0104	0.0031

MAE values for the predicted soil moisture are lower than those observed when we predict both soil moisture and rainfall.

6.2.2 VAR

Figure 6.17 shows the predictions of the VAR model for the whole test set. We can say that, in our case study, machine learning approaches performed better than traditional regression approaches in capturing the temporal structure between the variables.

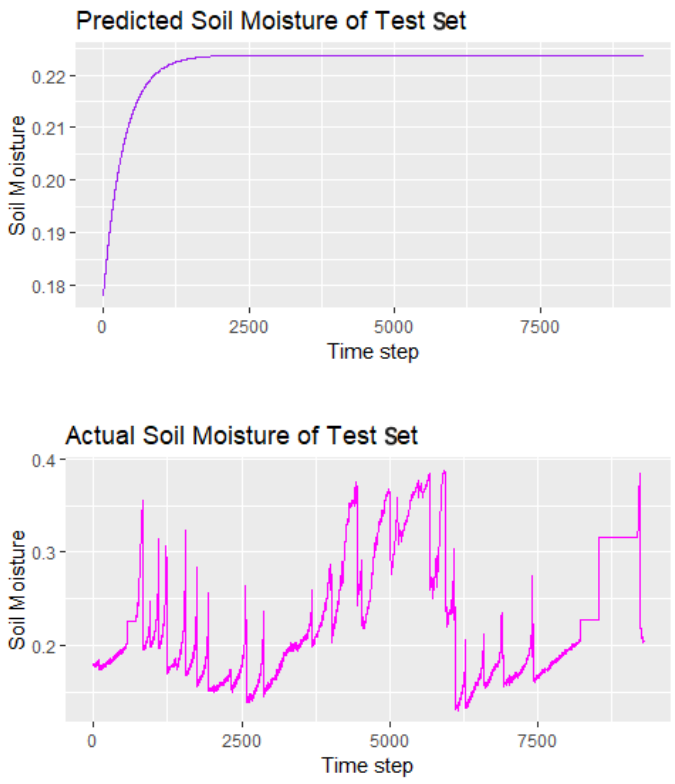


Figure 6.17: Predicted Soil Moisture - VAR

The model reaches a point where it cannot predict soil moisture higher than a certain value. Also, VAR offers limited flexibility in design choices compared to machine learning approaches. This is one limitation of parametric machine learning approaches. Table 6.10 shows the performance metrics of the VAR model for soil moisture predictions. The error is about 6%, which is higher than the error of the LSTM Autoencoder.

Table 6.10: Performance Metrics of VAR for Soil Moisture

R^2	**RMSE**	**MAE**
0.994	0.06	0.05

Figure 6.18 shows the predicted rainfall using the VAR model. We can see that the rainfall predictions increase for each time step and then reach a value beyond which the model cannot predict. However, the actual rainfall data is characterized by 0s with occasional peaks caused by rainfall events. This type of temporal structure is not captured by VAR. Hence, we conclude that predicting rainfall is challenging, as it depends on several factors. Table 6.11 shows the performance metrics of VAR for rainfall predictions. We see that the coefficient of determination

Table 6.11: Performance Metrics of VAR for Rainfall

R^2	**RMSE**	**MAE**
0.143	0.5822	0.14

is very low, which means the proportion of variance in rainfall that is predictable from lagged versions of both rainfall and soil moisture is only 14%, which is very low.

Whenever rain falls, part of the rainwater is absorbed by the soil. The amount of absorption depends on several soil characteristics, like the infiltration capacity, wilting point, and soil texture. If we are able to predict how much rainwater runs off into nearby streams, then we can forecast water levels across streams. The above set of experiments for the Titusville dataset indicates that there is still

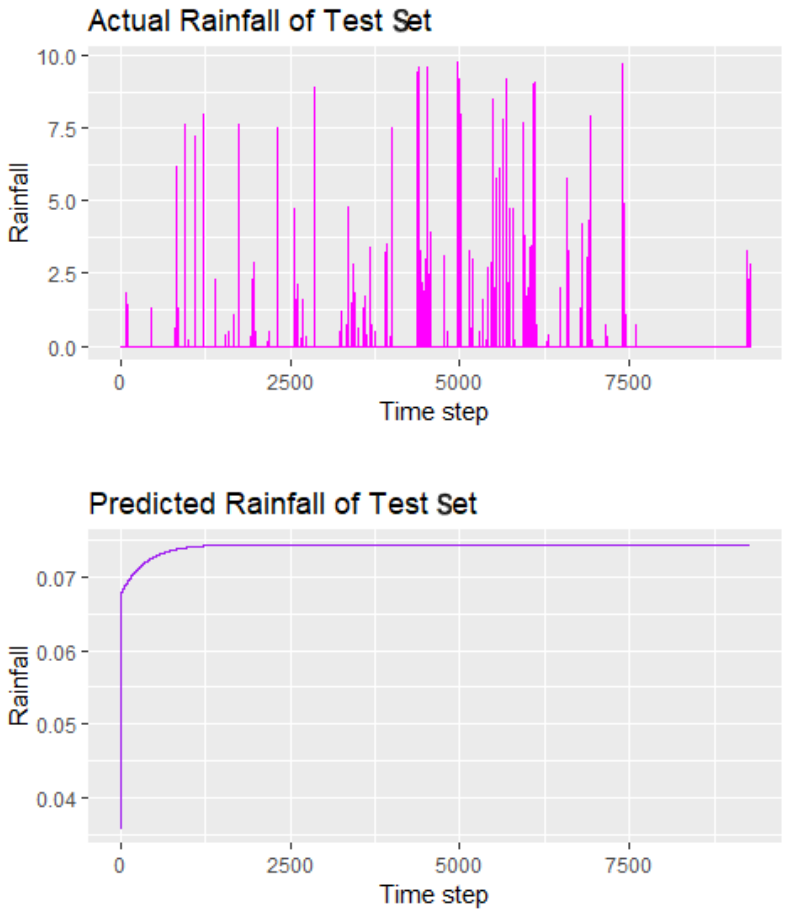


Figure 6.18: Predicted Rainfall - VAR

room for improvement in predicting soil moisture or rainfall using data-driven approaches, because these approaches rely solely on data. Getting a dataset that influence soil moisture and rainfall is very challenging. Also, since physics-based models are built using mathematical equations that govern the physical processes, developing a data-driven approach — which performs on par with the simulation-based models — to model such physical processes is very challenging. Therefore,

we conclude that the main purpose of this research is not to replace simulation-based models but rather to produce complementary or alternative approaches.

Chapter 7

FloodNet: A Deep Learning-based Water Level Forecasting System

The National Oceanic and Atmospheric Administration (NOAA) monitors the gage levels and stream flow across a stream and issues warnings using the Advanced Hydrologic Prediction Service (AHPD) [32]. It displays information regarding the magnitude and uncertainty of flood occurrence [33]. It publishes the forecasted water levels and discharge on its website. Its approach is capable of forecasting water levels for five days into the future with hourly updates. However, it does not have such a prediction system, which monitors water levels throughout the year, in place for Buffalo Bayou at West Belt Drive in Houston, Texas. So, we designed a system named **FloodNet - A Deep Learning Based Water Level Forecasting System** for Buffalo Bayou, which shows observed water levels for the past 2.5 hours and forecast water levels for the next 2 hours. It also

displays rainfall for the past 2.5 hours. It displays the flood history and different flood stages for the bayou. This application is hosted on a local system, and a user can refresh the page to view real-time water level forecasts. The USGS updates its data every hour. Our system fetches data when the user clicks Refresh and feeds the data into the model, thus generating forecasts for the next two hours. But since new data becomes available only hourly, users will see plots changes each hour. This system uses an LSTM autoencoder architecture, as described in Chapter 4, to produce water level forecasts. The web page displays the Buffalo Bayou watershed, a link that directs the user to view the USGS measuring station, and the river using Google Street View. It also displays the flood history, including all past hurricane events and their respective date of occurrence. As shown in Figure 7.2, we can see that the water level reached 71.22 feet during Hurricane Harvey on 08/31/2017.

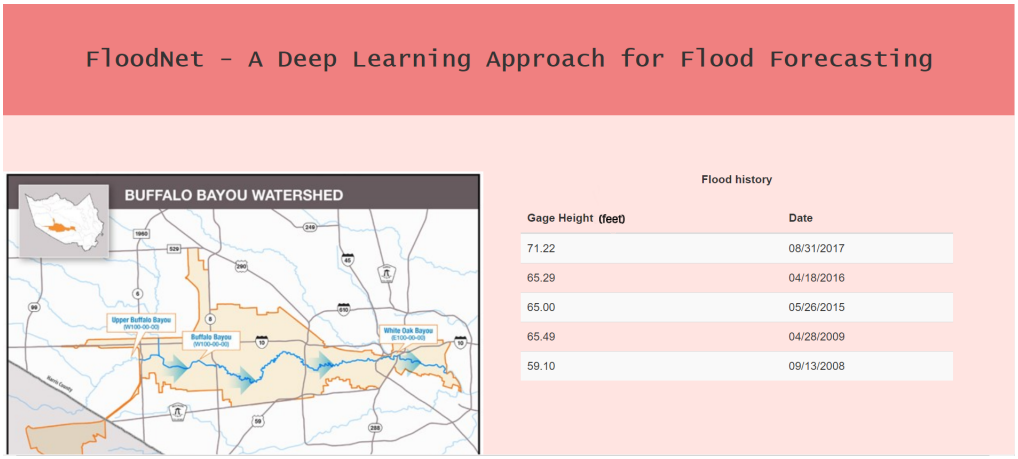


Figure 7.1: FloodNet - Watershed and Flood History

The web page also displays different flood categories and their respective gage heights set by the National Weather Service for Buffalo Bayou. For example, the

action stage is 59.2 feet and the flood stage is 62.2 feet. The web page displays the

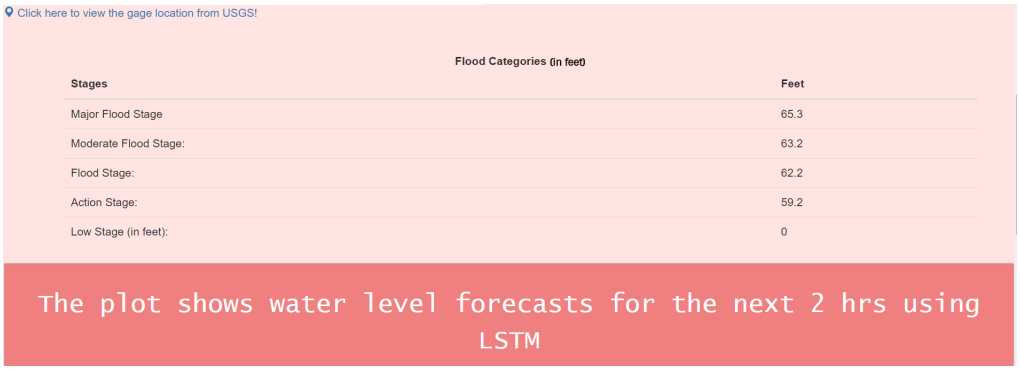


Figure 7.2: FloodNet - Flood Categories

observed water level for the past 2.5 hours and predicted water levels for the next 2 hours. It also displays the amount of rainfall received for the past 2.5 hours, as shown in shown in Figures 7.3 and 7.4. Figures 7.5 and 7.6 display the plots for

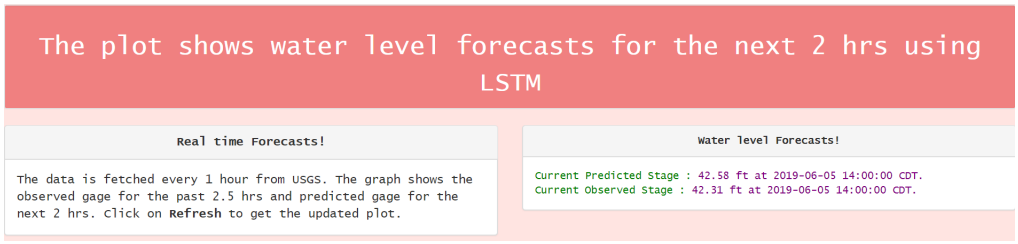


Figure 7.3: FloodNet - Forecasts

water level and rainfall. The plots on the left show the current observed and predicted gage height and different gage heights for the flood categories. The plots on the right show the actual amount of rainfall in inches in the last 2.5 hours. For example, it rained heavily on 06/05/2019. This resulted in increased water levels. The FloodNet system generated a series of forecasts for every hour between 2:30 AM and 2:00 PM. Figures 7.5 and 7.6 show the forecasts. The figures should be seen from left to right in each row. For example, in the first row, the figure on

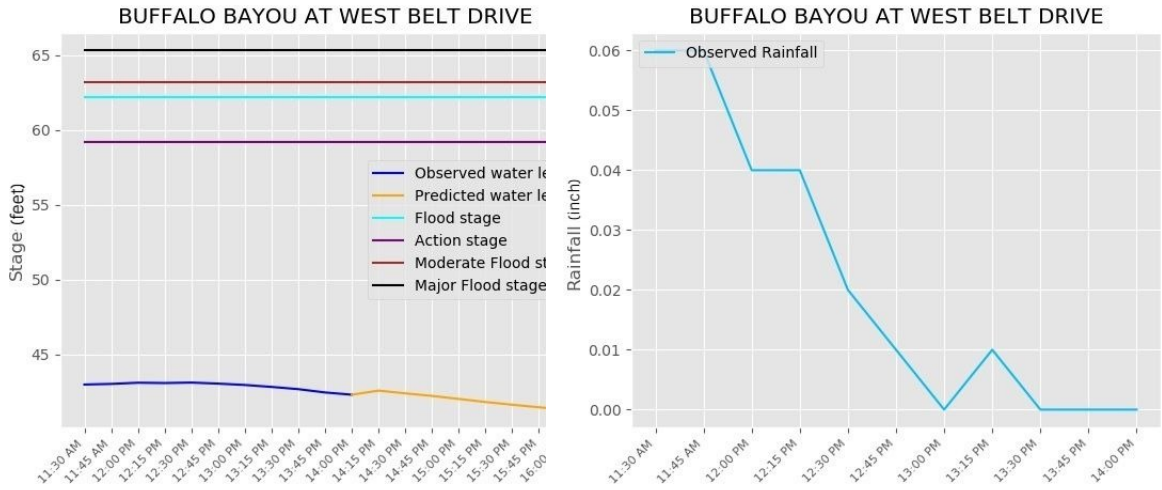


Figure 7.4: FloodNet - Water Level Forecasts and Observed Rainfall

left shows observed water levels from 2:30 AM to 5:00 AM and predicted water levels from 5:00 AM to 7 AM. The figure on the right shows observed water levels from 3:30 AM to 6:00 AM and predicted water level from 6:00 AM to 8:00 AM. It continues this way till the last row. The same explanation is applicable for Figures 7.7 and 7.8. The actual rainfall amounts for each hour starting with 2:30 AM are shown in Figures 7.7 and 7.8. We see that there is a significant amount of rainfall, and it rained continuously until 2:00 PM. From the water level forecasts, we can conclude that our system did pretty well in predicting the water levels. The forecasts show an increase in water level. After 2:00 PM, it stopped raining and hence water levels show a decreasing trend.

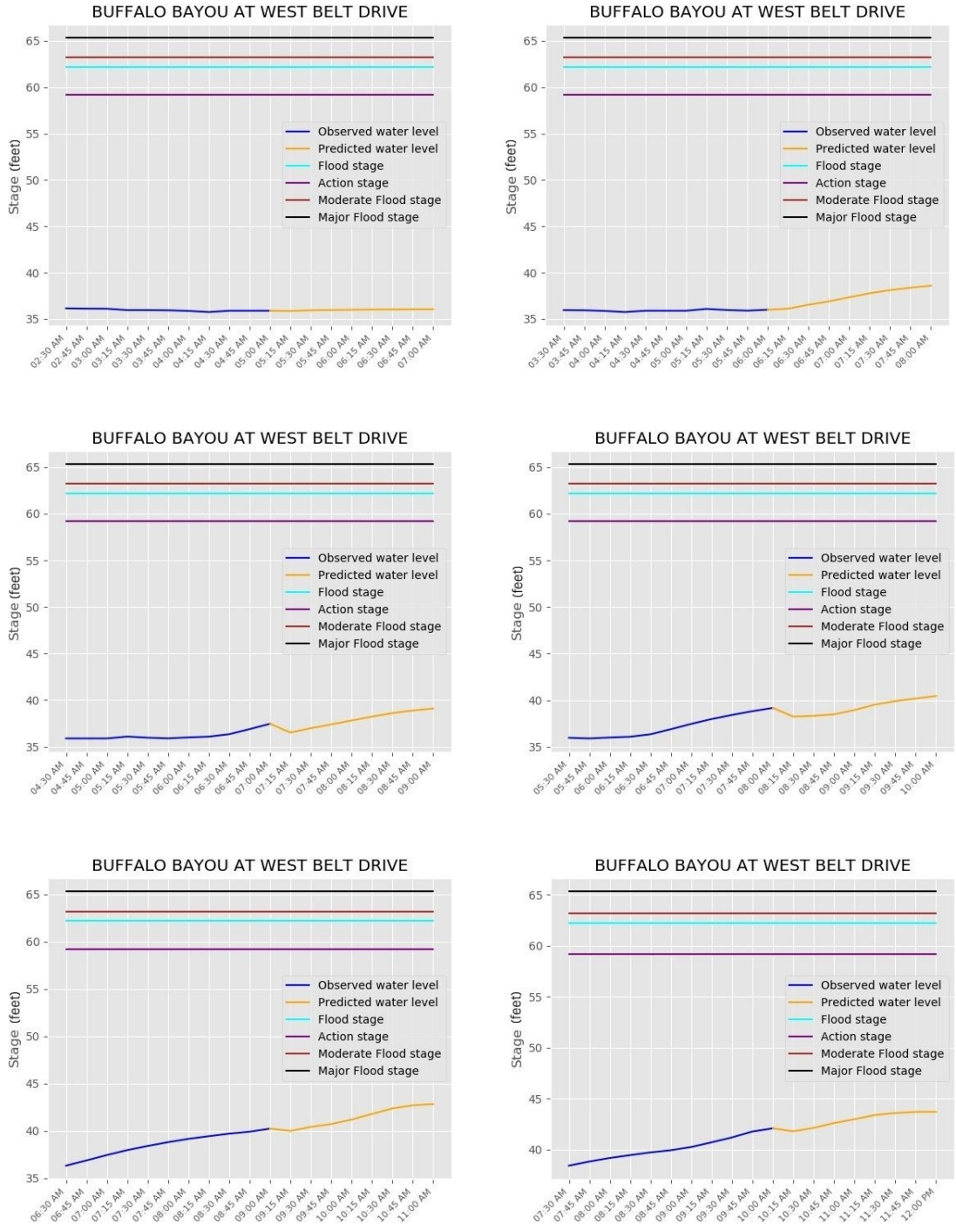


Figure 7.5: Part 1: FloodNet Forecasts on 06/05/2019

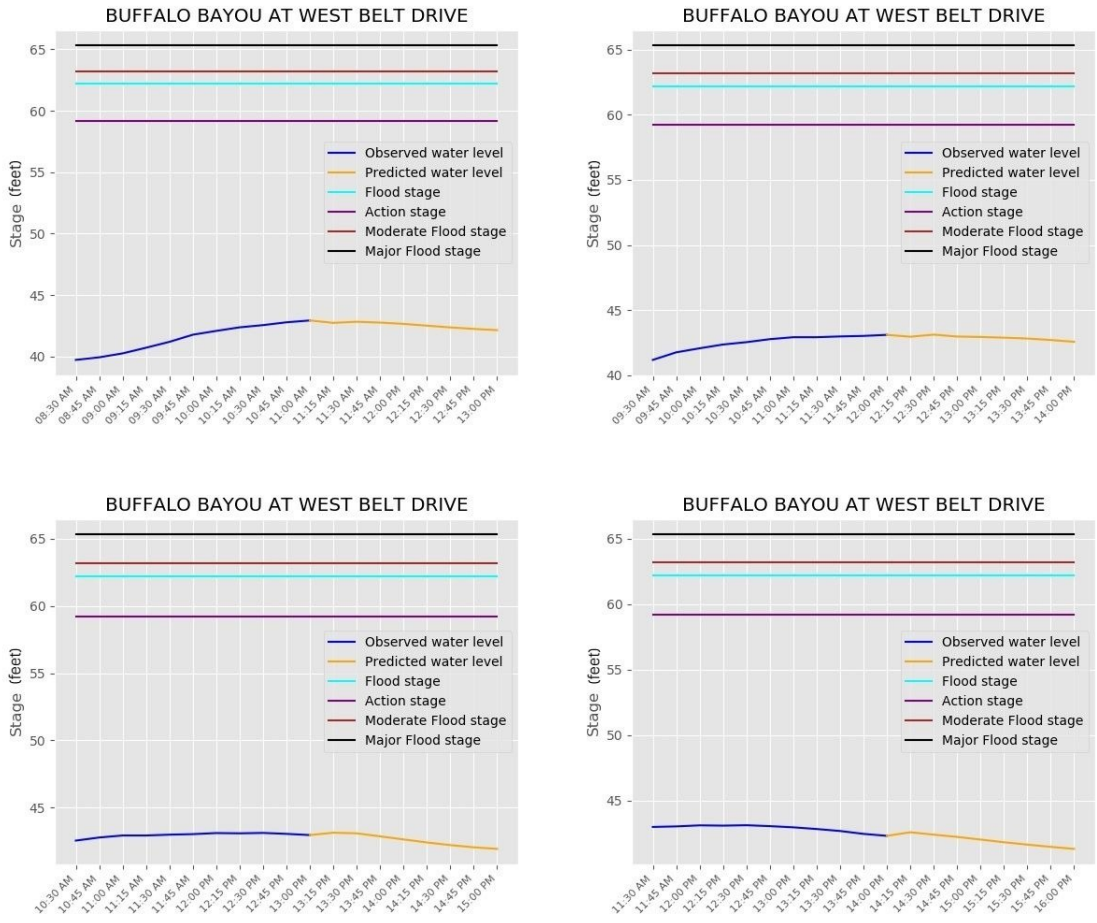


Figure 7.6: Part 2: FloodNet Forecasts on 06/05/2019

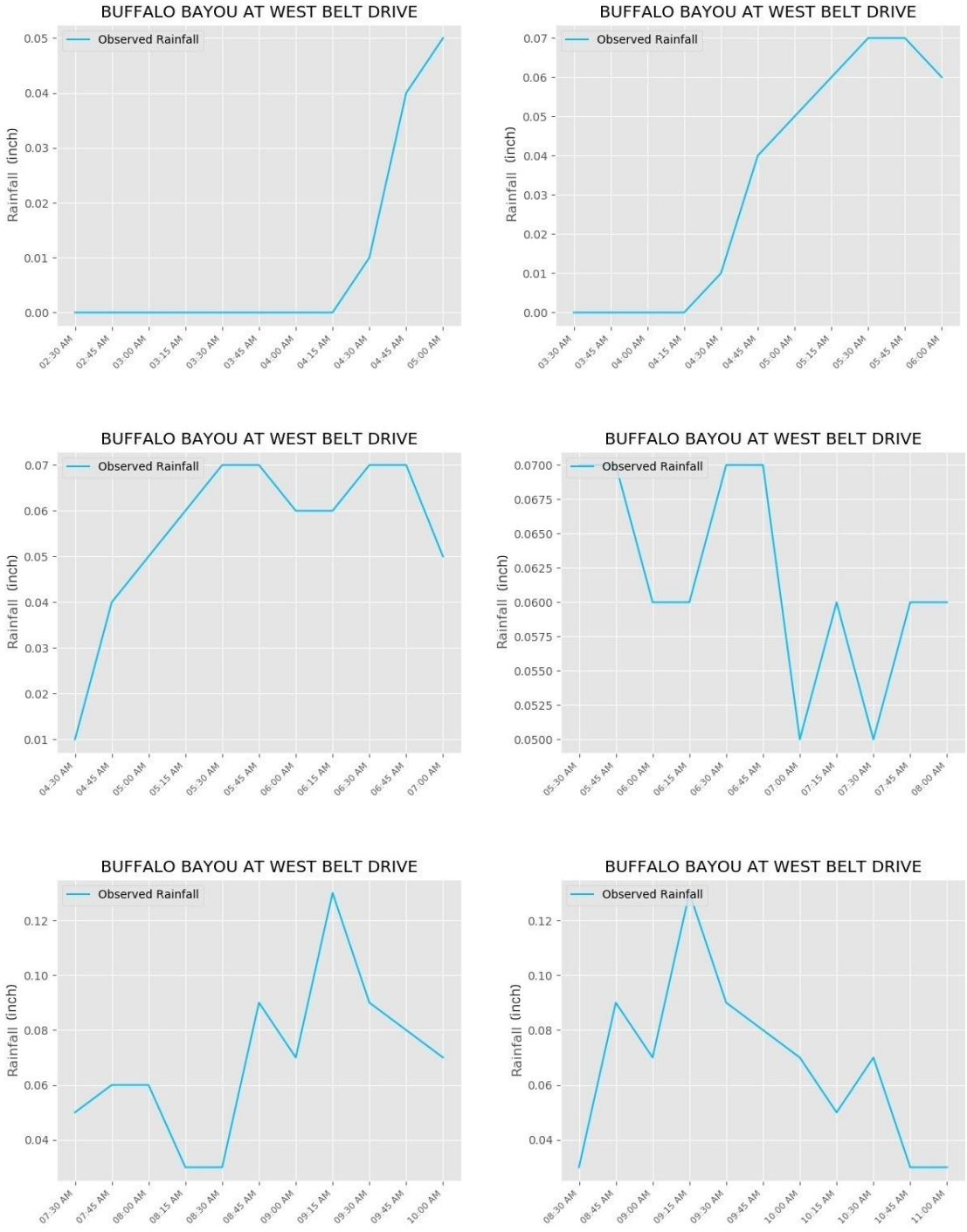


Figure 7.7: Part 1: Observed Rainfall Values on 06/05/2019

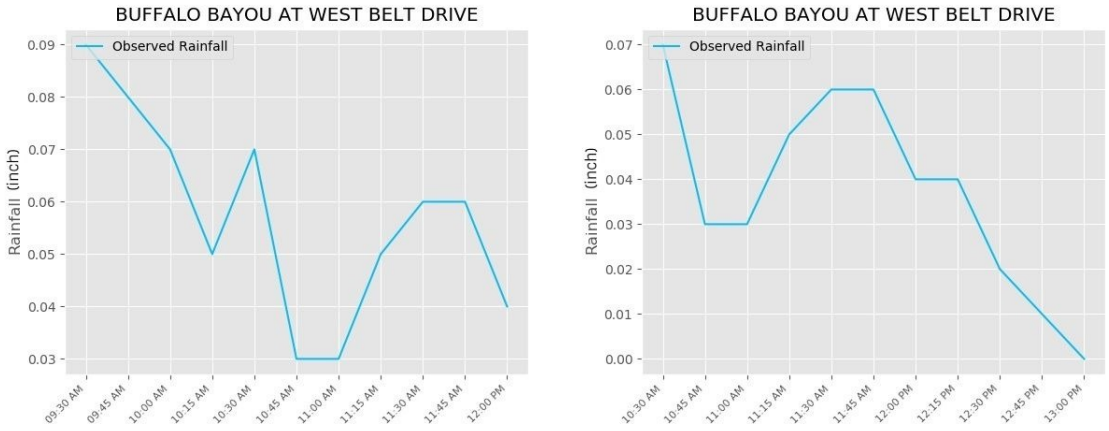


Figure 7.8: Part 2: Observed Rainfall Values on 06/05/2019

Chapter 8

Conclusion

Previous research work includes the design of a DAG-Based Multi-Target Prediction (DBMTP) Framework that models upstream-downstream dependencies as a directed acyclic graph. Parts of this research were presented at a Rice University data science conference in 2018 [8]. The purpose of this research is to generate alternative, complementary, and data-driven water level forecasting models using existing statistical models and recurrent neural networks that extrapolate the past into the future. We design and implement different architectures of LSTM to predict water levels and introduce tweaks to the models to improve their performance.

However, at the moment, evaluating the quality of these results is quite challenging, because designing experiments that compare these results with hydrology models is difficult and very time consuming due to the difference in the way data-driven and hydrology models work. To the best of our knowledge, there is

no commonly accepted water level prediction benchmark.

8.1 Challenges

Getting a dataset with all the necessary variables rainfall, discharge, water level, and soil moisture is difficult. Most sources contain only soil moisture data or only soil moisture and rainfall data. If we can model the rainfall-runoff using data-driven approaches, which is possible if we have a complete dataset, then we can analyze the importance of soil moisture in forecasting flooding. The experiments described in Chapter 6 related to soil moisture are the only results we can obtain for the given incomplete dataset; therefore, there is much more potential for research exploring the importance of soil moisture. However, for this research to be successful better datasets are needed.

The only way we can get a complete dataset is by merging datasets from different sources, but such datasets would be inaccurate for performing any type of analysis, as getting data for all the variables for a locations exact latitude and longitude is very difficult. The closest we got to obtaining this data was the dataset for Lost Creek in Utah. However, the location coordinates between the soil moisture data and the USGS data were mismatched. Due to the mismatch, the dataset will not generate accurate and reliable predictions, even if we try to fit the model to the data. One of the most promising resources we found is IFIS, which is a one-stop web platform to access community-based flood conditions, forecasts, visualizations, inundation maps, and flood-related information. We contacted the

person who heads IFIS, and we were informed that most of the sensors are new and have limited data. Nevertheless, the contact person sent us a share point link containing data, but it does not have soil moisture data. Instead, it contains Voltage Ratio measurements.

8.2 Future Work

There is much potential for improvement of the work presented in the thesis. As of now, the machine learning model is tuned to predict water levels for a specific geographic location. This means the model gives incorrect forecasts if we want to predict water level for a different location other than the one the model is trained on, because each bayou, stream, or creek is different in many aspects, and each bayou has different levels of flood categories. Therefore, there is a need to generalize flood forecasting. This goal requires a dataset having heterogeneous time series data about the different bayous for which we want to predict water levels and, hence, flooding. Since flooding can be categorized as a rare event, we can use feature-based machine learning models to tune the model. The feature-based machine learning approach can be accomplished using the automatic feature extraction capability of LSTM autoencoders or by using the **tsfresh** package in Python. We plan to design a system similar to [25]. This paper proposes an architecture that contains an LSTM autoencoder for automatic feature generation which then sends the features along with auxiliary inputs to LSTM forecasting model to finally generate forecasts. The authors of [25] used these forecasts to

achieve demand forecasting for rare events like Christmas and New Year's Eve in order to reduce the wait time of customers. The team at Uber used a dataset that

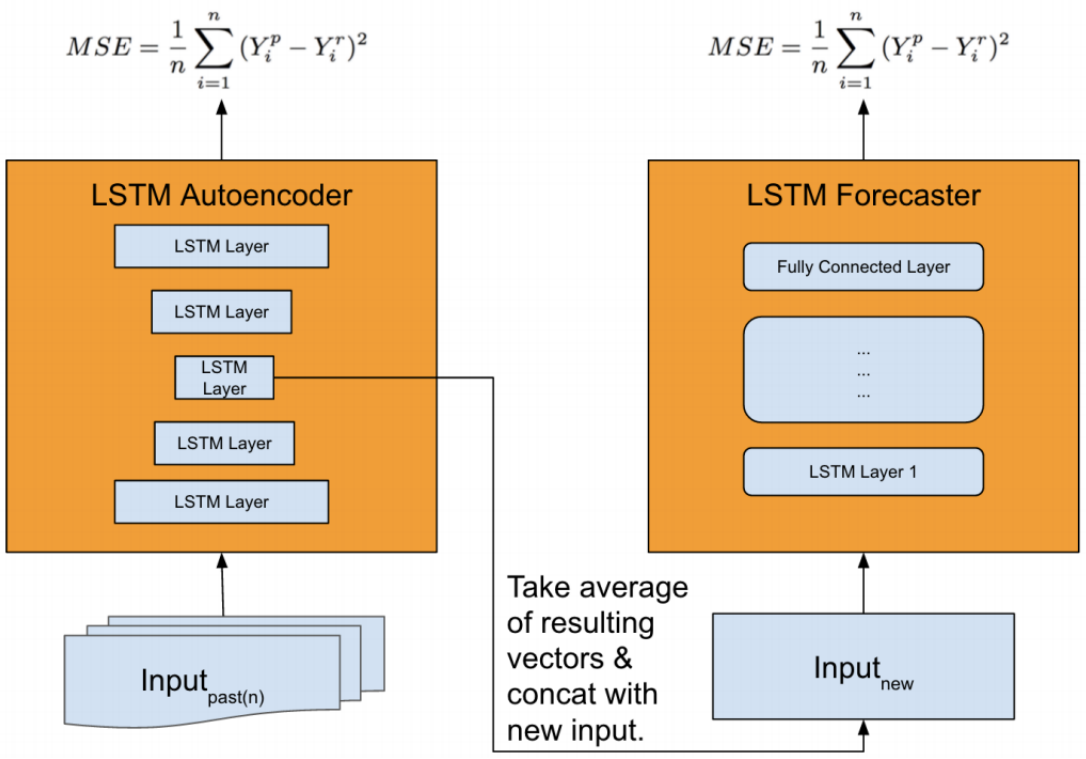


Figure 8.1: Architecture Used by Uber

contains heterogeneous time series data, i.e., time series data of rides taken across different cities. In order for us to use the same architecture to produce a generic model, the first step is to prepare a dataset that contains data for different bayous along with an identifier indicating to which bayou the data belongs to. The architecture is shown in Figure 8.1 [25].

Bibliography

[1] ATXfloods. ATXfloods. https://www.atxfloods.com/. Visited on 06/11/2019.

[2] J. R. Bence. Analysis of short time series: correcting for autocorrelation. *Ecology*, 76(2):628–639, 1995.

[3] M. Bray and D. Han. Identification of support vector machines for runoff modelling. *Journal of Hydroinformatics*, 6(4):265–280, 2004.

[4] Y. Cao. *Design and Implementation of a DAG-based Water Level Prediction Approach, Department of Computer Science, University of Houston*. PhD thesis, 2017.

[5] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[6] CornellUniversity. Soil hydrology. https://nrcca.cals.cornell.edu/soil/CA2/CA0212.1-3.php. Visited on 06/11/2019.

[7] CRAN. Introduction to r. https://www.r-project.org/about.html. Visited on 06/11/2019.

[8] C. Eick, A. Nemilidinne, Y. Cao, C. Hutapea, and K. Khaldi. A novel multi-target prediction framework and its application to flood forecasting. https://www.youtube.com/watch?v=vwBv8HUizLA. Visited on 06/11/2019.

[9] D. Entekhabi, E. G. Njoku, P. E. O'Neill, K. H. Kellogg, W. T. Crow, W. N. Edelstein, J. K. Entin, S. D. Goodman, T. J. Jackson, J. Johnson, et al. The soil moisture active passive (smap) mission. *Proceedings of the IEEE*, 98(5):704–716, 2010.

[10] Floodsite-Project. Ponding (or Pluvial Floods)," Floodsite Project, 2008. http://www.floodsite.net/juniorfloodsite/html/en/student/thingstoknow/hydrology/ponding.html. Visited on 06/11/2019.

[11] FloodTypes. Flood types. https://www.envirotech-online.com/news/water-wastewater/9/breaking-news/what-are-the-different-types-of-floods/31906. Visited on 06/11/2019.

[12] E. Funari, M. Manganelli, and L. Sinisi. Impact of climate change on waterborne diseases. *Annali dell'Istituto superiore di sanita*, 48:473–487, 2012.

[13] M. Gall, K. A. Borden, C. T. Emrich, and S. L. Cutter. The unsustainable trend of natural hazard losses in the united states. *Sustainability*, 3(11):2157–2181, 2011.

[14] M. Golnaraghi, C. Etienne, D. Guha-Sapir, and R. Below. *Atlas of Mortality and Economic Losses from Weather, Climate, and Water Extremes (1970-2012)*. World Meteorological Organization, 2014.

[15] HarrisCounty. Harris County Flood Warning System. https://www.harriscountyfws.org. Visited on 06/11/2019.

[16] S. Hochreiter. Recurrent neural net learning and vanishing gradient. *International Journal Of Uncertainity, Fuzziness and Knowledge-Based Systems*, 6(2):107–116, 1998.

[17] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780, 1997.

[18] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[19] Iowa-IFIS. Iowa Flood Information System. http://ifis.iowafloodcenter.org/ifis/. Visited on 06/11/2019.

[20] ISMN. International soil moisture network. https://www.hydrol-earth-syst-sci.net/15/1675/2011/hess-15-1675-2011.html. Visited on 06/11/2019.

[21] ISMN. International soil moisture network - data viewer. https://www.geo.tuwien.ac.at/insitu/data_viewer/. Visited on 06/11/2019.

[22] Jetbrains. Download pycharm. https://www.jetbrains.com/pycharm/. Visited on 06/11/2019.

[23] Jetbrains. Keras: The Python Deep Learning library. https://keras.io/. Visited on 06/11/2019.

[24] R. D. Knabb, J. R. Rhome, and D. P. Brown. *Tropical cyclone report: Hurricane Katrina, 23-30 August 2005*. National Hurricane Center, 2005.

[25] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl. Time-series extreme event forecasting with neural networks at uber. In *International Conference on Machine Learning*, number 34, pages 1–5, 2017.

[26] H. Luetkepöhl. Vector autoregressive models. European University Institute, Florence. Technical report, Department of Economics Working Paper ECO 2011/30, 2011.

[27] C. Massari, S. Camici, L. Ciabatta, and L. Brocca. Exploiting satellite-based surface soil moisture for flood forecasting in the Mediterranean area: State update versus rainfall correction. *Remote Sensing*, 10(2):292, 2018.

[28] N. Nicolai-Shaw, M. Hirschi, H. Mittelbach, and S. I. Seneviratne. Spatial representativeness of soil moisture using in situ, remote sensing, and land reanalysis data. *Journal of Geophysical Research: Atmospheres*, 120(19):9955–9964, 2015.

[29] NOAA. National Oceanic and Atmospheric Administration - About our Agency. http://www.noaa.gov/about-ouragency. Visited on 06/11/2019.

[30] NOAA. Storm surge and coastal inundation. http://www.stormsurge.noaa.gov. Visited on 06/11/2019.

[31] NOAA-FloodSafety. NOAA Flood Safety. https://www.weather.gov/safety/flood. Visited on 06/11/2019.

[32] NWS-AHPS. Advanced hydrologic prediction service. https://water.weather.gov/ahps2/index.php?wfo=sew. Visited on 06/11/2019.

[33] NWS-AHPS-ToolKit. Advanced hydrologic prediction service tool kit. https://toolkit.climate.gov/tool/advanced-hydrologic-prediction-service. Visited on 06/11/2019.

[34] NWS2014. Summary of Natural Hazard Statistics - 2014 in US. http://www.nws.noaa.gov/om/hazstats/sum14.pdf. Visited on 06/11/2019.

[35] NWS2015. Summary of Natural Hazard Statistics - 2015 in US. http://www.nws.noaa.gov/om/hazstats/sum15.pdf. Visited on 06/11/2019.

[36] C. Olah. Understanding lstm networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/. Visited on 06/11/2019.

[37] F. M. Ralph, M. D. Dettinger, M. M. Cairns, T. J. Galarneau, and J. Eylander. Defining atmospheric river: How the glossary of meteorology helped resolve a debate. *Bulletin of the American Meteorological Society*, 99(4):837–839, 2018.

[38] M. Rezaeianzadeh, H. Tabari, A. A. Yazdi, S. Isik, and L. Kalin. Flood flow forecasting using ANN, ANFIS and regression models. *Neural Computing and Applications*, 25(1):25–37, 2014.

[39] RiceUniversity-TMC. Flood Alert System. http://fas3.flood-alert.org/. Visited on 06/11/2019.

[40] P. Roy, P. Choudhury, and M. Saharia. Dynamic ANN modeling for flood forecasting in a river network. In *AIP Conference Proceedings, Portland, (Oregon)*, volume 1298, pages 219–225. AIP, 2010.

[41] N. W. Service. Summary of natural hazard statistics for 2015 in the United States. 2014.

[42] soilmoisture. Soil moisture types. https://www.drought.gov/drought/data-maps-tools/soil-moisture. Visited on 06/11/2019.

[43] Statsmodels. coint_johansen. https://blog.quantinsti.com/johansen-test-cointegration-building-stationary-portfolio/. Visited on 06/11/2019.

[44] R. Ul Islam, K. Andersson, and M. S. Hossain. A web based belief rule based expert system to predict flood. In *Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services*, page 3. ACM, 2015.

[45] USGS. USGS Automated Retrieval. https://waterservices.usgs.gov/rest/IV-Test-Tool.html. Visited on 06/11/2019.

[46] USGS. USGS Automated Retrieval Webservice. https://waterservices.usgs.gov/rest/IV-Service.html. Visited on 06/11/2019.

[47] USGS. USGS Build Time Series. https://waterdata.usgs.gov/nwis/uv. Visited on 06/11/2019.

[48] USGS. USGS Glossary. https://help.waterdata.usgs.gov/tutorials/surface-water-data/. Visited on 06/11/2019.

[49] VectorAutoRegressive. Package vars. https://cran.r-project.org/web/packages/vars/vars.pdf. Visited on 06/11/2019.

[50] J. P. Walker, G. R. Willgoose, and J. D. Kalma. In situ measurement of soil moisture: a comparison of techniques. *Journal of Hydrology*, 293(1-4):85–99, 2004.

[51] W. W. Wei. Time series analysis. In *The Oxford Handbook of Quantitative Methods in Psychology: Vol. 2*. 2006.